Providing Proactiveness

Sillitti, Alberto; Schomaker, Lambertus; Anakabe, Javier Fernandez; Basurko, Jon; Dam, Paulien; Ferreira, Hugo; Ferreiro, Susanna; Gijsbers, Jeroen; He, Sheng; Hegedus, Csaba

Published in:
The MANTIS Book

DOI:
10.1201/9781003339748-5

5

Providing Proactiveness: Data Analysis Techniques Portfolios

Alberto Sillitti[1], Javier Fernandez Anakabe[2], Jon Basurko[3], Paulien Dam[4], Hugo Ferreira[5], Susana Ferreiro[6], Jeroen Gijsbers[7], Sheng He[8], Csaba Hegedűs[9], Mike Holenderski[10], Jan-Otto Hooghoudt[11], Iñigo Lecuona[2], Urko Leturiondo[3], Quinten Marcelis[12], István Moldován[13], Emmanuel Okafor[8], Cláudio Rebelo de Sá[5], Ricardo Romero[6], Babacar Sarr[14], Lambert Schomaker[8], Arvind Kumar Shekar[15], Carlos Soares[5], Hans Sprong[7], Søren Theodorsen[12], Tom Tourwé[16], Gorka Urchegui[17], Godfried Webers[7], Yi Yang[11], Andriy Zubaliy[16], Ekhi Zugasti[2], and Urko Zurutuza[2]

[1]Innopolis University, Russian Federation
[2]Mondragon University, Arrasate-Mondragón, Spain
[3]IK4-Ikerlan, Arrasate-Mondragón, Spain
[4]Philips Consumer Lifestyle B.V., The Netherlands
[5]Instituto de Engenharia de Sistemas e Computadores do Porto, Portugal
[6]Tekniker, Spain
[7]Philips Medical Systems Nederland B.V., The Netherlands
[8]University of Groningen, The Netherlands
[9]AITIA International Inc., Hungary
[10]Technische Universiteit Eindhoven, The Netherlands
[11]Aalborg University, Denmark
[12]Ilias Solutions, Belgium
[13]Budapest University of Technology and Economics, Hungary
[14]3e, Belgium
[15]Bosch, Germany
[16]Sirris, Belgium
[17]Mondragon Sistemas De Informacion, Spain

5.1 Introduction

Data analysis is of paramount importance in the management of proactive maintenance. This chapter provides a deep analysis of the different techniques that can be adopted when dealing with the automation of maintenance processes. In particular, it focuses on three aspects of the maintenance that we have considered as the cornerstones of PM:

- **Root cause analysis:** aims to identify the causes of failures that have occurred in the past and avoid their appearance in the future. Basically, it provides a set of approaches intended at building a knowledgebase that relies on past experience to identify the core causes behind a problem in the system under investigation. This kind of analysis is useful when performing post-mortem analysis of the failures that have been reported and learn from them;
- **Identification of the remaining useful life:** aims to estimate the operational life of a component to support different activities including;
  - The proper design of a system based on the operational constraints and the expectations of the users to guarantee its usability;
  - The definition of a proper plan of maintenance activities to avoid unexpected down times.
- **Alerting and predicting of failures:** aims to identify possible failures before they actually happen and/or provide an alert about a set of still acceptable conditions that are unusual and that may result in a failure of the system if they are not managed properly in a timely fashion.

Besides these three main aspects, there are a number of additional ones that could be considered to create a comprehensive proactive maintenance environment. Since modern systems are very complex and taking decisions about maintenance requires taking into account many aspects, the decision-making approach needs the participation of multiple stakeholders and the implementation of collaborative decision-making strategies.

In a collaborative process, entities share information, resources and responsibilities, risks and rewards to jointly plan, implement, and evaluate a program of activities to achieve a common goal. Collaboration usually involves mutual engagement of participants to solve a problem together, which implies mutual trust. Coordination, that is, the act of working together harmoniously, is one of the main components of collaboration (Figure 5.1). In MANTIS, the common goal is the maintenance optimization of assets and the different systems and stakeholders that take part in maintenance tasks, will

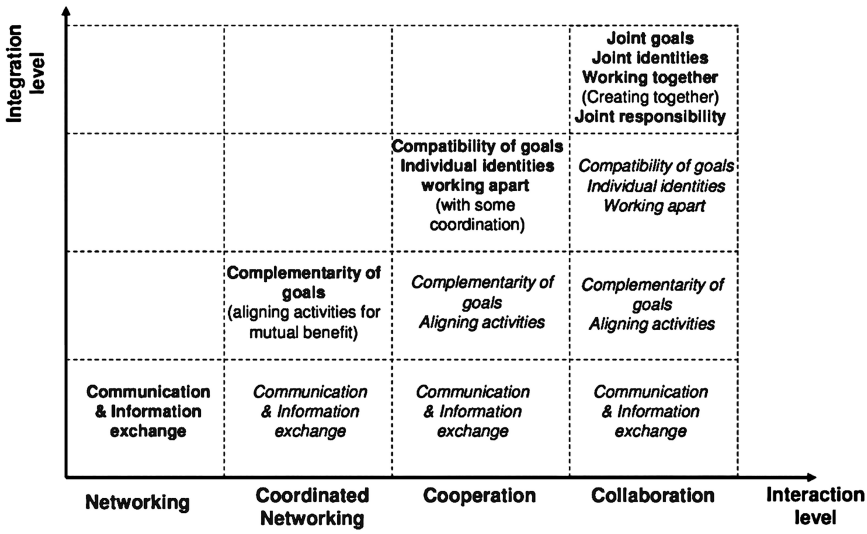


Figure 5.1 The collaboration concept.

have to share information, resources and responsibilities. From the process automation point of view, a key aspect of collaborative automation includes a single, unified environment for the presentation of information to the operator as well as the ability to present information in context to the right people, at the right time, from any access point within the system. This implies that the Proactive Maintenance Service Platform Architecture of MANTIS requires to collect data from all equipment regardless of type or age, store and organize the collected data over time, perform advanced analytics and provide decision support. Thus, MANTIS pursues a collaborative maintenance ecosystem.

Maintenance management can nowadays take advantage of many information sources for faster and more accurate prognosis. Data can be extracted and stored from machines and components. Machine based data analysis may be used locally to optimize the machine's maintenance. Better results may be expected if these particular data are combined with other data sources such as that from the MES, ERP, cloud-based open-data sources and the machine operators.

Proactive and collaborative decision support is an integral part of a proactive maintenance strategy. Condition monitoring is only one part of the equation, since the actual assets' condition needs to be combined with other relevant information, such as customer needs, ambient conditions, business models and service contracts in order for a service team to make the right

decision and take the appropriate actions. The goal of MANTIS is to provide the service team with an intuitive, proactive and context-aware system for industrial maintenance that supports maintenance by proactively pushing relevant information to the right people at the right time, by intelligently filtering and summarizing information to prevent information overload through context awareness, by automatically and dynamically scheduling and adapting maintenance plans, thereby keeping the human in the loop at all times. The chapter is organized as follows:

- Section 5.2 focuses on root cause analysis providing a short introduction to the theoretical background and a catalogue of techniques that have been demonstrated to be useful in this kind of analysis. Such techniques are mainly based on statistical and machine learning approaches. Moreover, a set of real-world applications are shortly introduced;
- Section 5.3 deals with the identification of the remaining useful life of components. The section provides a short theoretical background, a catalogue of the useful approaches, and an analysis of some use cases to demonstrate the applicability of the described techniques analysing different modelling approaches;
- Section 5.4 investigates how to alert and predict failures. The section provides an extensive catalogue of useful techniques. The presented techniques are also mainly based on statistical and machine learning approaches;
- Section 5.5 provides some real examples that come from different application domains where most of the techniques previously presented have been applied with valuable results.

5.2 Root Cause Failure Analysis

RCA is a methodology to identify the primary cause of a failure in a system. RCA is the function that makes PM possible, detecting and correcting root conditions that would otherwise lead to failure. Once the root cause is identified, corrective action can be taken to make sure that the issue does not re-occur.

5.2.1 Theoretical Background

Data-driven RCA uses data and data analysis techniques to identify root causes, based on the observable states of the system. These observable states may be directly or indirectly related to identifiable components in the system. Based on the design of the system, it is necessary to define the relevant

features in the available data and to identify the components that are most likely involved in the failure. The latter may require, for example, expert domain knowledge, historical maintenance data and historical performance data. If data analysis reveals that a single component caused the (majority of) issues, then the RCA is complete; otherwise the RCA is re-iterated with an updated set of features and/or components.

The CRISP-DM defines a methodology for structured data mining. For RCA, the following phases are used: business understanding, data understanding, data preparation, modelling.

Business understanding

Due to the size/complexity of systems and the amount of available data, it is necessary to narrow down the scope of RCA data analysis a-priori. Expert domain knowledge is required to scope the RCA in terms of machines, failure modes, parts and data sources. This is an iterative approach; insights in the data helps the expert to refine the scope. When the system is complex, expert knowledge may not be adequate to identify a-priori the relevant system components and data features. Probabilistic graphical models could, for example, be used to identify the most probable components and features.

Data understanding

There is a wide variety of visualization and statistical techniques to provide an overview of the data. Graphs and statistical characteristics are instrumental in understanding the data (-distributions) and to identify the relevant features in the data in the next analysis phases. Examples of data visualization are PCA and t-Distributed Stochastic Neighbour Embedding. These techniques can provide insights of the distribution and classification of data in the multi-dimensional feature space. Examples of statistical techniques are forward/backward selection, trend extraction, and classification. The purpose of these techniques is to reduce the multi-dimensional feature space to a manageable number of features, which represent the system adequately for RCA purposes.

For specific RCA cases, the influence of features evolves over time while the system evolves. This adds time as an additional dimension. There are statistical techniques that include implicitly time as dimension, but this raises the risk that the statistics get biased by the more recent data.

Data preparation

The collected data usually originate from a variety of data sources. These can comprise machine data, e.g., sensor data obtained by sampling in real-time,

or machine logs that represent the internal condition of the system. Service maintenance records are another source of data. These data contain, for example, specifics of parts replaced (quantity, quality). Each data source may require specific pre-processing and transformation techniques before it can be used by data analysis algorithms. Examples include aligning timestamp formats, correcting for missing data and erroneous data and the aggregation of noisy data.

Modelling

Once the data features are identified, it is possible to develop algorithms (models) that identify the specific components -which are the most probable root cause of a specific issue - and the failure modes. The input of the algorithms are the data, which are prepared specifically for each algorithm. The output of the algorithm is a statistical measure that can be used to identify the specific component as the root cause.

5.2.2 Techniques Catalogue

There is a wide variety of modelling techniques available. Table 5.1 shows an excerpt of techniques.

Based on research of practical systems, a selected set of algorithms is described in the next sections. For each algorithm, the main aspects are outlined, and some practical examples are given. Some of the described algorithms are used for RCA only; other algorithms are used for root cause analysis and for failure prediction (Figure 5.2). Some of the examples illustrate how to embed the algorithm in the system, while other examples illustrate how to the use the algorithm outside the system.

Table 5.1 Root Cause Analysis Techniques

Modelling technique	Algorithm
Classification	Random forest classification, Naive Bayes, Support Vector Machine (SVM), Limit and trend checking
Regression	Partial least square regression (PLS), Random forest regression, Least absolute deviations regression, Logistic regression, Bayesian network (BN)
Neural network	Artificial Neural network, recurrent neural network, convolutional neural network
Unsupervised learning	Hierarchical clustering, K-means clustering, attribute-oriented induction
Pattern analysis	Hidden Markov model (HMM), expectation maximization,

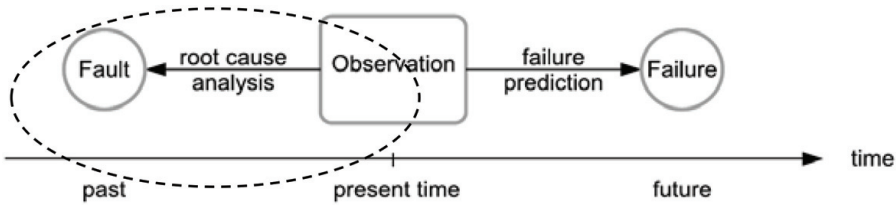


Figure 5.2 Overview for RCA.

5.2.2.1 Support vector machine

A SVM is a supervisory learning algorithm that can be used for binary classification. It uses feature data as input and it calculates the most probable class. This algorithm is trained by providing feature data that is already classified, i.e., labelling. The algorithm then constructs a hyper plane or a set of hyper planes based on linear combination of features in the multi-dimensional feature space and optimizes the distance between the feature values and the hyper plane(s). For some problems, it may be required to first linearize the feature values by a (non-linear) transformation. Once the hyper plane(s) have been constructed, the algorithm is applied to a test dataset. The percentage of correctly classified cases is a measure for the performance of the algorithm.

SVM is capable to handle high-dimensional feature spaces adequately and is insensitive to outliers in feature values. Domain expertise can be embedded in the transformation of feature values. Furthermore, SVM has proven its use in numerous application domains, such as text mining, face recognition, and image processing.

5.2.2.2 Limit and trend checking

Limit and trend checking are a classical algorithm where (derived) feature values are compared against limits (one-sided or two-sided). Feature values that are within the limits represent normal behaviour of the system, feature values that are outside the limits can be caused by defective component(s). Limit and trend checking are commonly integrated in process automation systems. These algorithms are in many cases sufficient to prevent larger failure or damage. However, faults are detected rather late and a detailed component diagnosis is mostly not possible. The limits can be based on design and/or historical data and this represents, in general, a trade-off between early detection (narrow limits) and inherent statistical variation of the system (wide limits). Limits may be fixed values, may vary in time or may vary with

other feature values, depending on the characteristics of (components of) the system.

Example: press machine

In a press machine, the forces in the ram (which consist of 2 rods) are measured during the stamp. A misalignment (eccentricity) in the rod forces creates undesired loads in the press and may lead to premature wear and future malfunction of the press. Piezoelectric sensors measure the force in the press rods. During each stroke, the eccentricity is calculated and compared against limits. When the eccentricity is outside limits, a maintenance action is triggered by the press.

Example: photovoltaic plant

For a photovoltaic plant, the performance loss ratios are determined by each energy conversion component. These ratios are aggregated for a given time span, e.g., one day up to several months. These ratios are compared against configured limits. The limits are derived from the technical specifications of the components in the system and from field measurements of "known good" photovoltaic plants. Furthermore, the limits are corrected for weather conditions in the given time span. When the performance is outside the limits, a maintenance action is triggered.

Example: health equipment

In a health equipment, the ambient conditions are monitored to make sure that the system operates within its operational limits. Sensors continuously measure ambient temperature, humidity and cooling water temperatures. The health equipment compares these values against configured limits. When the ambient conditions are outside the limits, a maintenance action is triggered by the health equipment. The (wide) limits are based on the design of the health equipment. The limits can be localized (narrowed) using the configuration capabilities on the health equipment, if local environment operational conditions justify this.

5.2.2.3 Partial least squares regression

PLS is an algorithm that constructs a hyper plane or a set of hyper planes based on linear combination of features in the multi-dimensional feature space and minimizes the distance between the feature values and the hyper plane(s), based on mathematical projections to a lower-dimensional feature space. The weight of a feature in the linear fit is a measure of the relative

impact of the component, associated to that feature. The higher the weight, the higher the detection capability of a defective component.

The upside of this algorithm is that it can handle a large (more than 10) number of features, it can handle correlated features effectively, and it can handle missing data points. The downside is that the output of the algorithm may be difficult to interpret, for example confusing causality with co-linearity.

PLS was initially developed for econometrics and chemo-metrics. Since then, it has been applied in numerous domains, such as education, manufacturing, marketing and social sciences.

Example: shaver production plant

In a shaver production plan, the product quality is measured and is expressed as 4 geometry features (Y). The production line quality is measured and expressed as 12 sensors features (X). Product features are measured on a sample-basis; production line features are measured continuously. A unique product identifier is used to join the product features and the production line features, furthermore, short terms sensor failures are removed from the production line features. The dataset was fed into the PLS algorithm where production line features are the independent variables (X) and the product quality features are the dependent variables (Y). This provided the weighting factors and the root mean square error. It was concluded that the root mean square error is acceptable for each of the product features, given the acceptable bandwidth of product quality. This enabled the shaver production plant to implement the algorithm in the production line to monitor product quality in real-time.

5.2.2.4 Bayesian network

A BN is a probabilistic directed acyclic graph of nodes where the nodes represent observable stochastic variables of a system. Such graphs can be used to calculate the probability distribution of specific conditions or states, where multiple nodes interact. More specifically in the case of RCA, nodes can represent machine components and the node state can represent the failure modes and non-failure states of the corresponding components [3]. One can therefore calculate the probability of a state (for example failure or no failure) of a component based on the states of the other dependent nodes (marginal probability). The process of building a BN graph uses expert domain knowledge to identify the primary components and their interactions. In general, many components work independently and their failure does not

propagate to all other components. This simplifies the BN and allows for computation that is more efficient.

Common Applications:

Because the BNs are a general probabilistic inference method, it can be applied to a wide range of problems where objects can be assigned a probability of being in a state. Examples of application include:

- Medicine (diagnosing disease or illness based on symptoms);
- Document classification (classifying a document belonging to a given subject based on the word content);
- Image processing (for example assigning a pixel to a region based on its colour components and the region membership of its surrounding pixels);
- Spam filters (determining if a message is spam or not);
- Biology (inferring the network), etc.

Strengths and challenges:

The BN model is a very good fit for the RCA problem. It is only necessary to identify the components and their failure modes. The components are then mapped to the nodes and the states of each node are used to hold the conditional probabilities. The problem with constructing this model however is twofold. The first is determining the state dependency between each pair of components (establishing the network) and the second is acquiring enough failure data for each and every component (conditional probabilities). This requires good domain knowledge and much effort. Since it was used a very small model (modelled only fail or non-fail states), calculating the marginal probabilities could be done efficiently.

- Advantages:
    - Intuitive model construction;
    - Allows for on-line model updates;
    - General model applicable to many problems.
- Disadvantages:
    - Difficult to automatically infer the network;
    - Requires detailed domain specific knowledge to construct the network;
    - Has high computational costs.

Example: sheet metal machinery

One of the subsystems in a sheet metal machine is the hydraulics subsystem. Domain expertise is used to define a set of hydraulics failure modes, which are easy to understand and easy to describe. To further reduce the feature dimensional space, it is opted to use a binary representation of the state of the hydraulics subsystem: healthy or faulty. As there was no live data available from a working sheet metal machine, a BN simulation was built using hypothetical probability distributions and using a limited set of components, such as motors, encoders, actuators, sensors, valves. Each node in the graph represents one of these components. An interactive User Interface was developed to select the state of each node; the BN then calculates the failure probability distribution of each of the nodes in the network. The node with the highest probability is the most likely defective component. This interactive tool can be used to obtain a list of most probable defective components, based on real observations.

5.2.2.5 Artificial neural network

An ANN is a collection of computational units called artificial neurons that mimic the human brain cells. Each neuron performs a weighted summation of its input values and applies an activation function, such as ReLU, sigmoid, Softmax. The output of a neuron is input to other connected neurons. The neurons are organized into layers: an input layer where the input values are fed into, the hidden layers that perform the learning process and the output layer that performs the final decision making process. The variability in number of layers, number of neurons and the activation functions results in a wide variety of possible ANN architectures. Similarly to SVM, the ANN algorithm is trained using a training data set and is validated using a test data set.

In recent years, the advances in computation power and effective implementations have enabled ANN to learn through large datasets and in numerous application domains.

Example: health equipment

To research predictive maintenance capabilities for healthcare equipment, ANN was used to predict part replacement, with ANN training based on historical machine data. Historical part replacements were extracted from service business data. The matching machine errors in an observation window of 14 days before (failure sequence) and after the part replacement (non-failure sequence) was retrieved from machine logs. The machine data were

grouped into features and represented as 1-dimensional and two-dimensional images and provided as input to the ANN. To test the influence of the chosen ANN architecture, five deep learning architectures were tested: artificial ANN, 1D convolution ANN, 2D convolution ANN, LeNet ANN, LSTM. For each architecture, the ANN is trained using 70% of the feature values while the remaining 30% is used to test the ANN. To judge the performance of each architecture, the percentage of correctly predicted part replacements is determined. All the above-mentioned ANN architectures yielded a percentage between 50% and 70%. This percentage does not significantly increase when doubling the number of features or doubling the historical period of time.

Example: metal cutting machine

To research predictive maintenance capabilities for a metal cutting machine, ANN was used to predict the texture classification of machined products. The hypothesis is that machine defects may lead to surface defects in the machined products. The surface could be measured using cameras in the production line and the resulting images could be inspected in real-time. A ANN is a well known algorithm to classify image data. For that purpose, a convolution neural network has been built and trained. The Northeastern University, or NEU, surface defect database, publically available, was used to train the ANN. This database contains labelled images that cover multiple texture classes and contains hundreds of images per texture class. Each texture type is related to a certain, known, root cause. The ANN was tested using another dataset, containing images from machined products. The ANN yielded a performance of 95% of correctly classified textures. This is a good starting point to further explore how to improve maintenance of the metal cutting machine, based on visual inspection of metal surfaces.

5.2.2.6 K-means clustering

The purpose of clustering is to group features in such a way that features of elements inside one cluster are more similar to each other than to the elements in other clusters. Clustering is useful to gain insights on the internal structure in data. In the clusters, it is easier to detect recurrent patterns and underlying rules. K-means clustering is a widely used algorithm that groups data points around centroids or means. The algorithm iteratively determines the centroids by minimizing the distance between data points and the centroids. The number of iterations depends on the dimensions of feature-space and the number of means. There is a variety of strategies how to select the centroids best, e.g., farthest point selection, K-means++.

Example: press machine

In a press machine, the acceleration of the ram is measured during stamping. The acceleration is a measure for the vibration state of the press. The frequency spectrum is calculated from the accelerometer data. The peaks in the spectrum are identified using a peak detection algorithm. Data of multiple ram cycles are fed into a K-means cluster algorithm. The K-means algorithm is designed to detect the highest peak clusters in the frequency spectrum, covering the different press operating conditions. The higher the peak, the more it can contribute to a vibration induced failure. From the data analysis, it was concluded that 4 main peaks in the frequency spectrum can be clearly identified and can be related to excessive vibrations in the press. This knowledge is an enabler to implement this algorithm in the press machine in real-time and to trigger a maintenance action when if required.

Example: special purpose vehicles

To research predictive maintenance capabilities in special purpose vehicles, log messages from hundreds of these vehicles have been analysed. The data set contains log messages from multiple vehicle types, multiple customers and multiple service providers. Each vehicle produces a time-series of log messages for what is considered as a stochastic process. The K-means algorithm is designed to detect log message patterns as function of vehicle type, customer and service provider. It was concluded that there is a strong relation between log message patterns and vehicle type but that there is no relation between log message patterns and service provider. Depending on the clustering size, there is a weak relation between log message patterns and customer; this indicates a weak impact of vehicle utilization.

5.2.2.7 Attribute oriented induction

AOI is considered a hierarchical clustering algorithm for knowledge discovery in databases. Specifically, it is considered a rule-based concept hierarchy algorithm, due to the fact that the representation of the knowledge is structured in different generalization-levels of the concept hierarchy. The execution of the AOI algorithm follows an iterative process where each variable or attribute will have its own hierarchy tree. Later, data must change from one generalization-level to another, generalizing all the data in the dataset. That step is denoted concept-tree ascension. AOI is an algorithm whose power resides on defining thresholds and generalization hierarchies for the attributes of the data. This means that it is an algorithm where domain expert feedback is very useful. Basically, AOI must be trained first with healthy data, and

build the base tree using these healthy data. Later, in the test part it is checked whether the evaluated data creates the same tree or it creates a different one. If the data creates the same tree, the asset can be considered to be healthy; if new instances are created, the asset had probably some issues in the meantime.

This would be the first level of maintenance: the anomaly detection. In order to provide RCA, there should be a database of each type of damage, to allow to correlate the cluster apparition with the damage itself.

Example: clutch brake machine

AOI was successfully applied to a clutch brake machine monitoring use case. Brake pads are consumable parts of the clutch brake, and it is important to know when they wear out. Historical data from a clutch break machine was extracted. These data comprise information of pressure, rotating speed, air pressure and trigger, of a clutch brake both with complete brake pads and with worn out brake pads. First, using information provided by the domain expert, the generalizations of the AOI algorithm were established. Using data with the healthy clutch brake, the AOI algorithm was trained. Later, using the test data, the anomaly detection part of the AOI was used in order to model the wear of brake pads. Finally, thanks to the wear model, the solution was able to provide a RUL estimation for the asset.

5.2.2.8 Hidden Markov model

A HMM is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved (hidden) states. The HMM is represented by a collection of states, state transitions probabilities and observations. It is the task of the HMM model to calculate, in the best way, the probability for a particular sequence of observations, without prior knowledge of the states. Similarly to other algorithms, such as SVM and ANN, the HMM is trained using a training data set and is validated using a test data set.

Example: off-road and special purpose vehicles

To research predictive maintenance capabilities for forklift trucks, HMM was used to predict the probability of part replacement, based on historical machine data. Historical part replacements were extracted from service technician logs, based on text keywords. The matching machine data in an observation window of 30 days before the part replacement data (failure sequence) were retrieved from machine error logs. Machine errors after a part replacement (non-failure sequence) were also retrieved from machine

error logs. These data sets were joined and fed to the HMM algorithm to train it. The performance of the HMM algorithm was determined using a test data set, randomly selected failure sequences and non-failure sequences.

5.3 Remaining Useful Life Identification of Wearing Components

The RUL is the useful operational life left on an asset at a particular instance in time. Depending on the scientific or engineering field, RUL definitions can also include the *usable* or *productive* keywords, when defining the end of the asset useful life [ISO 13381-1, 2004].

5.3.1 Theoretical Background

An essential question for design and operation of technical components is their expected useful lives. The estimation of useful life is important for designers and manufacturers for design improvements and marketing purposes, as well as product users for making decisions on which product to buy. Furthermore, the estimation of remaining useful life after the product has been put into operation is important for making decisions on maintenance, repair, reinvestment and new investments. RUL estimation is also important in relation to possibly existing service agreements between the users and manufacturers and possible guarantee periods. In this sense, the estimation of RUL of any asset is the first step towards CBM. A RUL prognostic model will typically provide at least two components: the RUL estimation and an associated confidence limit. Models that are more informative provide for instance an estimate of the failure distribution function.

5.3.2 Techniques Catalogue

Various RUL models classification systems exist within the scientific literature. For instance [Si et al., 2011] classify RUL prediction models into two main types depending on whether they use: 1) direct condition monitoring data, or 2) indirect condition monitoring data. The first type is then divided into models which model the state evolution as a continuous process or alternatively as a discrete state space; while the second type is divided into the subcategories: a) filtering type of models, 2) covariate-based hazard models, and 3) hidden Markov model based methods. [Welte and Wang, 2014] classify RUL prediction models into physical, stochastic, data-driven and artificial

Table 5.2 RUL techniques categories

Classification	Techniques
Physical modeling	Application specific
Artificial Neural networks	RUL forecasting, Parameter estimation
Life expectancy models	Trend extrapolation, Auto regressive mean average methods, Proportional hazard models, reliability function, (Hidden) Markov models, Kalman filter, Particle filters
Knowledge based models	Expert systems, Fuzzy rules

intelligence model, while [Sikorska et al., 2011] categorize models into four main groups and a varying number of subgroups where the latter consist of the various RUL techniques. The classification system of Sikorska et al. is applied in the setup of this section and Table 5.2 shows the four main classification categories together with the techniques that fall within each group.

5.3.3 Physical Modelling

The application of RUL techniques depend on expert knowledge that is distilled into a model of the physical asset being evaluated. The model families used in this context can differ between applications, and a few selected families are described in this section.

5.3.3.1 Industrial automation

Manufacturing defects combined with severe working conditions and lack of maintenance, accelerate structural damage in the press head that can lead to failures in the form of fractures in the structural components of the press. The structural components, such as the press head, are typically welded steel parts. The welds are one of the sensitive parts where cracks can initiate because of notch stress concentration, residual stress and base material properties degradation.

Crack growth can be divided into three stages: initiation, stable propagation and fracture after an unstable or fast propagation. The first stage is difficult to predict and difficult to detect during regular maintenance service by traditional methods. Cracks are usually only detected when large enough to be visually localized during inspection. In such cases, a corrective maintenance action is taken to repair the failure. For this application, two degradation models have been used to predict the RUL: classical high cycle fatigue damage and crack propagation according to Paris' law.

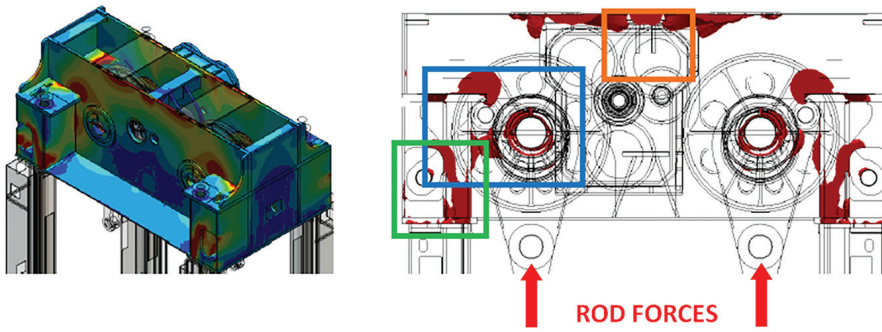


Figure 5.3 Press machine head critical zones and measured forces at the rods.

High cycle damage: RUL to crack initiation

Classical high cycle fatigue is used to estimate the RUL to a certain threshold damage value, assuming that it is associated with a predefined initial crack length. Three methods oriented to welded structures recommended by the [International Institute of Welding, 2008] are used to evaluate the damage. The steps consist of 1) the stress evolution is calculated by Finite Element models applying the real forces measured at the press rods, see also Figure 5.3, 2) once the stress time history is calculated, the stress cycles during operation are obtained with the standard rain-flow cycle counting algorithm, then 3) the fatigue damage due to n_i constant stress cycles $\delta\sigma_i$ is obtained by the corresponding structural detail SN curve: $D_i = n_i/N_i$, where N_i indicates the maximum allowable stress cycles of range $\delta\sigma_i$ that a structural detail can withstand before failure for a given S-N curve characterized by C and m (slope). The SN-curve can have a different slope depending on the stress range, that is $N_i = C/(\delta\sigma_i)^m$. The total damage D due to different stress cycles is calculated according to Miner's rule, $D = \Sigma D_i = \Sigma n_i/N_i$.

Finally, the RUL, considered as the estimated cycles (time) to the end of the first stage (crack initiation), is estimated at each critical zone setting up a damage threshold and calculating the remaining cycles. A fatigue damage indicators map is created in the studied component in order to identify the most probable crack initiation locations due to the real forces history applied in the press.

Crack growth, Paris' law and particle filters

During the stable propagation stage, in the case of one-dimensional fracture, the crack growth rate is governed by a power law such as Paris' law.

This model gives a measure of the crack growth rate proportional to the stress intensity factor

$$\frac{\mathrm{d}a}{\mathrm{d}N} = c(\Delta K(a))^m, \tag{5.1}$$

with a the crack length, N the number of stress cycles; C and m material specific parameters ΔK=K$_{max}$-K$_{min}$, the stress intensity factor, which is a function of the applied stress (load) range $\Delta\sigma$, $\Delta K(a)=Y(a)\ \Delta\sigma\ (\pi a)^{1/2}$, and $Y(a)$ the geometry function taking into account the geometry of the surrounding of the crack.

Under the hypothesis of a crack of certain length started at a critical zone of the structural component, it is possible to estimate when the crack will reach a threshold length for a given loading. If experimental data on crack length propagation are available, the estimation of the RUL can be improved combining the measurements and the physics based crack propagation model. In this case, a particle filter technique has been applied to obtain the RUL. The technique handles the error associated to the measurements and to the model and updates the RUL, and it is supplemented with an interval of confidence every time a new observation becomes available.

In Figure 5.4, RUL estimation based on real data from a component under constant stress loading is shown. The initial crack size is 14 mm and the maximum crack size is set to 32 mm. The observed crack propagation is given by the blue line. An initial RUL estimation is done from the initial crack size based on the physical model (blue dotted line).

5.3.3.2 Fleet's maintenance

Within MANTIS one of the goals is to find ways towards predictive fleet maintenance. With respect to RUL prediction, the main goal is to develop a method to optimize the maintenance protocols for defence vehicles and other types of complex machines, such that maintenance paradigms can be modified from a "time-of-usage" and a "number-of-driven-kilometres" approach to an approach taking into the account the severity of usage during operation. Thereby, the predictive power for when maintenance is truly necessary to be performed on the vehicle in order to prevent failure is increased.

An algorithm for determining the remaining useful life for a vehicle as one whole unit is introduced here. The goal of the algorithm is to estimate the wear induced on a vehicle by surface induced vibrations. The wear is quantified by calculating a wear index that is weighted with both the magnitude and the duration of an excitation. Damages due to the long-term use of a structure are typically associated with fatigue failure. This failure mode occurs when a

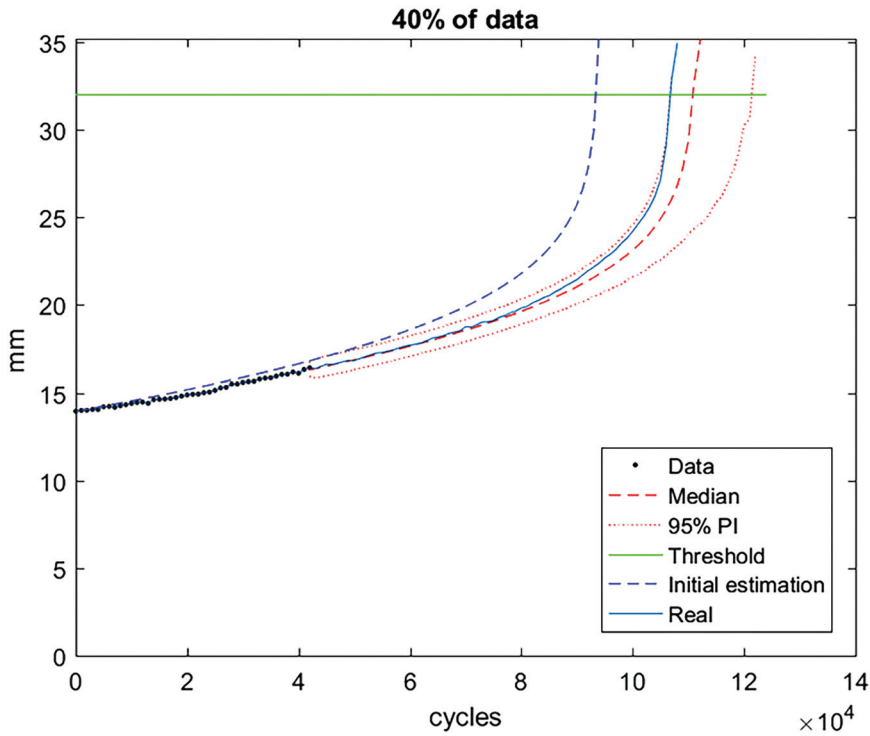


Figure 5.4 RUL calculation at 4.5 104 cycles.

structural member is exposed to a repeated loading a critical number of times. The fatigue strength of a certain material is often represented by an SN-curve. An SN-curve shows the mean number of constant stress-cycles it takes to break the material versus the applied, constant, stress magnitude. The critical number of cycles as a function of the applied constant stress level often follows, approximately, an exponential function of the form: $N_i = (S_0/S_i)^m$, where N_i is the critical number of cycles, S_0 a constant, S_i the constant stress magnitude applied, and m an exponent. In real-life, structures are exposed to excitations of varying amplitudes. Cycle counting then yields a histogram of cycles binned according to amplitude range. In this case, to compute the cumulative damage due to the cycles with various stress levels, Miners rule is applied, which states that:

$$D = \sum_{i=1}^{k} \frac{n_i}{N_i}, \tag{5.2}$$

with n_i the number of cycles with stress level S_i; N_i denotes the number of cycles after which (approximately) failure will occur when a constant stress level of S_i is applied and k denotes the number of discrete stress levels possibly applied to the structure. When $\mathrm{D} > 1$ it is typically assumed that the structural member is expected to fail.

The concept behind the developed RUL algorithm is to estimate the wear of a vehicle to surface excitation in a manner similar to the Miner's rule damage estimation described above. However, as it is not feasible to measure the applied stress directly, as a measure of the stress the force applied to the structure of the rigid vehicle frame is used. To that purpose, the acceleration of the vehicle is measured with a 3D accelerometer sensor, and the amplitude of the force follows then directly from this measurement. The overall concept has some challenges. First of all the wear has to be calculated for the vehicle as one unit, although it consists of multiple structural members. Further, the loads cannot be measured directly, and instead the measured parameters are the accelerations. Finally, no parameters for the SN-model are available, moreover, these parameters are also expected to depend on other factors such as the location of sensor on the vehicle, the suspension system and the vehicle type.

To overcome many of these challenges a *position translation algorithm* has been developed. The *position translation algorithm* transforms the observations made at one location of the vehicle, the physical position of the sensor, to find the impact at another location of the vehicle, the physical position of the desired place of impact for the RUL computation. The so-called *Inertial Measurement Unit* measures translational accelerations and rotational velocities, and is located in the moving reference frame mounted on the vehicle frame close to the centre of gravity of the vehicle. Denoting the reference frame by x' y' z', the task is to transform the data measured in the reference frame to a similar aligned reference frame x" y" z" defined by displacing the reference frame by a displacement vector s_p'. The prime in s_p' indicates that the displacement vector for the new double primed reference frame is written in terms of the primed vehicle fixed system. Denoting the angular rotation vector as ω' =[ω_x', ω_y', ω_z']^T, which is the vector filled with data from the *Inertial Measurement Unit*, then the transformation of the translational accelerations from the primed to the double primed system are governed by the following relation $\ddot{r}_P^{'} = \ddot{r}^{'} + \omega_M^{'} s_P^{'} + \omega_M^{'} \omega_M^{'} s_P^{'}$, in which

ω_M is the so called skewed symmetric matrix defined through ω as

$$\omega_M = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{5.3}$$

The numerical differentiation of the angular velocities is necessary to estimate the angular acceleration of the primed system.

5.3.3.3 Eolic systems

A degradation model is applied to predict the RUL of REBs in wind turbine gearboxes. The model can be viewed as a hybrid of a statistical model and a qualitative physics-based model. The objective of qualitative physics-based models is typically to select a proper damage indicator to construct a statistical model. The statistical model can then be fitted to the damage indicator data. Prior knowledge concerning probable values for the model parameters can be incorporated into the model by applying a Bayesian approach. By gathering an increasing amount of on-line monitoring data over time, the uncertainty related to the value of the model parameters decreases by applying a Bayesian updating method. In parallel the probability density function describing the remaining useful life of the REBs, which uses the parameter estimates, is updated accordingly.

Implementation

Among four vibration features extracted from the full frequency domain vibration spectrum, the so-called High Frequency Peak Value was selected as the damage indicator to apply for the diagnosis and the prognosis of REBs. An exponential statistical model with multiplicative error terms is used, originally proposed by [Gebraeel, 2005] and [Gebraeel, 2003]. The mathematical form of the model is $S(t_i) = \theta \exp[\beta t_i + \varepsilon(t_i)]$, in which the parameters θ and β are assumed to follow, respectively, a log-normal and a normal distribution. Generally, the logarithm of the raw vibration signal $S(t_i)$ is taken to fit the model to the data.

Given a specific time t_k, RUL denoted by T is defined as the duration from t_k to the critical time reaching the critical threshold denoted by D. The probabilistic distribution of T is equivalent to the logarithm of High Frequency Peak Value to reach D, and can be expressed by: $P\ (T < t|L_1,L_2,\ldots,L_k) = P\ [L(t_k+ T) = D|L_1,L_2,\ldots,L_k]$, with L_1, L_2,..., L_k the logarithm of the observations. The data related to a number of REBs was used to infer prior information for both the model parameters and the pdf of RUL. For each of

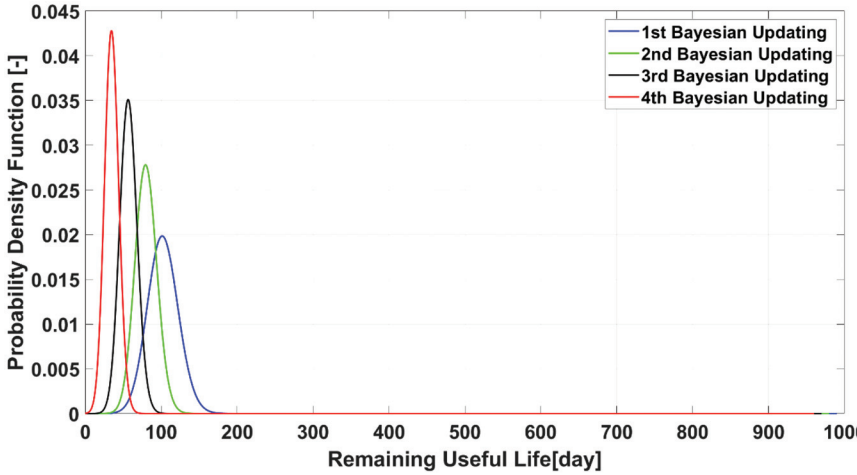


Figure 5.5 Illustration of Updated pdf of RUL for a REB.

these REBs, part of the signals was approximated by an exponential trend, over a period from the start-up of increase in vibration signal amplitude (day t_s) to the occurrence of cut-off (day t_e), and used to fit the statistical model. The statistics of θ and β of this specific REB is estimated based upon the other N-1 REBs, also known as leave-one-out cross validation, and is used as the prior distribution of θ and β for this specific REB. Bayesian updating is performed four times with the interval of 10 days (e.g., first Bayesian updating performed at Day (t_s+10), and so on), with the updated pdfs of one specific REB illustrated in Figure 5.5.

5.3.3.4 Medical systems

The activity concerns the wear of cathode filaments in X-ray tubes that are used in interventional X-ray systems. X-ray tubes are the most expensive replacement parts of an interventional X-ray systems and therefore of major concern for the service organization. Because of the major impact, in terms of downtime and costs related to an X-ray tube replacement, it is important for Philips to be able to predict an upcoming failure of X-ray tubes accurately in order to provide in time replacement. A failure analysis was performed and the dominant failure mode turned out to be a blown cathode filament.

X-ray tube cathode filaments are heated to a high temperature during operation, in order to emit sufficient electrons to produce the desired X-ray dose. The high temperature, however, also makes the Tungsten—of

which the filaments are made of—to evaporate. Thereby, a hot-spot forms locally at a certain location, where the generated heat and the evaporation increase exponentially over time, until the material melts at the hot-spot resulting in the opening of the filament, see also [Webers et al., 2016] and [Horster et al., 1971]. The physics of failure of the X-ray tube is very similar to the physics of failure for incandescent lamps [Rice, 1997]. Understanding the stress applied and having a damage indicator at hand are two prerequisites for building a physical model for RUL prediction of the filament. From the previous literature and experience, the filament stressor (which is the filament temperature) and a proper damage indicator (which is the filament resistance) are known. However, monitoring the damage increase over time by simply plotting the filament resistance as a function over time is problematic for two reasons: 1) individual interventional X-ray systems are daily used with various settings, meaning that is different filament currents are applied from run to run, implying that time is not a proper aging factor to apply as independent variable. 2) The filament resistance cannot be measured directly.

The first issue is dealt with by using a method similar to Miner's rule [10]. Instead of using the accumulated run time, an accumulation of linear damage (W) is used

$$W = {}^{\mathrm{X}}\Delta W_i \tag{5.4}$$

with the sum applied over the total number of runs, and

$$\Delta W_i = \frac{t_{ij}}{T_j} \tag{5.5}$$

where $\mathrm{t}_{i,j}$ is the time duration of run i, carried out with an applied current of A_j, $\mathrm{j} = 1, \ldots, \mathrm{k}$, and k the total number of possible Ampere values to be applied; and T_j the mean lifetime for a filament when a constant current A_j is applied to it.

The second issue is overcome by using the *c-factor* as a damage indicator rather than the filament resistance. The relationship between the resistance and the c-factor is given below. If the resistance of the filament at a particular temperature is R_0 at the start of use, while after being used for i runs the resistance at the same temperature has changed to R_i, then the c-factor for that run is:

$$c_i = \sqrt{\frac{R_0}{R_i - R_0}} \tag{5.6}$$

The value of $c_i$can be derived for each run, without involving measurement of R_i. Two small modifications are made to the two variables in order to

use them for monitoring the degradation over time. For the ΔW calculation, the mean lifetime T_j— mean lifetime for a filament when a constant current A_j is applied to it— is not used. In fact, the sum is divided by the mean life time for a current A_j/c_i. Without going into detail, by dividing the applied filament current by the c-factor, the accumulation of wear per unit time remains constant if the user keeps on using the system in the same way, and the new variable is referred as linear wear (*LW*), that is

$$LW = {}^{X}\Delta LW_i \tag{5.7}$$

with the summation over all runs *i*, and

$$\Delta LW_i = \frac{t_{i,j}}{T'_j} \tag{5.8}$$

with T'_j the mean lifetime for a filament for which a constant continuous current A_j/c_i is applied. The need for this modification is a consequence of using a constant heating current during the lab experiments rather than constant heating power, which would have resulted in a constant filament temperature during a single experiment. In Figure 5.6, the logarithm of the c-factor is plotted as a function of linear wear for lab data.

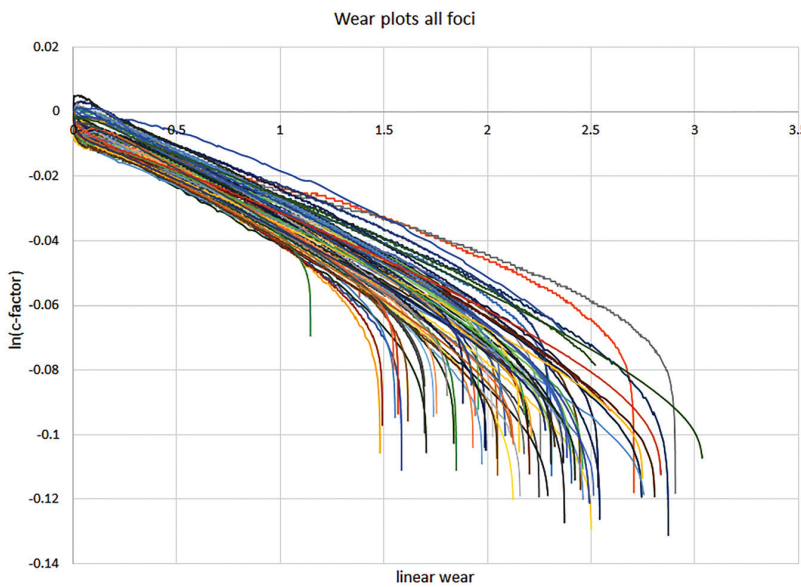


Figure 5.6 ln(c-factor) vs. linear wear from lab experiments.

It was observed that the slope of the linear part of the curve and the slope at time of failure are correlated, meaning that the relative large initial negative slope implies relative large negative slope at time of failure. This observation was quantified and incorporated in the final RUL estimation method.

The RUL estimation method entails now roughly the following steps

- For the linear part of the curve a simple linear regression is used to determine the slope of this segment. For the curved segment a third order polynomial fit is applied, leading to *c-factor*= $a\ LW^3 + b\ LW^2 + d\ LW + e$, with *LW* representing linear wear, and a, b, d and e coefficients that need to be found experimentally;
- The first derivative with respect to the linear wear is taken, resulting in the quadratic function: *c-factor'*= $3a\ LW^2 + 2b\ LW + d$. This quadratic function needs to fulfil two requirements. First, it needs to be a negative parabola, i.e., downward opening and second the top of the parabola should be located before the split of the two segments;
- If the quadratic function is found, and writing the empirical found relation between the slope (a_{ini}) of the initial part of the curve and at failure (a_{end}) as $a_{end} = p\ a_{ini} + q$, then by solving *c-factor'*=a_{end} for *LW*, gives the prediction for when failure will occur;
- Finally, by extrapolating the observed increase of linear damage with time, an estimate can be found in terms of calendar date when the *LW* obtained under step 3) presumably is reached. Similarly, the latter can be done to obtain two confidence limits for which the filament will fail with 95% confidence.

5.3.4 Artificial Neural Networks

ANNs are Machine Learning algorithms inspired by biological nervous systems such as the human brain. ANNs process information using a set of highly interconnected nodes, also referred as neurons, organized into layers. The structure of ANN's are typically split into three types of layers: one *input layer*; one or more *hidden layers;* and one *output layer* (see Figure 5.7). The input layer receives the data and is connected to the first hidden layer, which in turn is connected either to the next hidden layer (and so on) or to the output layer. The output layer returns the ANN's predictions.

A node links to other nodes by weighted connections. At each node, an activation function combines these weights into a single value, which may limit signal propagation to the next nodes. These weights, therefore, enforce or inhibit the activation of the network's nodes. Neural networks can

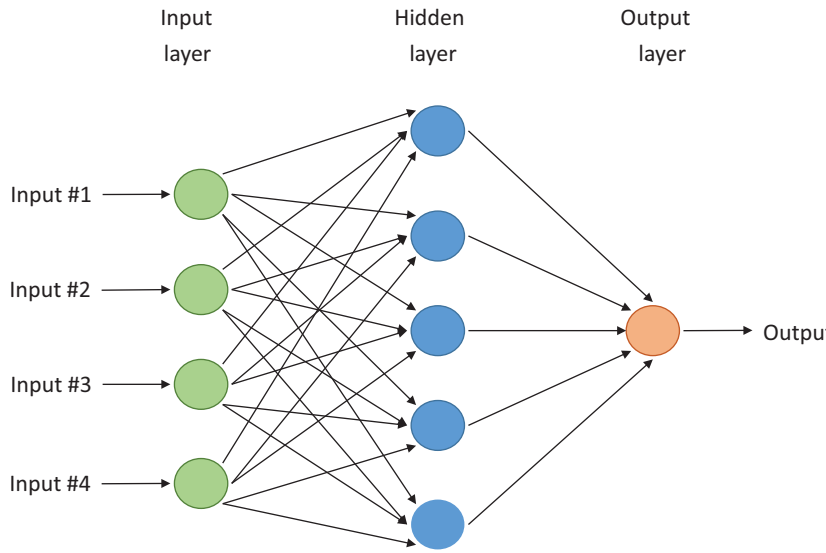


Figure 5.7 Schema of an artificial neural network, from [Haeusser, 2017].

detect complex nonlinear relationships between dependent and independent variables and require very little feature engineering.

5.3.4.1 Deep neural networks

DNNs (see Figure 5.8) brought major advances in solving some problems that were previously difficult to overcome in the artificial intelligence field [LeCun et al., 2015]. They have proved to be good at finding intricate structures in high dimensional data, which makes them relevant for many fields of study. Despite the fact that there is no clear border between what distinguishes a DNN from the others, a simple definition is that, a DNN contains many hidden layers in the network [Schmidhuber, 2015].

Common applications

- General classification [Baxt, 1990; Widrow et al., 1994];
- General regression problems [Refenes et al., 1994];
- Prediction of medical outcomes [Tu, 1996];
- Environmental problems [Maier and Dandy, 2000];
- Stock market index predictions [Moghaddam et al., 2016];
- Remaining useful life (RUL) [Ali et al., 2015];
- DNNs for image processing [Egmont-Petersen et al., 2002];

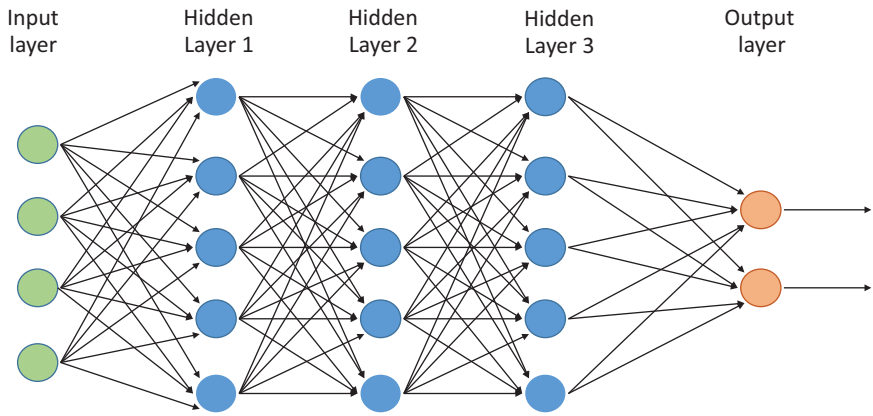


Figure 5.8 Deep Neural Network with 3 hidden layers, from [Nielsen, 2017].

- DNNs for audio [Lee et al., 2009];
- DNNs for text [Collobert and Weston, 2008];
- DNNs for regression tasks and remaining useful life [Tamilselvan and Wang, 2013].

Strengths and challenges

- Some of the strengths of ANNs are:
  - ANNs are effective at modelling complex non-linear systems;
  - Once an ANN has been trained, it processes the data efficiently;
  - ANNs (and DNNs in particular) require little to no feature engineering [Tran et al., 2011].
- Some of the challenges are:
  - Training a model usually incurs high computational and storage costs;
  - Lack of interpretability ("black boxes") despite efforts to extract knowledge from them [Andrews et al., 1995; van der Maaten and Hinton, 2008; Merrienboer et al., 2014];
  - Many hyper parameters need to be tuned;
  - Many configurations are possible;
  - DNNs typically require a lot of data to be trained [Chilimbi et al., 2014];
  - Collecting large labeled datasets required for training the DNNs is a challenge.

Research has been carried out in order to reduce the training time by using simple learning functions [Guo and Morris, 2017] or by transfer learning [Schmidhuber, 2015]. In regards to hyper-parameter tuning, several approaches have been proposed to automate this process; for example [Domhan et al., 2015].

Implementation

The use-case considered has a dataset that represents a time-series of sensor readings from a fuel system. Each sample provides information from several sensors and the current health state of the system (range from 0% to 100%). Exposing these sensors to extreme driving or ambient conditions for several years can lead to their failure or deviations in measurement. The goal is to predict the current health state of the fuel system as accurately as possible while being robust to sensor failure and deviations. Experiments based on small feed-forward neural network with only 3 hidden layers (Input layer: 161 neurons; Hidden layer 1: 128 neurons; Hidden layer 2: 256 neurons; Hidden layer 3: 128 neurons; Output layer: 6 neurons) and using drop-out and noise injection were able to maintain accurate predictions with as much as 40% of incorrect sensor data (using simulated errors).

5.3.5 Life Expectancy Models

In some cases, RUL computation leverages on models of life expectancy of assets. This is the case, in particular, when time series are used to express collected data over data, and to drive the prediction regarding particular parameters that can be useful to obtain a reliable RUL.

5.3.5.1 Time series analysis with attribute oriented induction

A time series is a collection of data measured over time and represents the evolution of a certain quantity over a certain period. The main feature of a time series is that the collected data values are successive in time and typically strongly correlated with neighbouring observations. In many scenarios the goal is to predict the future outcomes of a variable of interest. The overall research field dealing with time series is referred to as time series analysis. Commonly, in order to analyse a time series the signal is: 1) decomposed in various parts, that is in a *trend* component, a *cyclic* (seasonal) component (both nonstationary) as well as in the *random* (stationary) component that remains after removing the cyclic and trend effect from the observed series; 2) analysis is performed on the various parts, which are then 3) recombined

in order to make predictions. One of the most utilized time series forecasting models are autoregressive and moving average models (ARIMA). The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged, i.e., previous, values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The "I" (for "integrated") in the name ARIMA indicates that in many applications, instead of analysing the original data, the data is preliminary differentiated, one or multiple times. The purpose of each of these features is to construct an ARIMA model to fit the data as well as possible.

AOI is a data mining hierarchical clustering algorithm. The main characteristic of AOI is the capacity to describe the information in more general concepts/terms, reducing the dimension of the data. Its power resides on combining the monitored data with domain expert knowledge, in order to provide a more trusty knowledge generation.

The execution of the AOI algorithm follows an iterative process where each variable or attribute will have its own hierarchy-tree. Later this data must change from one generalization-level to another, generalizing all the data in the dataset. That step is denoted concept-tree ascension. AOI is an algorithm whose power resides on defining thresholds and generalization hierarchies for the attributes of the data. This means that it is an algorithm where domain expert feedback is necessary. Based on the defined hierarchy trees, generalizations are performed iteratively in order to conform similarity clusters. The higher the generalization level on the hierarchy tree is, the more general—but thereby also the more ambiguous—the cluster description gets. The more general the cluster, the lower the value of significance of the cluster. Such clusters refer to different states of the behaviour of the monitored asset.

Implementation

Time Series analysis with AOI was applied in relation to a clutch brake machine monitoring use case. Two main phases are employed on RUL calculation: (i) quantification, and (ii) time series analysis. The aim of quantification is to describe the signal(s) to be forecasted in a measurable (numeric) way. When quantification is applied over a set of signals, its information representation becomes simpler and easy to deal with, but also some information-loss occurs. With the help of AOI, different work-cycles (clutch or brake cycles) were labelled by a training process with a value corresponding to their level of normality, called Normality Factor, according to the ambiguity values of clusters. Thereby, a minimum value for a work

cycle to be considered as normal is calculated based on training phase data. Finally, the evolution of the Normality Factor value can be modelled with a time series based ARIMA model, in order to check when a work-cycle, which does not meet the normality conditions, occurs.

5.3.5.2 Application to a pump

A methodology for quantifying and monitoring the evolution of the performance of a pump has been formulated, in order to take maintenance decisions based on this information. The performance of a pump is influenced by two main causes: the wear/tear of its components and the clogging situations. In both cases, maintenance tasks may be needed to recover the best pump performance. A maintenance task is recommended when the performance decreases more than 20%. Therefore, in this context, RUL is understood as the time (in days) until a pump's performance decreases more than 20%. The methodology is based on data typically available for pumps: the operation frequency (Hz) and the pumped flow rate (m^3/h).

First of all, the performance of the pump is estimated each day using the available frequency and flow rate historical data. Later, the evolution of the pump performance is used for estimating the RUL. Thus, the trend of the pump performance is calculated applying a linear regression algorithm. Finally, this linear regression is used for predicting the evolution of the pump and calculating the RUL, i.e., the time until the performance reaches the value of –20%.

The RUL estimation error has been analysed for three clogging situations which vary in duration and magnitude. For each clogging period, Figure 5.9 shows the RUL estimation error on the Vertical Axis, as a function of the remaining days to clogging situation (Horizontal Axis). In particular, day 0 corresponds to the day of a -20% performance.

The RUL estimation error ended up being between ± 5 days during the analysed situations, which is considered good enough taking into account the duration of the clogging situations and the randomness of its nature.

5.3.5.3 Application to industrial forklifts

This use-case used data collected from industrial forklifts. The overall goal is to provide RUL estimates for forklift tires. Two different datasets were available to achieve this goal. The Fleet Manager system provides data on the usage by the on-board computers of different forklifts and generates 10-minute aggregated data reports. It contains fields like driven time, distance travelled, number of direction changes and consumed energy amount.

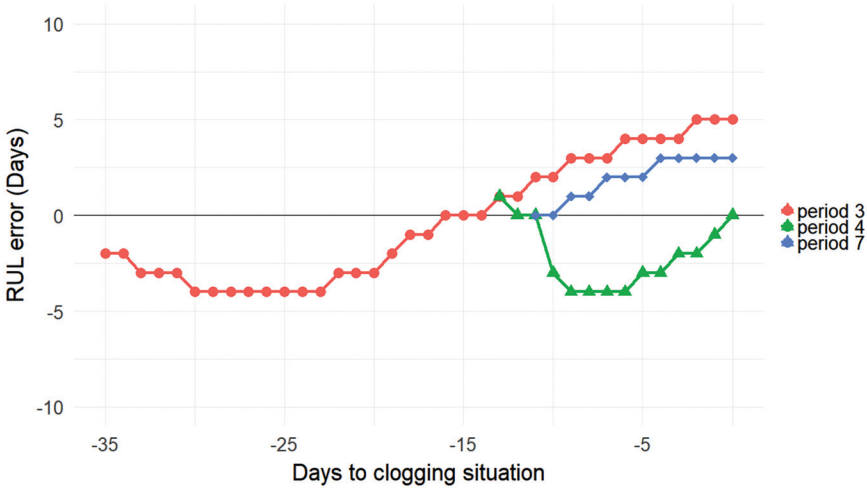


Figure 5.9 RUL error vs remaining days to clogging situation.

The second data source are the service reports from service technicians, in which they reported in free text both the problems identified in a forklift and the repairs that were performed.

Text analysis was used to identify and extract the dates from the service reports when a tire replacement occurred due to the tire being worn-out (and not due to any other cause). Thereafter, the Fleet Manager data set was further aggregated into time intervals where forklifts used a given set of tires. The regression model was built using this pre-processed dataset. The Fleet Manager data set was also clustered into three driving profiles by using expert knowledge, which resulted in useful categorical variables in the regression model.

The clustering was done in two steps. In the first step, a centroid-based k-means algorithm was used, and then the final 3 driving profile classifications were created with a hierarchical clustering method. This was needed to leverage the advantages of both algorithms, while minimalizing their drawbacks. For example, the k-means clustering algorithm tends to create equally sized convex clusters, which is not physically meaningful in relation to this data set. A more detailed analysis of these clustering solutions, highlighting the strength and weaknesses of both can be found in [Kaushik and Mathur, 2014].

The second step involved building an *ensemble tree-based* regression model, called *Gradient Boosted Decision Trees* [Friedman, 1999]. Like other ensemble regression models (e.g., Random Forest), this algorithm provides

good defence against overfitting at the cost of interpretation, in the sense that the predictions of the model are less transparent. The model handles different variable types well, and it is insensitive to correlated variables. With adequate parameter optimization, it is shown in [Caruana and Niculescul-Mizil, 2005] that the *Gradient Boosted Decision Trees* provides great accuracy compared to many other classical machine learning models. The model training included cross-validation to reduce the level of overfitting due to the limited amount of data points, and hyper parameter optimization to find the optimal algorithm parameters (e.g., number of trees or tree depth). The results were evaluated based on the Root Mean Square Error measure.

5.3.5.4 Application to a gearbox

One of the main objectives in this use-case in relation to RUL prediction is in improving the failure prediction of a gearbox by incorporating an additional covariate besides produced GWh to a failure model, and to test whether or not incorporating this additional covariate adds predictive power. The additional covariate is allowed to change with time and so it is a so-called time dependent variable.

The additional covariate is incorporated by applying proportional hazard modelling with a time-dependent covariate. The Incorporation of a time-dependent covariate within a proportional hazard model, requires the knowledge at each failure time *t* the value of the time dependent covariates of all assets that did not fail until time t. The latter demand is in practice in many situations hard to fulfil. Through the last two decades, applying time dependent covariates has become more and more common and standard software is available in for instance [R Core Team, 2014] to incorporate them [Therneau, 2015; Therneau and Atkinson, 2017; Fox and Weisberg, 2010].

Denoting w the produced energy [GWh], then for an object with one single GWh-dependent covariate the hazard function takes the form [Thomas and Reyes, 2014]

$$\lambda(w|z(w)) = \lambda_0(w)\exp(\beta z(w)), \tag{5.9}$$

with *z(w)* the value of the covariate at n of *w*, and w ≥ 0. By the hazard rate the survival function conditional under *z(w)* is

$$S(w|z(w)) = \exp[-\Lambda(w|z(w))] \tag{5.10}$$

with the cumulative hazard conditional under *z(w)* defined by

$$\Lambda(w|z(w)) = \int_0^w \lambda(u|z(u))\mathrm{d}u. \tag{5.11}$$

The latter two equations taken together imply that to define the survival function for an asset at w=0—or at the current value of w—it is necessary to know the future values of the covariate as a function of w, which is often an impossible requirement to fulfil [Fisher and Lin, 1999]. For more details concerning including time-dependent covariates in a proportional hazard model, please refer to [Fisher and Lin, 1999] and [Collett, 2003].

5.3.6 Expert Systems

An extended Expert system can be used as support for both RUL and RCA algorithms. The extension includes a Petri net [Aghasaryan, 1997] based execution scheduling, which allows for further enhancements of the system. The method entails introducing a supporting function in order to use an expert system along with a RUL estimation algorithm. In this use case the method is applied together with a Proportional hazard model based RUL estimation of tire wear (see Section 5.3.5.3).

The objective of the Expert System is to verify and make decisions based on the result of the RUL estimation algorithm. The RUL estimation is performed online, and updated every time a new sensor measurement set arrives. However, the estimated RUL needs to be validated, and then a decision is taken. The Expert System is performing the validation, and makes a decision accordingly. A rule-based system is selected as it scales well, provides results faster than case based systems [Simpson and Sheppard, 1998], and does not require large amounts of labelled training data like machine learning based methods. The proposed method mimics the behaviour of a human expert as follows:

- The Expert System is triggered by a rule;
- It extracts key parameters from the trigger description and based on this, a course of actions is selected;
- It initiates elementary investigation checks, search routines and possible correlated processes in a simultaneous manner;
- Once a result of a check is available, new routines are started using the new pieces of information;
- It continue to perform checks and tests until a result is found.

The course of actions to be executed for a specific trigger is based mainly on expert knowledge. For each monitored component, a different rule set is used, specific to the component.

Petri nets present an efficient way to implement data flow-driven programming. They can be used to mimic the simultaneous data processing

capabilities of a human expert. System specialists make ad-hoc plans or follow predefined procedures to find the root cause of an alarm. Consciously or not, they fetch further input data e.g., environment data, to start measurement or analysis processes. Simultaneous processes finish asynchronously, which forces the expert to make decisions on what to do next: what kind of further checks can be initiated using the gathered input data? In short, the expert's behaviour that consists in scheduling elementary checks to solve a RCA problem can be naturally modelled with Petri nets. In the use-case, the Petri net describes rules, extended with elementary checks. The elementary checks are typically basic checks like database lookups, more detailed data requests as well as they may include active measurements like on-demand vibration analysis. The various triggers associated with the definition of such Petri nets are based mainly on expert knowledge.

Implementation

In a complex system the input of RCA and RUL calculation is big data aggregated from multiple distinct sources, and the analysis of such big data is performed in the cloud. For implementation, Microsoft Azure has been chosen which provides extensive functionality for collecting and processing data from different sources. Azure IoT Hub is used to handle incoming chunks of data such as real-time measurements. The failure prediction is performed in Azure Stream Analytics based on the trigger conditions, which need to be uploaded to Azure Storage in advance. The IoT Hub receives the error logs and other measurements from the trucks and forwards this data to Stream Analytics, which checks for trigger conditions. A simple rule set can be implemented in Stream Analytics directly. The Stream Analytics process is defined in stream analytic query language, which has a similar syntax to SQL. When a failure is predicted, a Web based remote application will be informed, which does the actual verification. The remote application implements the Petri-net based scheduler and the elementary checks. The output of the application can be visualized in Azure Power BI, and it also integrates in the Mantis HMI (see Chapter 6).

5.4 Alerting and Prediction of Failures

Alerting is a method to trigger a corrective action. The purpose of the corrective action is to make sure that the failure does not re-occur. Data driven alerting is based on algorithms that detect and predict failures. These algorithms process historical data and, based on domain knowledge, produces a failure probability (see Figure 5.10). Based on business understanding, these probabilities are used to define the precise corrective action.

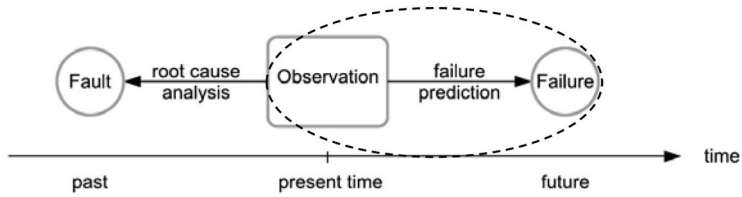


Figure 5.10 Overview for Alerting and prediction of failures.

5.4.1 Theoretical Background

There is a wide variety of algorithms available to detect and predict failures. The algorithm selection is based on business understanding, data understanding, and data preparation.

The CRISP-DM defines a methodology for structured data mining. For prediction and alerting of failures, the following phases are used: modelling, evaluation, deployment.

Modelling

Once the relevant data features have been identified, the algorithm to predict failures can be developed. The output of the algorithm is a statistical measure that indicates the failure probability. Based on business rules, an alert can be created if the probability warrants this.

Evaluation

The model is applied to a selected set of real-life cases. These cases are evaluated to judge whether there is a sufficient match between prediction and reality.

Deployment

The model is brought into production after successful evaluation. The model can be implemented in the system or outside the system (depending on the required data sources and computation capabilities) to create alerts in real-time.

5.4.2 Techniques Catalogue

There is a wide variety of modelling techniques available. Table 5.3 shows an excerpt of techniques.

Based on research on practical systems, a selected set of algorithms is described in the next sections. For each algorithm, the common application, strengths and challenges are outlined, and some practical examples are given.

Table 5.3 Techniques for Alerting and Prediction of Failures

Modelling technique	Algorithm
Pre-processing	Nearest Neighbor Cold-deck Imputation
Classification / Pattern Recognition	SVM, LDA, Pattern Mining
Dimensionality reduction	Temporal Pattern Mining, PCA
Probabilistic graphical models	Hidden Semi-Markov model with Bayes Classification
Neural networks	DNN, Autoencoders, Convolutional neural network with Gramian Angular Fields, RNN with LSTM
Time Series Analysis	Change detection
Statistical tests	Fisher's exact test, Bonferroni correction, Hypothesis testing using univariate parametric statistics, Hypothesis testing using univariate non-parametric statistics, Mean, thresholds, normality tests

5.4.2.1 Nearest neighbour cold-deck imputation

General description

Many machine learning algorithms assume that all records in the data set are complete, i.e., there is no missing data. Industrial data, however, often contains gaps e.g., due to misconfigured sensors, connectivity problems or manual input. One approach for addressing the missing data problem is to impute the missing values. Let us assume that the data is provided as a table, with columns representing the features and rows representing the records. The common hot-deck methods include mean imputation (the missing values in a column are filled in with the mean of that column), fixed value imputation, random imputation (the missing values are filled by randomly sampling the given values in the column), considering donor samples from the *same* data set. Nearest neighbor cold-deck imputation fills in the missing values from donor records that are selected from a *different* data set. It selects the donors by searching for the nearest neighbours using meta-features describing the original data set or its column. The meta-features can be derived from the available data samples (e.g., mean, range, ...) or external to the data (e.g., specification of the data source, conditions under which the data was collected . . .).

Common applications

Cold deck imputation in general is commonly used in for imputing missing data in structured collections of data such as tables and questionnaires.

Strengths and challenges

Nearest Neighbour Cold-deck Imputation performs better than the traditional hot deck imputation methods for data sets for which a significant proportion of the data is missing. The performance of Nearest Neighbour Cold-deck Imputation also depends on the meta-features used for selecting the nearest neighbour donors, and how they partition the donor space. If donors are selected based on meta-features derived from the data set, then (similarly to hot deck imputation) the more data is missing the less accurate the imputation. If external meta features are used, then the imputation accuracy does not depend on the amount of missing data, but on the relationship between the external features and the underlying data generating process.

5.4.2.2 Support vector machine

General description

A SVM is a supervised learning model that can be used for binary classification and regression analysis. Supervised learning requires labelled training data consisting of input objects (e.g., a feature vector) and a desired output in order to infer the model parameters from the data and to be able to predict new incoming data. A SVM constructs a hyperplane or a set of hyperplanes to separate classes in the feature space. SVM defines optimal hyperplanes for linearly separable patterns and can be extended to patterns that are not linearly separable, by transforming the original data to a new space by using a kernel function.

A hyperplane is defined as $f(x) = b + w^T x$ where w is the weight vector, b is the bias and x the training dataset. Infinite hyperplanes can be used to separate the classes by scaling w and b (See Figure 5.11). Rewriting the hyperplane in canonical form states $\left|b + w^T x\right| = 1$.

The optimal hyperplane is therefore defined as the plane maximizing the margin of the training data. In SVM the data points that lie the closest to the decision surface and therefore the most difficult to classify, are the ones influencing optimality. These points are called support vectors.

The distance between points is defined as $d_i = \frac{\left|b+w^T x_i\right|}{\|w\|} = \frac{1}{\|w\|}$

In a binary classifier, the margin M is defined as twice the distance to the closest examples $M = \frac{2}{\|w\|}$

Therefore, the problem of maximizing the margin M is the same as the problem of minimizing the following function L subject to constraints:

$$\min_{w,b} L(w) = \frac{1}{2}\|w\|^2 \ \ subject\ to\ y_i(\ b + w^T x_i\ ) \geq 1\ \forall i \qquad (5.12)$$

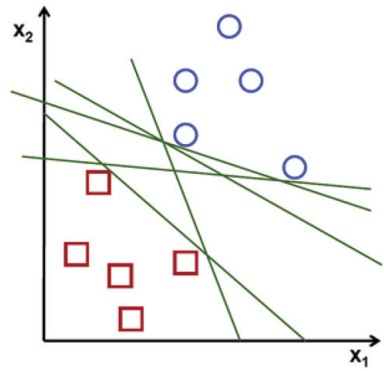


Figure 5.11 Linearly separable point can be separated in an infinite number of ways.

Where y_i represents each of the labels in the training dataset. This is a constrained optimization problem that is solved by Lagrangian multiplier method as:

$$\mathrm{L} = \frac{1}{2}\|w\|^2 - \sum \alpha_i [y_i \left(b + w^T x_i\right) - 1] \tag{5.13}$$

By equating the derivatives to 0, the result is $w = {}^P\alpha_i y_i x_i$ and ${}^P\alpha_i y_i = 0$ (Figure 5.12).

In case the data are not separable by a hyperplane, SVM can use a soft margin, meaning a hyperplane that separates many but not all data points (Figure 5.13). Different formulations are possible for adding soft margin by adding slack variables s_i and a penalty parameter C. An example is the L1-norm problem:

$$\min_{w,b,s} L\,(w) = \frac{1}{2}\|w\|^2 + C\sum_j s_i \;\; subject\ to\ y_i(\;\; b + w^T x_i\;) \geq 1\ \forall i\ and\ s_i\;\; \geq 0 \tag{5.14}$$

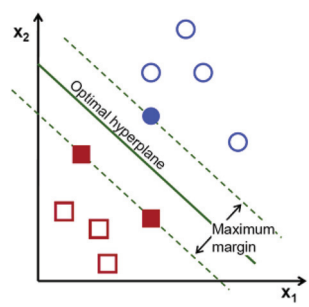


Figure 5.12 Optimal hyperplane in a 2D feature vector obtained by maximizing the margin.

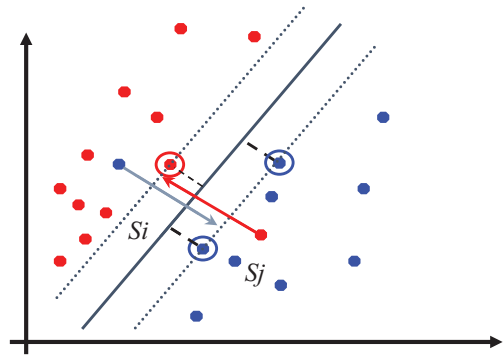


Figure 5.13 Slack variables introduced for not linearly separable problem.

Some problems that do not have a simple hyperplane as a separator may require nonlinear transformation with kernels. The idea is to gain linearly separation by mapping the data to a higher dimensional space (Figure 5.14).

SVM are inherently a binary classifier that can be extended to the multi-class problem (e.g., by reducing the multiclass problem into multiple binary classification problem).

In the context of anomaly detection, one-class SVM can be used to detect if the system is behaving normally or not. In order to define the normal behaviour of the system to be monitored, historical data of the system running in normal behaviour is required. The support vector model is hence trained on the anomaly-free data and afterwards each new measurement point is scored by a normalized distance to the decision boundary.

Common applications

SVM was developed by Vapnik and Chervonenkis [1971]; Vapnik [1995] and became popular because of its success in handwritten digit recognition.

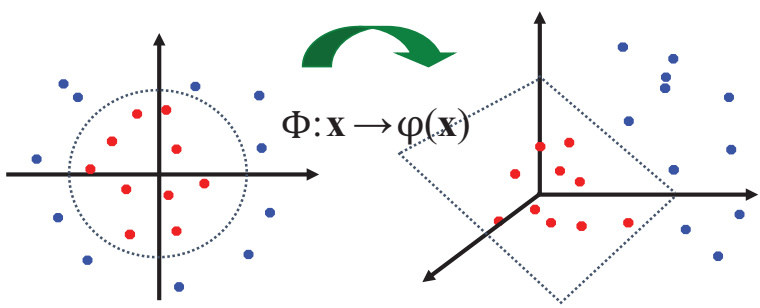


Figure 5.14 SVM kernel transformation to gain linearly separation.

SVMs have demonstrated highly competitive performance in numerous real-world applications such as bioinformatics, text mining, face recognition, and image processing, which has established SVMs as one of the state-of the-art tools for machine learning and data mining.

Strengths and challenges

- Some strengths of SVM are:
  - Training time necessary depends on the number of supports and the kernel chosen, while the execution time is very short so that it is suitable for real-time application;
  - SVM do not suffer from the curse of dimensionality;
  - SVM is not much affected by local minima in the data;
  - Does not require lot of data to be trained, however the data should come from all the possible normal behaviours of the system.
- On the other hand, it also has some limitations, some of them being listed below:
  - The feature space of a SVM is often high dimensional and therefore has many parameters and is hard to estimate;
  - Selection of the kernel function is not trivial.

5.4.2.3 Linear discriminant analysis

General description

Let P(x) be the probability distribution of the input data, and P(y) the class distribution. The goal of a classifier is to assign a sample x to the class y, which corresponds to the largest posterior probability P(y|x) that input x belongs to class y. The LDA classifier relies on the Bayesian theorem

$$P\left(y \mid x\right) = \frac{P\left(x \mid y\right) P(y)}{\int_y P\left(x \mid y\right) P\left(y\right) dy} \tag{5.15}$$

to compute the posterior using the likelihood P(x|y) of input x being generated by class Y, the prior P(y) and the marginal probability P(x) of input x. The P(y) can be easily estimated from the data by taking the fraction of samples that belong to class Y. Assuming that likelihood P(x|y) is normally distributed, it can be estimated by fitting the mean and variance parameters to the data. See James et al. [2014] for more details, including the formula for the discriminant function used to select the most likely class for input x.

Common applications

LDA is a general classifier that can be applied in various applications. In the context of failure predictions, it can be applied to classify process measurements as those likely to result in a failure.

Strengths and challenges

- LDA is applicable in cases when there are few samples, or the classes are overlapping or well separated (in contrast to other classification methods, e.g., logistic regression, which become unstable when classes are well separated). LDA performs best when the classes (i.e., P(x—y)) are approximately normally distributed.

5.4.2.4 Pattern mining

General description

Pattern mining is one of the most known methods to extract useful information from the data. By looking for patterns, i.e., closely linked events or items, this technique allows to analyse behaviour and make prediction on future behaviour.

By knowing this behaviour, pattern mining can predict when a failure will occur (by detecting also when the start of the sequence occurs) and can signal the problem before it occurs. This method can be deployed in many various fields, such as maintenance of a factory or a wind mill farm, determining a disease, predicting the next medical prescription, analysing the behaviour of cyclists in a town.

The first pattern mining algorithm, Apriori, was created in 1995 and was only able to find frequent item sets, i.e., to find items that usually occur together. Now pattern-mining methods can also look for sequences of items where the order of the items matters.

Common applications

The list below only mentions the main topics, but many others exists:

- Frequent item sets
  A supermarket can analyse the list of goods bought by their customers to find those usually bought together. It can help the supermarket managers to reorganize their shelves or to propose special offers;
- Sequence of events
  A logistic company that continuously monitors events occurring in their vehicles, e.g., engine oil level too low or motor temperature too hot.

As they also know when the vehicles are break, they can find patterns leading to failures, i.e., the sequence of events that usually precede a failure;

- Multi-level
  In multi-level pattern mining, not only the items are considered, but also their hierarchy. Going to the supermarket use case, multi-level pattern mining can output purchases containing apples, pears, kiwis. All of them are considered as distinct items although they are all fruits. Therefore, the supermarket may also be interested to find patterns containing fruits and not the more specific sub-level items, e.g., apples. In addition, by considering the fruits as a whole, it is possible to find patterns that would be hidden if selected independently.
- Multi-domain
  The multi-domain pattern mining methods consider multiple properties of the items when looking for patterns. The supermarket could have stores all over a country and would be interested to check if there are different patterns depending on the location of the purchase. These methods could also analyse the patterns depending on gender or income of the customer. Multi-domain methods will answer questions such as which goods are usually bought together by rich women in Brussels.
- Temporal pattern mining
  When considering a sequence, the interest property can be not only the next item of the sequence, but also the usual gap time before the occurrence of that item. This is particularly interesting for maintenance purposes. If a pattern leads to a failure or defective product, it can be interesting to know how many times it is necessary to prevent that problem and ensure that there is enough time to do so. Indeed, nobody is interested in a pattern than only allows to predict a car's failure 3 seconds before it happens.
- Constrained pattern mining
  This topic covers many different methods that allow to impose constraints to the patterns. A constraint can be defined on the length of the patterns, to only retrieve patterns of more than 5 items. For example, directed pattern can search for patterns containing items of interest, e.g., to find patterns containing fine Belgian chocolate. Aggregate constrained pattern mining imposes a constraint on an aggregate of items, e.g., to find patterns where the average price of all items is above 50 euro.

Strengths and challenges

- Strengths
  - Insights about the sequences of events;
- Challenges
  - Computation time on large datasets.

5.4.2.5 Temporal pattern mining

General description

Temporal abstraction is the process of transforming a point time series into a series of state intervals and is widely used in data mining to aggregate multivariate time series into a representation that is easier to analyse. This higher-level representation is equivalent to a smoothing of the data. Instead of a series of points, the values are discretized using an abstraction alphabet Σ that represents the set of possible value ranges a variable can assume (e.g., Σ = {High, Medium, Low}). Each time series is then represented by a series of intervals during which the value is constant. Temporal Pattern Mining then searches for (multivariate) temporal patterns that are common among the input samples, where a pattern encodes the temporal ordering between the intervals present in the sample (e.g., "Sensor1.High before Sensor2.Low and Sensor2.Low concurrent with Sensor3.Medium"). Each sample can then be reduced to a binary vector indicating which temporal patterns it contains. The input dimensionality can be further reduced by keeping track only of those patterns, which are sufficiently long and unique. See Batal et al. [2013] for more details.

Common applications

Temporal pattern mining is applied for mining symbolic patterns in various domains, including natural language processing, bioinformatics (e.g., finding discriminative patterns, sequence alignment).

Strengths and challenges

Temporal Pattern Mining can be used as a dimensionality reduction technique for multi-dimensional time series, as a pre-processing step for classifying noisy multivariate time series with irregular time intervals between measurements, different granularity across time series, and missing values.

5.4.2.6 Principal component analysis

General description

PCA is a space transformation method used to reduce the dimensional space from a multivariate dataset while retaining most of the information (components). This statistic technique converts a set of observations of possible correlated variables into a set of values of linearly uncorrelated variable called principle components by applying a transformation in such a way that the first principal component has the largest possible variance [Pearson, 1901]. The key idea is that if two signals are highly correlated or dependent, one is enough to carry the information.

Two steps are performed in the PCA transformation [Jolliffe, 2002]:

- Find principal components (Figure 5.15);
  - Compute the covariance matrix Σ of the dataset X;
  - Find P = (n,n) the n eigenvectors $\overrightarrow{p_1}, \overrightarrow{p_2}, \ldots, \overrightarrow{p_n}$ of Σ and the corresponding eigenvalues $\Lambda_1 = \Lambda_2 = \ = \Lambda_n$ where P represents an orthonormal basis.
- PCA approximation by ignoring components with lower significance.

Important properties of the PCA transformation are 1) that the principal components are uncorrelated since they are orthogonal to each other; 2) the data can be transformed between the bases without loss of information; 3) the principal components are ordered so that the first retain most of the variation in the data.

The PCA can be used for different applications, one of which is fault detection. This requires having historical data of the system to be monitored

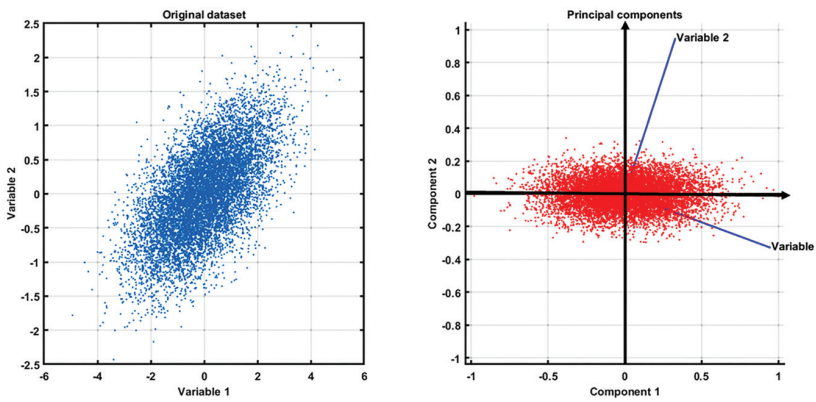


Figure 5.15 Principal components of a 2-dimensional data.

in a healthy state *Xh*. For each new input data *Xobs*, the projection on the eigenvectors is calculated, and then anti-transformed in order to calculate the normalized reconstruction error, which is a measurement for determining whether the input belong to normal or to abnormal conditions, as follows:

- Normalize the data so that the mean is zero and standard deviation is one;
- Calculate the principal components *P* of *Xh* and retain only the first *n* components containing a predefined variance (e.g., 95%);
- Reconstruct the input data Xobs by projecting in the transformed basis, keep only the principal components and anti-transform: $X_{rec} = X_{obs}PP^T$;
- If $X_{rec} = X_{obs}$ then the input data belong to the normal condition, otherwise it belongs to the abnormal condition. This distance need to be defined based on threshold values.

Common applications

PCA is often used as a technique to reduce the amount of information from an original dataset before a classification or a clustering step is applied. Moreover, it can be used for finding patterns in the data and for data compression.

Strengths and challenges

- Some strengths of PCA are:
  - Training time necessary to find the principal components is proportional to the number of measured signals, usually denoted as n, while the execution time is very short so that it is suitable for real-time applications;
  - Reduce the dimensionality of the problem, retaining only the relevant information;
  - Sound statistical technique;
  - Does not require lots of data to be trained, however the data should come from all the possible normal behaviours of the systems.
- On the other hand, it also has some limitations, the most prominent of them are listed below:
  - PCA has low robustness on highly correlated measurements;
  - PCA produce unsatisfactory results for dataset characterized by highly nonlinear relationships.

5.4.2.7 Hidden Semi-Markov model with Bayes classification

General description

HMM is a form of Markov Mixture Model, which contains a hidden and observable stochastic process. The former is a hidden process representing the system states while the latter is a Markov chain of the output of the system known as observations. A complex system (of which we cannot model its internal operation) can be represented by a HMM. A model can be trained with observation sequences which can (using a proper structure and sufficient number of training sequences) estimate the likelihood distribution of states at each transition, e.g., the most likely state of the system.

The Hidden Semi-Markov Model is an extension of HMM with the duration parameter, e.g., we can use sequences where observations do not follow in constant time.

Common applications

HMMs have been used in speech recognition for many years, using it for failure prediction is quite new. It is a supervised learning approach, so training data is needed. Multiple models need to be trained, at least one for failure sequences and one for non-failure sequences. Based on observations, sequences are assembled and sequence likelihood is calculated on each model. A Bayes classification algorithm is used to detect a failure-prone case and the probability of failure.

Strengths and challenges

For the successful outcome of using the HMM approach, a proper model structure needs to be selected. The number of states needs to be selected based on the length of the observation sequences. In addition, this length needs to be the same for each sequence. Shorter sequences need to be padded with synthetic symbols.

5.4.2.8 Autoencoders

General description

An autoencoder is a neural network specifically designed for learning representation (encodings) from data. It can be used in an unsupervised setting such as dimensionality reduction. It mays also be used for creating generative models, which were exploited in the MANTIS project.

An autoencoder is a standard feedforward multilayer neural network, however, its topology (structure) is characterized as follows:

- The number of output nodes is the same as the inputs in order to allow for signal reconstruction;
- A central layer with the minimum number of nodes is used to encode the data (minimal representation), denoted as the **z** layer;
- The connections from the input nodes through 0 or more hidden layers to the z layer are used to encode the signal;
- The connections from the output nodes through 0 or more hidden layers to the z layer are used to decode the signal;
- The network on left (input) and right (output) are usually symmetric.

The simplest autoencoder network consists of one input layer, a single hidden z layer and one output layer. Autoencoders are trained to minimize the reconstruction error by measuring the difference between the input and the output. One such loss function can be for example the squared error.

It is important to note that the z layer must have a reduced number of nodes, otherwise it will not be able to learn an encoding and simply "memorize" all of the examples. In effect, the z layer should be small enough to *compress* the data into a reduced set of useful features, although not too small, otherwise the learned features will not be useful. Several variations of these networks have been developed with the aim of finding such useful features, i.e., capturing essential information and find a richer set of features.

One variant is the **denoising encoder** that assumes that the input data consists of a stable set of higher level features that are robust to noise. The aim is to extract those high-level features that represent the data distribution robustly. In this variant, the learning procedure first corrupts the signal and uses that as input to the network. Learning follows the usual procedure of loss evaluation and backpropagation of the error. However, it is important to note that the loss function is between the original uncorrupted signal and the network's output signal.

The **sparse autoencoder** works under the assumption that by making the hidden layers sparse, it is possible to better learn the structures that are present within the data. Sparsity is imposed by either manipulating the loss function or deactivating some of the hidden nodes with higher activation values (for example using a L_1 regularizer). Intuitively, the network will learn to select a subset of features for a given number of inputs that are characterized by those features only.

The **variational autoencoder** has its roots in Bayes inference and graphical models. It considers the hidden nodes as a set of latent variables connected in a regular graph (fully connected layers) and whose parameters (weights)

are unknown. Variational Bayesian methods are then used to provide an approximation to the posterior probability (weights) of the unobserved variables (hidden nodes), and base their mechanisms on an iterative procedure similar to expectation maximization to determine the solution. More concretely, they use a loss function with additional penalty terms (function includes a term using Kullback–Leibler divergence) and trains the network with the Stochastic Gradient Variational Bayes algorithm. Note the posterior probability that is learned can be seen as two conditional probabilities each with its own set of parameters, one for encoding and another for decoding. Moreover, these are encoded separately in the loss function.

The **contractive autoencoder** [Rifai et al., 2011] uses a loss function that promotes learning models that are robust to variations of the input. The loss function uses a Frobenius norm of a Jacobian matrix as a regularizer. The partial derivatives of this norm apply only to the activations of the input nodes. This penalty has the effect of ensuring that only significant changes of the input cause changes in the hidden layers. In other words, if the partial derivatives are zero, then feature learning is reduced. The intuition here is that only important changes will promote significant changes in the learned parameters.

Common applications

The autoencoders ability to encode data and regenerate the input can be taken advantage of in the following way:

- Dimensionality reduction: use the z hidden layer as an output for visualization;
- Feature engineering: use the z hidden layer as an output to generate input for other machine learning algorithms (regression, classification);
- Clustering: use the distance between the z hidden layer's outputs measure similarity and difference between input signals (feature engineering for exploratory analysis). The same technique can be used for classification;
- *Pre-training* of Deep Network: some deep neural networks are difficult to train due to the vanishing and exploding gradients. The deep networks can use autoencoders' learning process in an initial stage in order to initialize the parameters (weights) that will facilitate learning in the next stage (the use of ReLU as activation functions has made this largely unnecessary);
- One-Class Classification: the error of the regenerated signal can be used to determine if the input signal has any new features that differentiate it from the training dataset;

- Image denoising: robust autoencoders can regenerate noisy input signals by extracting and using only the most salient features that are resistant to noise;
- Natural language processing (NLP): autoencoders have been used to solve problems in word embedding (converting words or sentences to vectors), build bilingual word and phrase representations, document clustering, sentiment analysis and paraphrase detection.

Strengths and challenges

The main difficulty in using the autoencoder was the set up of the hyper-parameters and the topology. A simple grid search was performed. We found that a simple 3 layers network was effective (the input and output layers consisted of 32 nodes). We could successfully detect changes in the normal distribution's standard deviation (z layer with 16 nodes) and the mean and/or standard deviation (z layer with 25 nodes).

Advantages

- No need to understand the data;
- Automated feature engineering (data transformation);
- Reduced pre-processing required (except for normalization or scaling).

Disadvantages

- Requires large datasets of relevant high quality data;
- Requires large amounts of CPU time and memory;
- Many hyper-parameters to tune;
- Difficult to select the most appropriate loss function and network topology;
- Model validation is difficult (requires correct data labelling);
- Training is difficult if the data is not representative of the problem (for example, non-failure data contaminated with failures);
- May learn irrelevant features (only detectable during model validation). One further issue regards how to bias the network to learn those important features;
- Features are not interpretable.

Use in MANTIS

The technique described above can be used to detect anomalies that may occur in a press brake machine. Signals indicating the movement (velocity and distance) of the press ram and back gauge, oil temperature and vibration can be used to detect failure.

Denoising encoders allows us to encode and regenerate the signals that represent the machine with no failure. To test for failure, the signals is encoded, and it is verified that its reconstruction is within a given bound. If it is not, it can be concluded that the signal signature has not been encountered before (because it cannot be decoded successfully) and it most probably represents a failure. Unlike the statistical test, we are not testing if a given sample belongs to the same population. We are in effect testing if the "curve" that describes the displacement of the ram or back-gauges have the expected "shape". Samples of non-failures "curves" are the dominant class that were used during an initial training phase. Samples of failures "curves" can then be used to establish the cut-off threshold between the fail and no fail classes during a second phase of learning. This threshold is then used again for model evaluation.

Due to a lack of data, synthetic data were used for the tests, generated using a normal distribution by varying either the mean, standard deviation or both. The autoencoder was trained for datasets with a single set of known statistics (no failure). The autoencoder was then used to reconstruct the signal for data with a different set of statistics (with and without failure labels). During the learning phase, we determined the minimum square error threshold that was necessary to reduce false positives. During a second learning phase, signals with different statistics were used to determine the minimum square error threshold that was necessary to reduce false negatives. The final threshold was then set to the mean of these two separate thresholds.

Test sets were then used to regenerate the input signal and calculate the reconstruction error (containing both failure and non-failure signals). If this error was below the threshold no failure was assumed, otherwise the data was labeled as a failure. The *Mathews correlation coefficient* was then used to evaluate the performance of the model. The *Mathews correlation coefficient* emerged from the experiment was evaluated as 1 (perfect score).

5.4.2.9 Convolutional neural network with Gramian angular fields

General description

According to a survey from 2014 [Schmidhuber, 2015], one of the currently most successful deep learning methods uses RNNs with LSTM or CNNs with max-pooling. This technique description will focus on the latter one.

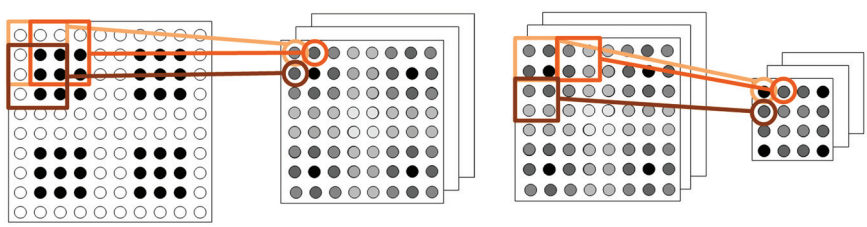
In image classification, the goal is to recognize (classify) what is shown in a given input image. Typically, an image is encoded as a matrix with values between 0 and 1 or 0 and 255, along with the correct label (i.e., what is

shown in the image). Colour images are typically encoded as 3-channel (red, green, blue) images. Several benchmark datasets are available in this domain, including MNIST[1] (grayscale images of handwritten digits), ImageNet [Deng et al., 2009] (colour images of objects, animals and scenes) and CIFAR[2] (colour images of vehicles and animals), to name a few.

In essence, CNNs rely on three basic concepts, which are described below and illustrated graphically in Figure 5.16. To simplify discussions, we assume that the input consists of 1-channel images. Note that in a CNN, the input image is treated as an h $\times$ h square, with h the number of pixels (or: the height and width of the image). Hence, the input layer of a CNN contains h $\times$ h input neurons, where each neuron contains the intensity of one pixel of the image.

Each neuron in the first layer after the input layer, known as the convolutional layer, is only allowed to connect to a small part of the input layer. In particular, each neuron in the latter layer connects to a small square of the input neurons (e.g., 3 $\times$ 3), known as the *local receptive field*. In this work, the size of this square region is denoted as *recepsize*. Each hidden neuron will learn a weight over the connections from its local receptive field, along with a bias. Intuitively, each neuron in the convolution layer thus learns to analyse the data in its local receptive field.

In Figure 5.16a, a 10 $\times$ 10 pixel input image is shown (i.e., h = 10). The neurons of the next convolution layer use local receptive fields of *recepsize* = 3 to connect to a small square region of the input. The first convolution neuron (top-left neuron first row, light orange circle) connects to the top left receptive field (light orange rectangle) of the input. For the next neuron,



(a) Convolution (b) Pooling

Figure 5.16 Detection of the presence or absence of a 3 $\times$ 3 square in the input image by a Convolutional Neural Network.

[1] Y. LeCun, C. Cortes, and C. J.C. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/

[2] A. Krizhevsky. CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/kriz/cifar.html

the receptive field is shifted by 1 pixel horizontally, obtaining the receptive field for the second convolution neuron (dark orange). This procedure repeats itself, up to the borders of the input image. For the first neuron on the second row of the convolution layer (brown circle), the local receptive field of the first neuron in the first row is shifted by 1 pixel vertically (brown rectangle). The amount of pixels shifted (either horizontally or vertically) to obtain local receptive fields for convolution neurons is known as the stride length, denoted *stridelen*. Subsequent horizontal or vertical local receptive fields therefore always overlap by *recepsize* x *stridelen* pixels from the input.

The number of pixels that remain of an h pixel image after applying convolution with receptive fields of size *recepsize* and stride length *stridelen* can be computed according to the following formula:

$$conv_{size} = \frac{h - recepsize}{stridelen} + 1. \tag{5.16}$$

In Figure 5.16a, h=10, *recepsize=3* and *stridelen=1* resulting in a $\frac{10-3}{1} + 1 = 8$ pixel image. In order to match this reduced image size, there are 8×8 convolution neurons in the subsequent layer.

Shared weights and biases

In a CNN, all the weights and biases of the neurons in the convolutional layer are shared. This allows the network to detect a particular feature in the input image at any position; this property is also called *translation invariance*. Note that the term feature in the context of CNNs has a different meaning than the feature from feature selection methods. In the former context, a feature typically represents a certain part of an image (such as a horizontal line). In the latter context, it refers to a piece of information in a dataset (e.g., a column or a combination of columns) that can be used for classification, to distinguish failures from non-failures.

The weights and biases from the input to the convolutional layer are often referred to as a feature map; a CNN usually has multiple feature maps to learn different features in the input image (e.g., horizontal, vertical and diagonal lines). In this work, the number of feature maps is denoted as *nrmaps*. In Figure 5.16a, three feature maps (*nrmaps* = 3) are used to analyse the 10×10 input image, where each of the neurons in the other feature maps has the same connectivity pattern of the top feature map shown here.

Pooling

The objective of pooling is to simplify or summarize the information from a convolutional layer in a small square region (e.g., 2×2), for a particular

feature map. The size of the pooling region is also known as the pooling size, denoted *poolsize*. Different types of pooling exist, but the most common form is max-pooling, where a pooling unit outputs the maximum activation in its pooling region. Pooling is a nice way to reduce the number of parameters of a CNN, because it reduces the size of the remaining input image (after applying convolution). In particular, if the size of an image after applying convolution is *convsize*, then this is reduced to a $\frac{convsize}{poolsize}$ image after max-pooling.

Max-pooling is illustrated in Figure 5.16b. Each neuron in the pooling layer uses the convolution layer from Figure 5.16a as input. The first neuron in the pooling layer (light orange circle) connects to a small square pooling region of size 2 (light orange square). For the second neuron (dark orange circle), this square is shifted horizontally by the pooling size, obtaining the pooling region for this neuron (dark orange square). This procedure is repeated up to the borders of the input. For the first pooling neuron on the second row (dark brown circle), the square from the first pooling neuron of the first row (light orange) is shifted vertically by the pooling size to obtain a new pooling region (dark brown square). As in convolution, pooling is done per feature map separately and so there are again 3 feature maps in Figure 5.16b that follow the same connectivity pattern. Unlike convolution however, there is no overlap in subsequent horizontal or vertical (pooling) regions. According to the calculations from above, *convsize* = 8 in Figure 5.16a after convolution. With *poolsize* = 2, this image is further reduced to $\frac{8}{2} = 4$ after applying pooling; there are therefore 4×4 pooling neurons in Figure 5.16b.

Because of these three basic concepts, CNNs are powerful networks that can learn useful spatial representations of images and at the same time require relatively few parameters when compared to other types of networks. Generally speaking, if a neural network has few parameters, it also has a shorter training time. How well a neural network performs is not only determined by its test performance, but also by the time needed to train the network. After all, a network that performs very well (e.g., achieves over 90% accuracy) that takes weeks (or even months) to train is not very useful.

Common applications

CNNs are typically used to perform classification tasks on images, for instance on the ImageNet dataset [Deng et al., 2009]. Images are not time series however and in some approaches [Wang and Oates, 2015; Bashivan et al., 2015], a transformation technique is employed to transform the raw time series data into a sequence of images, before providing it as data to some kind of CNN. Other approaches make use of EEG, video or log data

to do time series classification with CNNs, for instance in the context of making predictions in the medical domain [Hajinoroozi et al., 2016], human action recognition [Ijjina and Chalavadi, 2016], automatic annotation of clinical texts [Hasan et al., 2016] and automatic identification of predatory conversations in chat logs [Ebrahimi et al., 2016].

Strengths and challenges

- Some strengths of CNN are:
  - CNN is one of the strongest algorithms with a high accuracy for image recognition;
  - CNN can automatically extract high-level features from the raw data that are suitable for solving classification or regression problems, without relying on features designed manually by an expert;
  - CNN are robust for distortions in the image caused by the camera lens; different lighting conditions, horizontal and vertical shift, etc. [Hijazi et al., 2015];
  - Convolving filters over the input vector allow to reduce the number of trained parameters, thus reducing the resource requirements for training and storing CNN models.
- On the other hand, the usage of CNN gives rise to some challenges:
  - The algorithm requires huge amounts of data. Most problems using neural nets require somewhere around 10.000 examples, but preferably more. To collect and label these examples can be very time consuming;
  - CNN and artificial neural networks in general are very computationally expensive to train. Due to the advancements in GPUs in the last decade, it is now viable to train neural networks;
  - While visualizing the feature maps can provide some insight into the features that a CNN extracts in the intermediate layers, the inclusion of dense layers makes it difficult to interpret how the neural network model produces its prediction and to trust the result. For some domains, e.g., medical and fraud investigation, this is unacceptable;
  - The design of CNN architectures for solving a particular machine-learning task remains an art.

5.4.2.10 Recurrent neural network with long-short-term memory

General description

According to a survey from 2014 [Schmidhuber, 2015], some of the most successful deep learning methods nowadays use Recurrent Neural Networks RNNs with LSTM or CNNs with max-pooling. This technique description focuses on RNN with LSTM.

In a RNN [Zachary et al., 2015], both forward and recurrent edges to neurons are allowed. Recurrent edges may form cycles in a network, including a cycle of length 1 (known as a self-connection or self-loop). Because of the recurrent connections, a certain neuron now not only receives input from the current data at time t, but also from the previous hidden state at time t - 1. The recurrent edges thus introduce a notion of time in RNNs and as such, they are naturally used to model sequences (including time series). An example of a time series classification benchmark dataset is the University of California, Riverside (UCR) repository [Chen et al., 2015]. Although RNNs are very powerful networks, they often suffer from either the vanishing gradients or exploding gradients problem. In the former problem, the gradients (used to update the weights in the various layers) grow exponentially smaller as they are propagated backward through the hidden layers. In the latter problem, the opposite occurs, and the gradients grow exponentially larger. Both problems cause learning to considerably slowdown in RNNs. To overcome this problem, a special type of neuron or memory cell has been introduced, known as a LSTM cell [Gers et al., 2000]. This memory cell provides a form of intermediate storage in between the long-term memory stored in the weights of an RNN (which tend to change slowly over time during training) and the short-term memory stored in the activations passed from one neuron to another (which can change much faster). A simplified overview of an LSTM is shown in Figure 5.17a. The idea of the LSTM cell is to allow a network to hold a value for a longer period of time, using a special internal state that has a self-loop with weight 1. Both the flow in and out of the cell are controlled via gates, as illustrated in Figure 5.17a (gates are shown in gray). If the input gate is set to high (Figure 5.17b), data from the rest of the network (e.g., a value) can flow into the memory cell and change the internal state. If the input gate is set to low (Figure 5.17c), no data can flow into the cell. A similar mechanism is used to control the output of the memory cell, through the use of an output gate (Figure 5.17d and e). The network can also 'flush' (remove) the contents of the memory cell using a special forget gate. In particular, if the forget gate is set to high (Figure 5.17f), the previous state is maintained; if the forget gate is set to low (Figure 5.17g), the content of the memory cell is

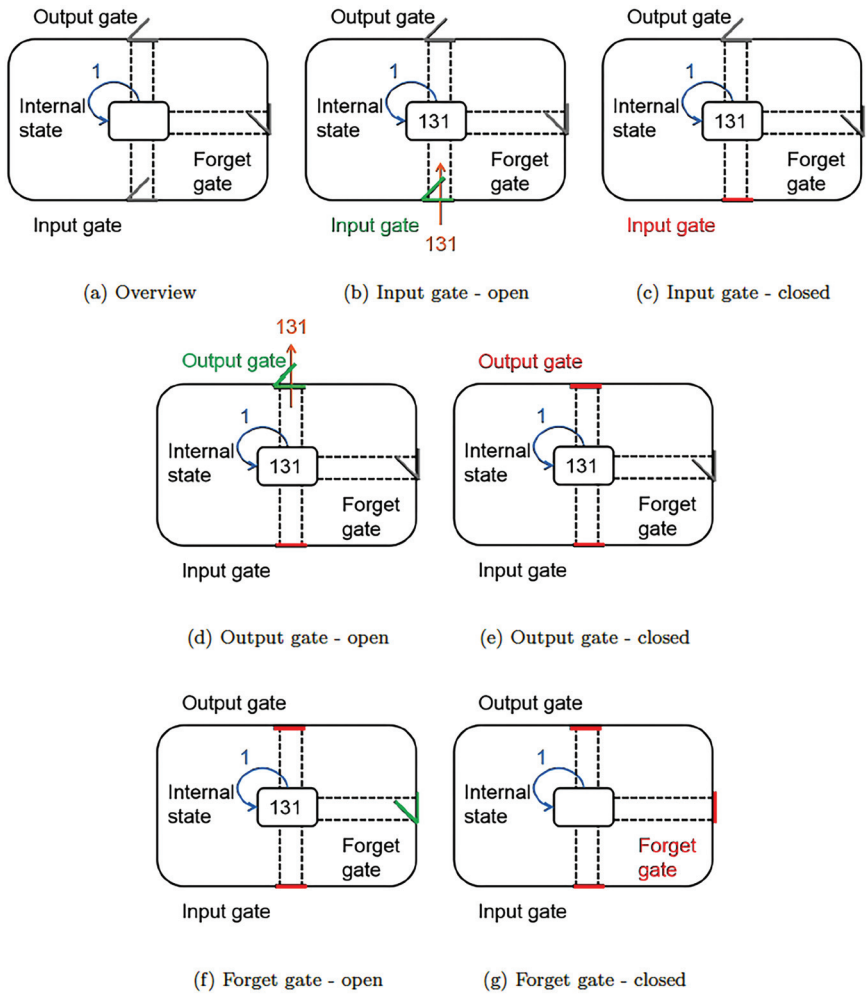


Figure 5.17 Illustration of the components of a LSTM cell.

removed (forgotten). During training, the RNN learns when it is appropriate to remember or forget data using the LSTM cells.

In an RNN, the neurons in the hidden layers (which typically have *sigmoid* or *tanh* activation functions) are replaced by LSTMs, obtaining an entire layer with LSTM cells. Using these cells, an RNN can model long sequences reliably without suffering from the vanishing or exploding gradients problems.

Common applications

RNNs are naturally used to model time series and have been applied successfully in many application domains, including video description [Yuea et al., 2016], speech recognition [Li and Wu, 2015; Li et al., 2016; Cai and Liu, 2016], computer vision [Chen et al., 2016] and diagnosing medical EEG data [Lipton et al., 2016].

Strengths and challenges

- Some strengths of RNN in combination with LSTM are:
  - Deep learning methods, such as RNN, have the ability to learn and model non-linear and complex relationships, which is a major advantage since in practice many of the relationships between inputs and outputs are non-linear as well as complex;
  - Recurrent Neural Networks are suitable for problems that require sequential processing of information;
  - LSTM models are powerful to learn and remember the most important past behaviours, and understand whether those past behaviours are important features in making future predictions;
  - The recurrent connections effectively allow RNNs to share parameters and thus capture long time dependencies using simpler models than deep feed-forward neural networks, making them suitable for cases with smaller data sets.
- On the other hand, the usage of RNN with LSTM provides some challenges:
  - RNN and artificial neural networks in general are very computationally expensive to train. Moreover, RNN are more difficult to parallelize (e.g., using GPUs) than feed-forward networks, due to the sequential dependencies between the states;
  - It is difficult to interpret how the neural network model produces its prediction and to trust the result. For some domains, e.g., medical and fraud investigation, this is unacceptable.

5.4.2.11 Change detection algorithm

General description

Change detection is the problem of finding abrupt changes in data when the probability distribution of a stochastic process or time series changes in order to determine the fault early enough and to provide an early warning alarm [Aminikhanghahi and Cook, 2017].

Change detection algorithm is therefore not a single algorithm but a family of algorithms that can be categorised as follows [Gustafsson, 2000]:

- Methods using one filter, where a whiteness test is applied to the residual
- Methods using two or more filters in parallel, each one following a specific assumption for the change to be detected (Figure 5.18).

A filter in system identification is a mathematical model that maps the relation between the input and output of the system and returns a sequence of residuals. Which filter to be used is a research topic by itself and it depends on the problem under study [Ljung, 1999] and does not matter from a change detection perspective [Gustafsson, 2000]. Examples of filters can be Kalman filter, kernel, parametric models, state space models and physical models.

The basic idea of a change detection system is that if there is no change in the system and the model is correct, given the input and output variables, the result is a sequence of independent stochastic variables with zero mean and know variance. When a change occurs in the system, the model cannot capture anymore the behaviour of the system and this affects the residuals. The essential step now is to define a rule of thumb to indicate whether the residuals are too large and an alarm needs to be generated or not.

This can be formulated into the problem of deciding between two hypotheses where a stopping rule is achieved by low-pass filtering of s_t and comparing the value with a threshold [Gustafsson, 2000]:

$$\begin{aligned} H_0 &: E(s_t) = 0 \\ H_1 &: E(s_t) > 0 \end{aligned} \tag{5.17}$$

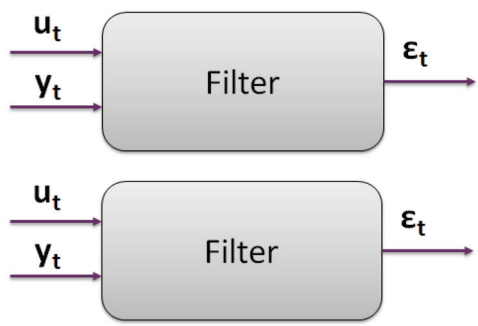


Figure 5.18 Filter that map the relation between inputs u_t and output y_t and generate residuals ε_t.

Below two examples of common rules are given [Gustafsson, 2000]:

- CUMSUM is a sequential analysis technique [Barnard, 1959]. It implies calculating the cumulative sum from the residuals of the filter as follows:

$$\begin{aligned} S_0 &= 0 \\ S_{t+1} &= \max(0, S_{t+1} + s_t - v) \end{aligned} \tag{5.18}$$

When the value of S exceeds a certain threshold value, a change in the timeseries has been found where *v* represents the likely-hood function.

- Geometric Moving Average:
  The Geometric Moving Average calculates the geometric mean of the previous N values in the time series as follows:

$$S_{t+1} = \lambda S_t + (1 - \lambda) s_t \tag{5.19}$$

  Where λ introduces a forgetting factor to account for the slow timevarying behaviour (i.e., degradation) of the system. When the value of S exceeds a certain threshold value, a change in the time-series has been detected.

Therefore, a change detection algorithm based on time-series requires the definition of the model to represent a system and a rule to define when the model does not match anymore the behaviour of the system and therefore an alarm must be triggered.

Common applications

Change detection has been used in a number of applications [Aminikhanghahi and Cook, 2017] from stock analysis, to condition based monitoring, to speech detection and image analysis. Moreover, it has been used for detecting climate change and human activity analysis. Generally, it can be used for any process in which time series are available and a model representing the system can be defined.

Strengths and challenges

- Some strengths of change detection algorithms are:
  - Expert knowledge can be included in the model;
  - Fast technique that can be used also on embedded devices in real-time;
  - Do not require historical data;
  - Can track dynamic change of the behaviour of the system if timevarying parameters are used.

- On the other hand, it also has the limitations listed below:
  - Requires defining or identifying a model of the system of interest;
  - Selection of the stopping rule is not trivial and can vary depending on the system of interest.

5.4.2.12 Fisher's exact test

General description

Fisher's exact test[3] is a statistical significance test used in the analysis of contingency tables. It is in particular suitable for small sample sizes, where an χ-squared test is less appropriate or would yield unreliable results. For a given contingency table, its p-value is given by the sum of all probabilities that are at most the probability of the given contingency table of any other contingency table that is realizable with the original row and column sums in the given contingency table [Fisher, 1922, 1954; Agresti, 1992]

Common applications

The test is useful for categorical data that result from classifying objects in two different ways. As such, it has a broad applicability in many domains. A typical example is where one classification constitutes a so-called *ground truth*, of which the validity is beyond doubt, and the other consists of the output of a machine-learning algorithm. In this case, one normally speaks of a confusion matrix instead of a contingency table.

Strengths and challenges

Fisher's exact test is particularly suitable when the sample size is small. Also, the fact that it is a nonparametric test is a strength, as no assumptions have to made about the shape of the distributions. It is, however, based on the assumption that the row- and sum-totals of the contingency table are fixed, which may be problematic in practice. In the current context, this assumption could be defended. Where, in the social sciences, p-values in the order of 0.01 or 0.001 are common, in the current context, values in the order of $10^{-7}$and even orders of magnitude smaller were often encountered. This has to do with the fact that, in a given dataset, there are many calls where neither a given part is replaced, nor the log pattern occurs. The latter, by the way, demanded an implementation of the binomial coefficients using logarithms and additions instead of multiplications.

[3] https://en.wikipedia.org/wiki/Fisher%27s_exact test

5.4.2.13 Bonferroni correction

General description

Statistical hypothesis testing is based on rejecting a null hypothesis if the likelihood of the observed data under the null hypotheses is low. If multiple hypotheses are tested, the chance of a rare event increases, and therefore, the likelihood of incorrectly rejecting a null hypothesis (i.e., making a Type I error) increases. The Bonferroni correction provides a remedy for this. For a given confidence level α and m hypotheses, by rejecting a hypothesis if its $p$value exceeds α/m, the so-called familywise error rate, i.e., the probability of making at least one type-I error is bounded from above by α [Mittelhammer et al., 2000] (https://en.wikipedia.org/wiki/Bonferroni correction).

Common applications

As this technique is quite general, it is broadly applicable to cases where statistical hypothesis testing is done repeatedly. In the context of alerting and predicting failures, it is one of the key methods by which log patterns are selected.

Strengths and challenges

Its broad applicability and simplicity as well as being non-parametric is an important strength. One of its weaknesses is that it may be overly conservative if there is a large number of tests. The correction comes at the cost of increasing the probability of accepting false hypotheses, i.e., missing good log patterns.

5.4.2.14 Hypothesis testing using univariate parametric statistics

General description

Statistical hypothesis testing allows one to compare two processes by comparing the distributions generated by the random variables that describe those processes [Stuart et al., 1999]. A distribution of a given process may be described by one or more random variables. Univariate statistics refers to the use of statistical models that use only one variable to describe the process. Several univariate statistics can be extended to cater for multi-variate statistics that use two or more random variables (for example, the 2 variable Gaussian processes using the correlation matrix instead of the mean and deviation).

There are two basic methods used in statistical inference. The first establishes a null hypothesis that stipulates the no differences between two

processes (distributions) exist. Inference therefore consists in testing the hypothesis and rejecting the null hypothesis. For example, given the heights of men and women we can test whether or not the mean heights are different. The null hypothesis states that no such difference exists. If we reject this null hypothesis, then the alternative hypothesis, that states that there is a difference, is assumed to be true.

The second consists of modelling each process (distribution) with a set of possible statistical models. Each model is a parametric mathematical equation that describes a possible hypothesis (given a set of random input variables, it outputs a frequency). Inference then consists of evaluating and selecting the most appropriate model. The models are selected according to a specific criterion such as the Akaike information criterion or the Bayes factor [Burnham and Anderson, 2002].

Parametric statistics refers to the use of parametric mathematical models of distributions. Examples of these distributions include both continuous (Uniform, Gaussian, Beta, Exponential, Weibull, etc.) and discrete distributions (Uniform, Binomial, Poisson, Bernoulli, etc.). Statistical inference requires that we estimate the parameters and use this directly or indirectly to compare the distributions.

When performing statistical inference, statistical significance is used to determine how likely it is to assume that the null hypothesis is not rejected under the current circumstances. The statistical significance level establishes a threshold, under which a null hypothesis rejection cannot be accepted. Note that, in addition to this threshold, additional limits on the type 1 (incorrect rejection of a true null hypothesis – false positive) and type 2 errors (incorrect retaining a false null hypothesis – false negative) can be set.

As a title of example, let us consider the p-value and establish the significance level of 0.05 (could be as low as 0.02 or lower). If the p-value of the statistical test is below the threshold, it is highly unlikely (less than 5% of the time) that such a distribution can be observed, and the null hypothesis is true. In this case, the null hypothesis can be rejected. Note however that if it fails, there is not enough information to reject the null hypothesis. Therefore, the test is inconclusive.

The **testing process** is as follows:

- Select the null and alternative hypothesis;
- Establish the assumption under which the test will be made (type of distribution);
- Select the significance level to use as the threshold;

- Select the test statistic to use and calculate it (single sample, two samples, pared test, etc.);
- Compare the "t" statistics with the p-value threshold and decide if it can be rejected or not.

Note that test statistic "t" can be calculated from the observations and compared with the p-value, or it is possible to use the statistical tests to compare the distributions' samples directly. In either case a "t" statistic that is comparable to the p-value is obtained.

The naive Gaussian parametric tests that are used are of two types: one checks for differences in the distribution mean and another one is used to test differences in variance. The variance tests allow us to check that a sensor signal is within a given threshold of a constant (zero for example).

In this case, the t-Test was performed. More concretely, the **t-test** was performed by means of a two-sample location test of the null hypothesis that states that the means of the populations are equal. The independent two sample t-test calculates the t statistic based on the number of samples and the means and variances of each population. The t statistic led to a 2-sided p value that is then used to compare against the threshold that allows either accepting or rejecting the null hypothesis.

Common applications

Statistical tests allow to analyse data collections when no prior theory exists that can be used to predict relationships. In such cases, one can design and implement experiments to evaluate if a hypothesis is likely to be true. In these experiments, data is collected under the conditions of the null hypothesis and the alternate hypothesis under test. For example, to test the effectiveness of a certain pharmacological treatment two data sets are collected, one where no treatment is applied and one wherein it is (note that in the clinical cases usually a third dataset that uses placebos is also used). The effectiveness of the treatment can then be established by testing if there is a statistically significant difference between the two distributions.

Other examples include [Larsen and Stroup, 1976]:

- Determining the effect of the full moon on behaviour (null hypothesis: no full moon);
- Establishing the range at which bats can detect an insect by echo (null hypothesis is the mean detection range);
- Deciding whether or not carpeting results in more infections or not (null hypothesis: no carpets).

Note that depending on the hypothesis that is under test and the experiment set-up, it is either possible to use a single distribution (for example the bat's echo location range) or two (for example testing whether or not carpets cause infections).

Strengths and challenges

The following is a list of general advantages and disadvantages of the parametric tests.

Advantages:

- Simple process;
- Easy to calculate;
- Provides additional information (confidence intervals);
- Some models can be learned (updated) and tested as streams of data (Gaussian model);
- Can deal with skewed data;
- Greater statistical power;
- Robust to samples with different variability (2-sample t-test or one-way ANOVA).

Disadvantages:

- Assumes the type of distribution is known and does not change with time;
- If the null hypothesis is not rejected, no conclusion can be reached;
- Requires large data sets;
- Lack of robustness (sensitive to outliers, sensitive to comparison point[4]);
- Assume all samples are i.i.d (Independent and identically distributed random variable);
- Experiments may produce unexpected effects resulting in failed tests (type 1 and 2);
- Each test has its own assumption (example t-Test of independent means assumes populations of equal variance).

Use in MANTIS

The technique described above was used to detect anomalies that may occur in a press brake machine. Signals indicating the velocity and distance of the

[4]For example, when comparing two Gaussian curves, because the tails fall off at an exponential rate, the difference in distributions is small when close to the mean but very large close to the tail ends.

press ram and back gauge, oil temperature and vibration were used as the univariate variables.

The parametrical statistical tests were used to determine whether new samples conform to the same distribution of previously sampled machine states that are known to be failure free. If the distributions are not equivalent, (null hypothesis rejected) then it was assumed that a failure occurred and alerts are generated.

In the case of the formal statistical tests, if the p-value does not allow us to reject the null hypothesis, it is not possible to infer that the machine has no failure (type 2 error). However, for simplicity, it was assumed that it was true and alerts were only sent when the null hypothesis was successfully rejected.

In the case of a test for a constant (for example position), the signal is normalized prior modelling and testing (a machine may stop its ram at any position, and the interest regards only knowing if it is stopped, and not were). In addition to this, two signals (for example the rams speed at the two different pneumatic pistons) are subtracted one from another. The result is a single sample whose means should be 0 (no need for normalization). The statistical test is then performed as is done for the first case.

Parametric Gaussian tests were performed on the mean (**t-Test**) and deviation (directly compare deviations, F-test, Bonett's test and Levene's test).

The test for a difference in the mean and deviation were also done using the naive statistical test. In this case, an online Gaussian model is obtained via the calculation of a mean and variance. These means and variances are then directly compared to the continually sampled signals from a working machine. If significant divergence is found, alerts are generated. Due to the high false positive and false negative rates (type 1 and type 2 respectively), an additional multiplicative threshold (in respect to one standard deviation) is used when comparing deviations. The initial values of this threshold are set automatically by selecting the lowest possible threshold that reduces type 1 errors.

Due to lack of data, initial tests were performed with artificial data. For large samples the effectiveness of the t-test appeared to be reduced (increased number of false positives and false negatives), and it was also concluded that the naive statistical tests using simple means and variance are very effective. The naive statistical tests were used on a sample of real data first before committing to the more complex t-test. The latest tests on the real data seem to show that the t-test is ineffective for some of the signals. Work in progress is still trying to ascertain why this is so (noise, incorrect labelling

of failure, non-normal distribution, variance does not follow the Chi-square distribution, etc).

5.4.2.15 Hypothesis testing using univariate non-parametric statistics

General description

As with the case of parametric statistics (see hypothesis testing using univariate parametric statistics in Section 5.4.2.14), the process of defining the hypothesis, sampling data and establishing a significance level as a threshold are the same. However, not all of the non-parametric methods provide a p-value for a significance level comparison. This section describes three types of statistical tests that were used in project MANTIS: the *Kolmogorov-Smirnov test* (K-S test), the *Man-Whitney U test* and the use of a *Kernel Density Estimation*. In the first case, the U statistic (as performed by the software library) is approximated by a normal distribution and a p-value is available. In this case, the p-value is used to establish a threshold to accept or reject the null hypothesis (distributions are the same).

In the case of the KDE, an online algorithm was used that generates a configurable number of (Gaussian) kernels. The kernels and respective parameters of two different distributions cannot be directly compared. Experimentation shows that the estimated densities may be visually very similar, but the kernels themselves differ significantly. However, since the kernel can be used to sample the underlying estimated distribution, the statistical tests were used to compare the samples (for both the parametric cases and nonparametric cases using the Kolmogorov-Smirnov test and Mann-Whitney U test). Another option could have been the use of alternate algorithms such as the earth mover's distance, but they were not applied to the case at hand because the naive parametric tests seemed to be working well [Levina and Bickel, 2001].

Kolmogorov-Smirnoff test

This test compares an empirical distribution function (estimate of the CDF) of a sample to a cumulative distribution of a reference distribution. The two sample K-S test can detect differences in both the location and shape of the empirical CDFs of both populations. Given two CDFs (the reference population distribution being a known function or sample), the Kolmogorov–Smirnov statistic is the maximum difference between these two CDFs. The two sample S-K test can be used to compare two arbitrary distribution functions of a single random variable. Here the Kolmogorov– Smirnov statistic

$D_{n,m}$ is the supremum of only two samples (with *n* and *m* being the number of samples in each population). This supremum is used to reject the null hypothesis. The rejection threshold is calculated by applying a significance level (α) to a known function ($c(\alpha)\ ^{p}n + m/nm$)). Note that this expression is an approximation. The critical values D_α are usually generated via other methods and can be made available in a table form [Facchinetti, 2009].

Man-Whitney U-Test [Mann and Whitney, 1947]

This test allows to check if a random variable is stochastically larger than another one. The test is done by calculating the U statistic, which has a known distribution. Usually the U statistic is calculated directly (see next paragraph) and not from the distribution itself. For large datasets however, the U statistic can be approximated by a normal distribution. The large U statistic associated to a given dataset indicates that the dataset's random variable is larger.

Two methods exist to calculate the U statistic: The first can be used in small data sets. In this case all pairs are compared, to sum the number of times the first sample is greater than the second (ties are assigned a value of 0.5). This sum is the U statistic U1. The process is repeated by swapping the pairs that are compared. This U statistic is U2. In the second case, both datasets are joined and ranked. The ranks of all elements from the first dataset are the summed to obtain the U1. U2 can now be calculated based on the total number elements in each dataset. In either case, the large U value indicates which dataset is stochastically higher.

The advantages of this test compared to the t-test is that it can be used for ordinal data, is robust against outlier (based on rank) and is a better estimator (higher efficiency) for large datasets that are not normally distributed (for normal distributions its efficiency is 5% below that of the t-test). Note that this test can also be used when comparing two populations of different distributions.

Kernel density estimation [Rosenblatt, 1954; Parzen, 1962]

Also known as Parzen–Rosenblatt window method, this is a data smoothing technique that estimates the shape of a function of unknown density.

The kernel density estimator is a sum of scaled kernel functions. A kernel function is a function that has an integral of 1. Density estimation consists of identifying the type and number of kernels to use and determining the kernels and smoothing parameters (also referred to as the bandwidth). These parameters are selected in order to reduce an error, usually the MISE. This process of estimating the density has similarities to that of manifold learning,

which includes among other techniques auto-encoders (see category Artificial Neural Network, technique auto-encoders in Section 5.4.2.8).

Several kernels exist, such as the uniform, triangular, Epanechnikov and normal functions, among others. Usually a normal (Gaussian) kernel is used for mathematical convenience. Its efficiency is close to that of the Epanechnikov, which is optimal with respect to MISE.

The selection of the parameters, based on the MISE, is not possible because it requires prior knowledge of the density function that is being estimated. Several techniques have been developed [Xu et al., 2015], which include the most effective plug-in and cross validation selectors. However, bandwidth selection for heavy-tailed distributions is difficult.

Besides the various techniques used to estimate the various parameters such as the bandwidth, kernel density estimators also differ in the number of random variables modelled (uni- or multi variate), the use of constant or variable bandwidth (variable kernel density estimation) and the use of a single kernel per data-point or a variable (optimal) number of kernels.

Common applications

Function smoothing in general has many uses that include:

- Kernel smoothing allows to estimate underlying function of a noisy sample;
- Kernel regression uses the learned function for prediction (non-linear regression);
- Conditional probability density estimates can be obtained from the data by estimating the density functions of the data that are selected according the desired conditionals (Bayes theorem);
- The maxima can be used for data analysis and machine learning such as clustering and image processing (multivariate kernel density estimation).

Strengths and challenges

Advantages

- Does not require the process to be Gaussian;
- Can be used in small datasets;
- Some tests have simple calculations and are easy to understand. MannWhitney U test is based on the more robust median (instead of the mean of the parametric tests, therefore not so sensitive to outliers);
- Mann-Whitney U test can analyse ordinal and ranked data (for example, data using the Likert scale use the de Winter and Dodou).

Disadvantages

- Lower probability of detecting an effect as compared to parametric methods;
- May have additional assumptions that are required of the dataset (for example the Mann-Whitney U-Test requires that the distribution be continuous and that the alternative hypothesis being tested represent the distribution is stochastically greater than that of the null hypothesis.);
- Still require the estimation of parameters for model generation;
- These tests are usually less efficient than the parameterized versions;
- CPU intensive (KDE);
- The K-S test requires large data sets to correctly reject the null hypothesis.

Use in MANTIS

The technique described above was used to detect anomalies that may occur in a press brake machine. Signals indicating the velocity and distance of the press ram and back gauge, oil temperature and vibration were used as the univariate variables.

The non-parametric statistical tests were used to determine whether new samples conform to the same distribution of previously sampled machine states that are known to be failure free. If the distributions are not equivalent, (null hypothesis rejected) then it can be assumed that a failure occurred and alerts are generated.

When testing for a constant (for example position), the signal is normalized prior modelling and testing (a machine may stop its ram at any position, we are only interested in knowing if it is stopped, but not where it happened). In addition to this, two signals (for example the rams speed at the two different pneumatic pistons) are subtracted one from another. The result is a single sample whose means should be 0 (no need for normalization). The statistical test is then performed as it is done for the first case.

The Mann Whitney U test was applied directly to the previously recorded samples of a correctly functioning machine and new samples were obtained from a working machine. As with the parametric case, a p-value was set to reduce the type 1 errors. In the case of the KDE, a model was generated for a sample of the correctly function machine. This model was used to generate a reference sample. The reference sample was then compared to the new working machine samples. Once again, Mann Whitney U test was applied as previously described.

Of all the parametric and non-parametric tests, the Mann-Whitney seemed to be the most robust and effective method. The KDE was tested by applying the t-Test, the Kolmogorov-Smirnov test and the Mann-Whitney U test. The result was no better than applying the Mann-Whitney U test directly. The Mann-Whitney U test is used were the naive statistical tests fails (note that it is necessary to use alternate tests when comparing variability because all the tests discussed here only deal with the mean).

5.4.2.16 Mean, thresholds, normality tests

General description

Descriptive statistics [Mann, 1995] is often used to provide a basic analysis of an unknown dataset and identify the basic properties of the data that can be used to select proper techniques to perform a deeper investigation and build statistical models. Moreover, such analyses are useful to identify the characteristics that allow the use of specific statistical approaches. The most common are the following:

- *Mean:* it measures the average value of a dataset;
- *Median:* it measures the central value of a dataset (half of the dataset is greater than the median value and the other half is smaller). In many cases, it is preferred compared to the mean since it is less affected by outliers;
- *Quartiles:* they provide a basic understanding on how the values are distributed. Usually, they are calculated to identify where the 25%, 50%, 75% of the data reside;
- *Variance:* it measures the variability and the spread of the data.

After this preliminary set of analyses, it is required to investigate more deeply the distribution of the data. In particular, it is required to understand if the distribution is normal (or Gaussian) since many statistical approaches can be used only if the data distribution is of this kind.

To test normality, there are many different approaches but the most used is the Shapiro-Wilk test [Shapiro and Wilk, 1965] (another popular but less powerful test is the Kolmogorov–Smirnov one [Wayne, 1990]). After verifying whether a dataset follows a normal distribution, it is possible to select properly the statistical tools to perform further investigations.

To define “normal” and “abnormal” values for a variable in a dataset, it is required to define thresholds, and it requires a dataset that includes a large number of repeated measures of the variable. Such measures should be labelled as “normal” or “abnormal” since the threshold can be built using

the "normal" measures and verified against both "normal" and "abnormal" ones to build a good descriptive model. In addition, in this case there are several approaches and one that is very popular when dealing with data not distributed normally is the Tukey's range test [Tukey, 1949]. This technique allows the definition of thresholds for outliers, such as the values outside the range are candidate to be considered "abnormal".

Common applications

Descriptive statistics is used as a preliminary investigation in any analysis of an unknown dataset regardless of the application domain. They include basic measures that are useful to characterize the data and decide further actions (e.g., subsequent analysis, split/merge of datasets, apply transformations, etc.).

After the extraction of descriptive statistics, testing the data for normality distribution is the second most common activity in any kind of application domains if the distribution of the data is not known. It is also possible to test the compliance with other distributions, but several statistical methods require the normality distribution of the data to be applicable. Therefore, this kind of tests is performed as a pre-requisite for the identification of the proper statistical tool to apply.

Thresholds definition using different kinds of approaches is frequently used to perform statistical process control, to define statistical models, and to evaluate their performances using only data from the field (e.g., historical data) and perform online checks of the behaviour of a system to rise alerts in case of violations. The developed models can also be used to make predictions of violations in conjunction with other statistical techniques that are able to identify trends in data (e.g., using time series analysis techniques).

Strengths and challenges

Strengths

- Simple mathematical approach;
- The definition of the model requires a limited amount of data;
- The model can be generated on-the-fly and adapted to different working conditions;
- Models can be generated using limited computational resources.

Challenges

- Can be sensitive to noise;
- Requires fully labelled data.

Use in MANTIS

The described techniques have been used for the characterization of the datasets coming from railway switches and to define their expected behaviour based on available data. In particular, the used data regarded the adsorbed current of the motor and the position of the switch over time to define the profiles of correct behaviour through the analysis of the variability of the data at each time instant. The distribution of the samples from different movements of the switch at each time instant was analysed, followed by the definition of thresholds of acceptable values based on an analysis of the outliers. Moreover, since the collected data are affected by the temperature of the environment, the adopted approach can continuously adapt itself using the data from the latest movements that were considered correct. Finally, the work on the project also proved that abnormal behaviours can be easily detected since the correct profiles have a low percentage of samples that are outside the threshold defined (less than 5%) while the abnormal ones have a much higher percentage and/or the duration of the movement is very different from the expected one.

5.5 Examples

This section entails some example use cases of the unsupervised algorithms and the recurrent neural networks for analysing and predicting temporal message log respectively, and investigates the use of convolutional neural network architectures for predicting textures of the metal surface, which is very useful for maintenance.

5.5.1 Usage Patterns/k-means

In the context of the MANTIS project, a small pilot experiment was performed by Liebher on a dataset containing log files with messages and time stamps for about 900 machines of two types. Assume a machine is a stochastic message generator, assume for the time being that the order is not important (in other words, that the order is of secondary importance if compared to the presence vs absence of a message). The message patterns emitted by the machine are a function of machine type, part characteristics, usage conditions, possibly by the conditions of repair (quality of the servicer/mechanic). This research attempted to answer the following questions.

- Are there groups (clusters) of message pattern generators?
- Are there machines that have a pattern of message generation?

- How many of such groups do exist in the data? Is there a relation to the service mechanic? Is there a relation to the customer? Is there a relation to the type of the machine?

5.5.1.1 Data analysis

Data were preprocessed in the following sequence;

- Extracting machine-specific information;
- Computation of message histogram row counts and message probabilities per machine;
- Use of own K-means clustering script for computing k centroid vectors for the message probabilities to identify patterns in the machine generators;
- Computation of an optimal number of clusters (machine groups). Figure 5.19 shows the Euclidean distance between cluster centroids and the input data. The figure represents the error curve for clustering results as a function of the expected number of customer groups. Average Euclidean distance between message histograms of k-means centroids and machine instances, respectively, was used. Each clustering attempt for a value k was performed 20 times and the mean curve as well as the standard-deviation band are shown. There is no 'knee' or convincing discontinuity indicating the optimal number of clusters. The variants of this method showed a similar behaviour (silhouette method,

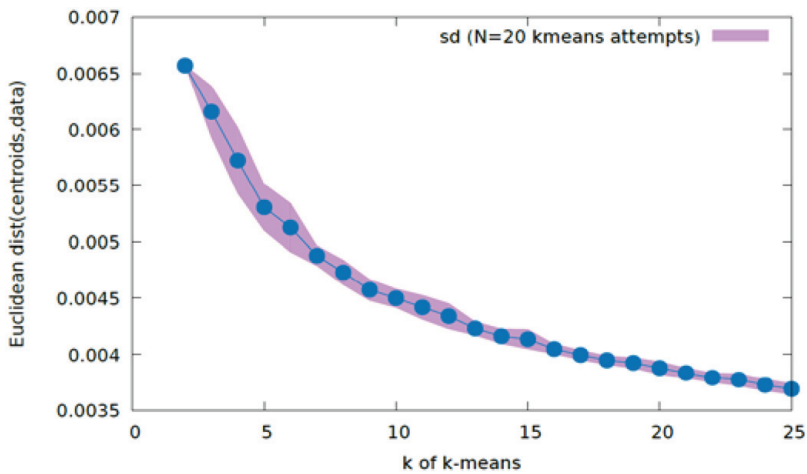


Figure 5.19 Error curve for clustering results as a function of the expected number of clusters k.

gap statistic). Hence, a chi-square was experimented with, which has, to our knowledge, not been used in this manner before. The rationale is that if there is an underlying distribution of k clusters in the data, there should be a distribution of instances over clusters that is deviating from a uniform or expected distribution. Conversely, if there is no underlying cluster structure, the assignment of instances to clusters is random over several Monte-Carlo clustering attempts and a statistical frequency test should yield non-significant results;

- Using χ 2 for testing cluster independence;
- Computation of factors such as : machine type, customer ID, servicerID;
- Random trials of k-means estimations for a range of *k*;
- Analysis of contingency table significance of interesting factor interactions. For this step, the research question was: what will be the average estimate of the significance of the count independence in the selected table (cls x cust or cls x serv)? In order to get reliable results, the average p values are calculated over one hundred (100) independent k-means model estimations for each of the $k = [2, \ldots, 25]$ values. For instance, Table 5.4 reports the ClusterID vs MachTyp: cls x typ (the other, cust and srv tables would be too big horizontally to fit here);
- The p value was extracted from the contab log file and was averaged over the 100 random retries of the kmeans estimation. In Table 5.4, p equals 'zero', because of the trivial fact that the messages for MachTyp=1 and

Table 5.4 ClusterID vs MachTyp contingency table

	typ2	typ1	Totals			
c1	115	0	115			
c2	267	9	276			
c3	103	0	103			
c4	4	0	4			
c5	27	6	33			
c6	53	0	53			
c7	4	0	4			
c8	52	1	53			
c9	142	34	176			
c10	38	67	105			
Totals	805	117	922			
Analysis	for cls	x typ:				
chisq		323.812158		df	9	P 0.000000
Cramer's		V				0.592627

Source: cls typ

MachTyp=2 are significantly different: These are physically different machines, each type sending messages with its own pattern statistics;
- Average expected significance probability: compute the averages for p values, their standard deviation (sd) and standard error (se). The results obtained are discussed in the next section.

5.5.1.2 Results

Average significance of clustering results for different k, for interaction 'cluster x customer' in Table 5.5.

5.5.1.2.1 *Plotting*

The data information from column 1 and 3 in Table 5.5 are plotted in a histogram as shown in Figure 5.20, with k on the x axis and avg p signif on the y axis. The left plot of Figure 5.20 is the average p-significance levels as a function of k (number of groups) for the interaction (clusters x customers). There is a clear minimum at $k = 7$, indicating that for this number of groups the distribution of machines over (clusters x customers) is not likely to be random, in case of seven groups of prototypical message generators. Although the minimum value ($p = 0.0588$) is above the commonly maintained '0.05', the result is sufficiently different from the other values to be actually meaningful (error = 0.0143). The right plot is the average p-significance levels as a function of k (number of groups), for the interaction (clusters x servicers) There is no tendency for non-random distributions, thus it is highly unlikely to find significant differences in the distribution of machines over bins defined by (clusters x servicers).

Table 5.5 Results of multiple Monte Carlo experiments (N = 100), for different k. A low average p-significance level indicates an underlying structure for this value of k

k	N	Avg_p_signif	sd	se	
2	100	0.2674	0.3294	0.0329	
3	100	0.1627	0.2307	0.0231	
4	100	0.1763	0.2902	0.0290	
5	100	0.1340	0.2811	0.0281	
6	100	0.1245	0.2708	0.0271	
7	100	0.0588	0.1426	0.0143	<==
8	100	0.1543	0.2932	0.0293	
9	100	0.2029	0.3608	0.0361	
...	...	...	...	...	
25	100	0.2630	0.3748	0.0375	

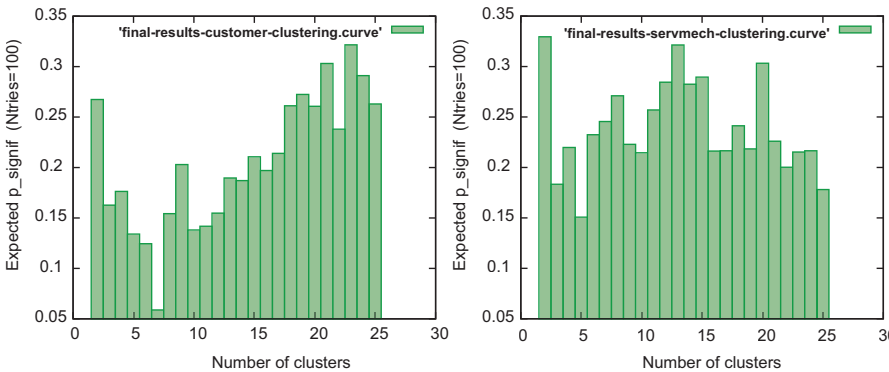


Figure 5.20 Average p-significance levels as a function of number of groups for interactions (clusters × customers) on the left, and (clusters × servicers) on the right.

5.5.1.2.2 *Replicability of results*

Since the k-means algorithm entails random initialisation and the data scattering is noisy by itself, the question is warranted what a repetition of the whole procedure would yield. When repeating the complete procedure 24 times, the average value for k was 6.6 with a standard deviation of 1.5 (see Table 5.6).

However, apart from the 'best' number of clusters, the most interesting finding is the contrast between the 'Customers' and 'Servicers' factors, where the latter do not seem to induce a clear clustering of message patterns generated by the machine instances.

5.5.1.2.3 *Summary of results*

There is a very strong relation $p < 0.000001$ between cls x typ (cluster-ID x machine type) i.e., the algorithm detects that machine type 1 and 2

Table 5.6 Results of k-means algorithm repetition

k	Freq	
4	1	*
5	2	**
6	12	**********
7	4	****
8	2	**
9	2	**
10	0	
11	1	*

generate different message patterns, in all possible cluster forms, with k in the range [2, 25]. There are no different messaging patterns for the machines in the clusters with respect to Servicer-ID. There is a clear significance if the counts in the clusters are based on Customer-ID. This occurs when the data is segmented into $k = 6 \pm 1$ clusters (see Table 5.6).

Since there are only two machine types, a grouping into six or seven prototypical message patterns cannot be the result of machine type alone. **Customers** may be using machines differently, leading to different message patterns for each of the user groups. On the other hand, for the **Servicers**, no indications are found for a cluster structure with different machine-messaging patterns for different clusters. Clustering results varies slightly over several estimations, but not drastically. Repeated runs (i.e., again a full set of 100 tests per value of k from 2–25) yielded comparable results.

5.5.2 Message Log Prediction Using LSTM

An efficient approach to time-series prediction of message code sequences generated from the Litronic machine (for Liebherr) is proposed. For this aim, the choice fell on an Integrated Dimensionality-Reduction LSTM (ID-LSTM) that mainly consist of one encoder LSTM, a chain decoder with three LSTMs, one repeat-vector that connects the encoding and decoding sections of the network architecture, and a time-distributed wrapper. Preliminary experiments were carried out using separate single layer predictive models (SL-LSTM or SL-GRU) and ID-LSTM for predicting two or more forms of the message code representations. The best approach from this earlier investigation is used to analyse all the message codes by considering ten unique subsets of the same size that sum up to the entire message codes. The results show that ID-LSTM using one-hot-encoding data-representation yields a performance that surpasses all other approaches on the small sample of the data. Additionally, the average prediction accuracy for the ten unique subsets of the entire message codes during training and testing phases of the proposed method is very good.

There exist groups of message pattern generator, and there is a need to know how well a neural network can predict this pattern. The resulting questions to be answered regard how we should encode arbitrary nominal machine codes to be useable in a vectorial processing paradigm, and whether we can predict error codes or not.

5.5.2.1 Data interpretation and representation

This section describes the Litronic dataset, and provides a brief description for each of the message code data-representations passed as input to the different predictive models.

5.5.2.1.1 *Litronic dataset*

This dataset contains several machine code generator. Each row within the dataset describes nine unique data fields that provide the machine ID, message code, time information and some other industrial properties. The total number of timestamps present in the dataset is 15, 461, 890 (i.e., 15M) which also corresponds to the total-time counts.

In the experiments, we considered two forms of data sizes:

- A small number of samples: this data size contains one machine ID (mach4545) over a short duration of 10,000 time-counts;
- A more significant number of samples: this data size involves a collection of different machine IDs with unique message codes for ten different subsets but with the same amount of data samples. Note that each subset uses the same number of time-counts 1.546×10^6. This sample size is convenient for a memory-based model estimation. We remark that the sum of each subset time-counts yields the total time-counts of 15.46×10^6, which is ten times as large.

5.5.2.1.2 *Data representation*

We considered three forms of data-representation, individually passed as input to the predictive models.

- Raw Code: only the raw message code is used for the time-series prediction with no transformation or pre-processing on the raw code. Some of the unique sets of message code include: $\{464, 465, 466, ..., 946, ..., 31662\}$;
- OHE Code: these codes are obtained by transforming the RC to an index encoded integers; then the output is further processed to a categorical representation of the message codes. Note the final versions of these message codes contain binary values $\{0, 1\}$. Note that the dimensionality of the feature vector of the encoding is dependent on the amount of the index codes for a given number of time counts. Note that for the entire 15,461,890 samples of codes, the vector matrix of the OHE is $R^{15,461,890 \times 1304}$, that is, there are 1304 codes (unique index);

- PCA Code: these codes are obtained by using a linear dimensionality reduction (DR) algorithm to reduce the feature dimensionality of the OHE codes into a reduced feature space that contains the most important information, based on the covariance structure of the data.

5.5.2.2 Predictive models

We employed Single Layer Predictive Models, namely LSTMs and GRU, which are mainly designed to avoid the long-term dependency problem.

An approach to sequence prediction using an integrated dimensionality reduction LSTM model was devised. The model is set-up using one encoder LSTM interconnected to a decoder section (that comprises three LSTMs) using a repeat-vector algorithm. The first-two LSTMs in the decoder section help in the reconstruction process of the squeezed feature vector output from the encoding section into a lower dimensional feature space. Later on, the final feature x from the last LSTM is wrapped with a time distributed algorithm that presents the reproduced data in a sequential series; this was achieved using a fully connected layer $f\ (x) = Wx + b$, that is activated using a rectified linear units, or RELU, activation function $max(0,f\ (x))$. Note that all the LSTMs use a return sequence (that is set to TRUE). Moreover, each of the LSTM uses 10 output nodes except for the last LSTM that contains output nodes that corresponds to the exact number of the input feature dimension, and is dependent on the input data-representation if they exist as either OHE or PCA codes. Figure 5.21 shows the block diagram of the integrated dimensionality-reduction LSTM architecture. The bold lines indicate the network connection while the dotted lines show the non-network connection, which represent data copying.

The network is compiled using an Adam optimizer [Kingma and Jimmy, 2014], while training for specific conditions depending on the data representation and data-size:

- Small sample data represented in OHE codes: trained for 100 epochs and setting the batch size to 10;
- Small sample data represented in PCA codes: trained for either 100 or 200 epochs and setting the batch size to 10;
- Large sample data represented in OHE codes: trained for 100 epochs and setting batch size to 10,000, presents a significant computational benefit as it computes faster using higher batch size compared to lower batch size (which require more extended training time).

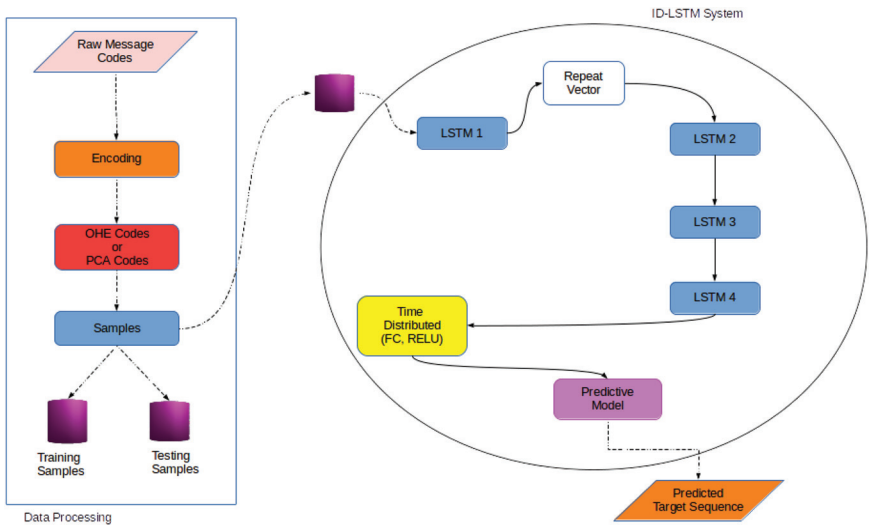


Figure 5.21 Block diagram describing the predictive recurrent neural network using an Integrated Dimensionality LSTM architecture applied on Litronic message codes.

Note that training the proposed network on the PCA codes presents varying trainable parameters, and this is due to the variation in the PCA feature dimensions $\{1, 2, 3, 4, 5, 10, 20, 40\}$ on the small data samples. In this method, we consider 70% samples of the data-distribution for training and the remaining 30% for testing, for each of the used data sizes (small or large).

5.5.2.3 Results

This section reports on performances of the various predictive models with respect to the used data sizes.

5.5.2.3.1 *Evaluation of predictive models on small number of samples*

The performances (accuracies and output-target plots) for the proposed predictive method on OHE codes are represented graphically in Figure 5.22. The experiments considered a small number of samples, and featured 10 K time samples, of 64 possible message codes. The upper left graph (Figure 5.22a) shows the ID-LSTM prediction accuracy for both training and testing phases while training the network for 100 epochs. The upper right graph (Figure 5.22b) shows a plot of the index prediction in the training and testing phases to their corresponding target values. The bottom left graph

(Figure 5.22c) and bottom right graph (Figure 5.22d) represent the confusion matrix, by showing the plot of the output predictions against their target values for both training and testing phases respectively. Additionally, the several predictive model's evaluations on the different data representations are reported in Table 5.7.

ID-LSTM Evaluation on OHE Codes

As can be observed from Table 5.7, the use of ID-LSTM on OHE yields a performance that surpasses all other approaches: the method shows a high level of precision in the predictive values compared to their corresponding target values. Other interesting techniques involve the use of ID-LSTM examined on high order dimensional PCA codes, which persistently outperform both ID-LSTM or single layer predictive models when applied on one-dimensional PCA versions or raw codes.

The second best approach is the ID-LSTM on 20-dimensional PCA codes; this is due to their performance in the testing phase. There exist some failure in the prediction for few epoch instances in the testing phase. An exception

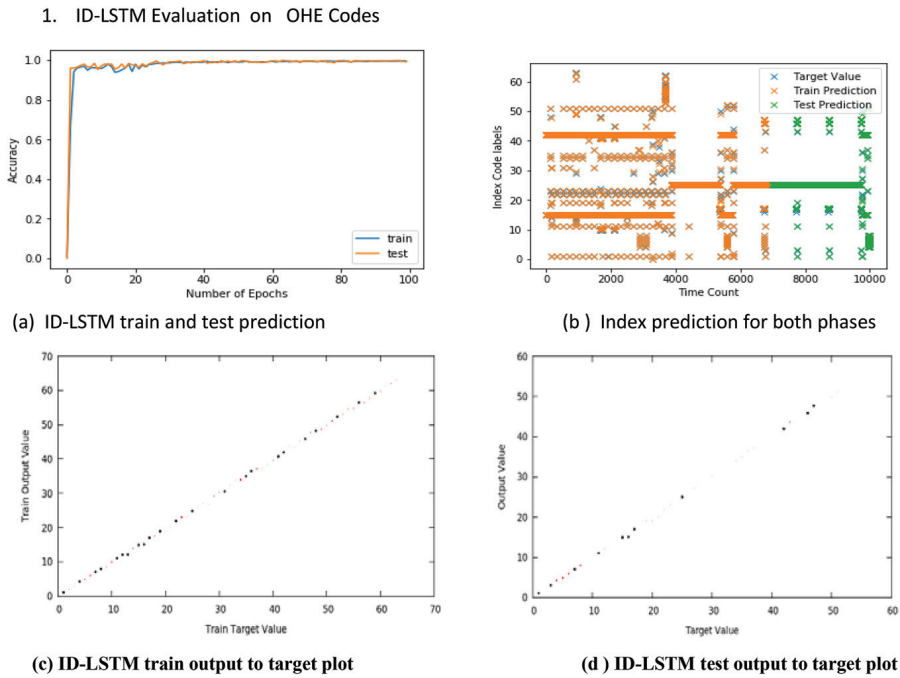


Figure 5.22 Result on small pilot test 10K time samples, 64 possible message codes.

Table 5.7 Training performance metrics on the different approaches using a batch size of 10

Method	Train Accuracy	Test Accuracy
ID-LSTM-I-OHE-Codes	0.9957	0.9920
ID-LSTM-I-20-DIM-PCA-Codes	0.9763	0.9843
ID-LSTM-I-40-DIM-PCA-Codes	0.9760	0.9733
ID-LSTM-I-10-DIM-PCA-Codes	0.9316	0.9727
ID-LSTM-I-5-DIM-PCA-Codes	0.9139	0.9593
ID-LSTM-I-4-DIM-PCA-Codes	0.9424	0.9410
ID-LSTM-I-3-DIM-PCA-Codes	0.9463	0.9593
ID-LSTM-I-2-DIM-PCA-Codes	0.9424	0.9590
ID-LSTM-I-1-DIM-PCA-Codes	0.8729	0.9340
SL-LSTM-I-1-DIM-PCA-Codes	0.8757	0.9340
SL-GRU-MSE-SI-1-DIM-PCA-Codes	0.8715	0.9316
SL-GRU-MAE-SI-1-DIM-PCA-Codes	0.8715	0.9316
SL-LSTM-MAE-SI-1-DIM-PCA-Codes	0.8715	0.9316
SL-LSTM-MSE-SI-1-DIM-PCA-Codes	0.8715	0.9316
SL-LSTM-I-Raw-Codes	$1.429 \times 10\hat{}-4$	0.0000
SL-GRU-MSE-SI-Raw-Codes	$2.858 \times 10\hat{}-4$	0.0000
SL-GRU-MAE-SI-Raw-Codes	$2.858 \times 10\hat{}-4$	0.0000
SL-LSTM-MSE-SI-Raw-Codes	$2.858 \times 10\hat{}-4$	0.0000
SL-LSTM-MAE-SI-Raw-Codes	$1.429 \times 10\hat{}-4$	0.0000

to this is the ID-LSTM on 20 or 1-dimensional PCA codes. Based on this observation, it can be deduced that it is essential to optimize PCA codes with different feature dimensionality to solve a large-scale sequence prediction problem; this will aid to provide a predictive model that will be free from prediction failure in the testing phase. Moreover, this allows to retain useful covariance information that can provide an alternative basis for good temporal prediction. Even the external cases of applying PCA and only using largest eigenvector yielded a reasonable performance of 93%.

Hence the experimental result indicates that the use of OHE or PCA codes with their respective index provide a better basis for temporal prediction. Due to the performance of the best approach, ID-LSTM only was employed to examine OHE codes from the more significant amount of the message sequence.

5.5.2.3.2 *Evaluation of the ID-LSTM on OHE codes for more significant number of samples*

In this subsection, the proposed network was trained on ten different subsets that sum up to the entire message codes. The ID-LSTM evaluation on 196 machines from subset 9, is shown in Figure 5.23 with subfigures that

describe the prediction accuracy curve, index prediction, and the confusion matrix plot in both training and testing phases respectively. In particular, the upper left graph (Figure 5.23a) shows the ID-LSTM prediction accuracy for both training and testing phases while training the network on subset 9 for 100 epochs. The upper right graph (Figure 5.23b) shows a plot of the index prediction in the training and testing phases to their corresponding target values. The bottom left graph (Figure 5.23c) and bottom right graph (Figure 5.23d) represent the confusion matrix, by showing the plot of the output predictions against their target values for both training and testing phases respectively. The performance metrics obtained after 100 epochs for the different subsets are reported in Table 5.8. In the training phase each of the models performed very well with an average accuracy of 0.9844, while the average test performance is 0.9506. The proposed predictive model can handle very complex sequential problem irrespective of the size of data (time-counts). Please note that the lower performance in the test phase may have arisen due to variation in the codes that were used during training of the

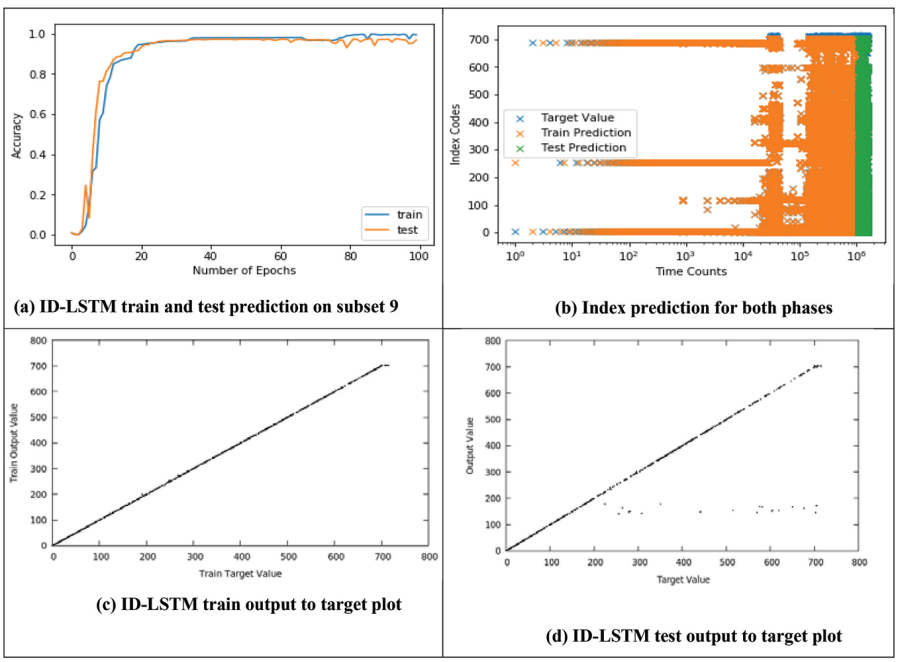


Figure 5.23 ID-LSTM prediction accuracy for both training and testing phases while training the network on subset 9 for 100 epochs.

Table 5.8 Train and Test performance assessment on the different subsets using a batch size set to 10,000

Data Type	Time counts	No. of Machines	No. of Index (class label)	Train Accuracy	Test Accuracy
Subset 1	0.00-1.54M	20	948	0.9826	0.9751
Subset 2	1.54-3.09M	30	606	0.9979	0.9695
Subset 3	3.09-4.63M	36	535	0.9886	0.9624
Subset 4	4.63-6.18M	48	619	0.9961	0.9021
Subset 5	6.18-7.73M	62	620	0.9837	0.9806
Subset 6	7.73-9.27M	109	675	0.9962	0.9347
Subset 7	9.27-10.8M	64	648	0.9205	0.9293
Subset 8	10.8-12.3M	95	679	0.9973	0.9576
Subset 9	12.3-13.9M	196	717	0.9943	0.9681
Subset 10	13.9-15.4M	263	624	0.9871	0.9268
Average				0.9844	0.9506

proposed network. Both the training and testing phases show outliers. Some of the outlier points represent few instances of absolute average accuracy error in prediction is $< 5\%$ for both phases. This performance measure indicates that the proposed method yields a good result.

5.5.2.4 Discussion

This research has addressed the two questions in Section 5.5.2: The research demonstrated the transformation of arbitrary machine codes (raw data) into a useable vector representation, with the objective to identify correlated patterns. To deal with this problem, OHE was used to convert the nominal codes to a vector representation. However, the vector dimensionality is high, and correlations are to be expected between message patterns. To reduce the dimensionality to a lower feature space, PCA was employed to select the most informative dimensions. The best result was obtained on OHE code, however with the risk of overfitting depending on the number of message sample and lexicon sizes. Additionally, the results obtained with PCA transformed data are very good, as the use of only 20-dimensional PCA yielded a prediction accuracy of 98%. However, the use of raw codes with no transformation yields the worst result; this implies that the raw codes are not suitable or sensible to the neural networks.

Furthermore, we investigated whether predictive recurrent neural networks can be trained and tested on datasets of varying samples sizes, message lexicon size, and underlying machines. For the experiment led to an integrated dimensionality reduction LSTM (ID-LSTM) and external LSTM

(SL-LSTM or SL-GRU) for predicting the discussed data. The results show that in the pilot experiment on the small samples, the use of the proposed method (ID-LSTM on OHE codes) yields performance that surpasses all other approaches. ID-LSTM also obtain better performance on higher order dimensional PCA codes than on one-dimensional PCA or raw codes. It is possible to conclude that, for a large data sample with PCA code, there is the is a need to optimize the dimensionality to obtain good performance. Moreover, the ID-LSTM on OHE codes obtained $< 5\%$ error on the predicted codes in a realistically large dataset.

The research suggests that it may be possible to combine the proposed model with an early anomaly detection algorithm to allow continuous prediction of physical problems in the machines generating the message logs. Finally, with advance knowledge on the message log data, it is possible to develop a precise system that can handle detection of an anomaly in the message log, and hence providing a comprehensive basis for the root-cause analysis.

5.5.3 Metal-defect Classification

This section describes using deep learning techniques for predicting textures of metal surface, which is very useful for maintenance. The defects on the surface of, e.g., a cutting tool in a production line are detrimental to the quality of the end product produced by the tool. Therefore, it is important to inspect the cutting tool regularly to maintain the quality of the product, which is usually done manually by human experts. However, manual inspection of cutting tools on production line is costly and difficult whereby the entire process slows down. Some defects are hard to measure by digital sensors, but might be easily spotted by the naked eye. For example, the pollution of tool surface due to scraped metal particles can be detected by a camera. Visual inspection is a promising way to automatically find out defects in the cutting surface in real-time in production lines. Images of metal surfaces of cutting tools can be obtained with sensors, such as the digital microscope. Different defects on metal surfaces result in different types of textures and recognizing the textural surface helps to reveal original (root) causes. For example, if a burnt texture is found on the metal surface, the possible reason is that the temperature is high and the cooling system is failing. The defect textures of metal surfaces are usually different on different metal materials. However, there are some generally known and well-documented defect textures in metal surfaces, which provide a good indication of the root causes. In this pilot project, CNNs are used for metal texture classification. Since data sets with

labels are hard to collect from the real-word production line, a public metal surface defect data set was used and later a new data set from metal images with different textures was collected.

5.5.3.1 Data collection

The NEU surface defect database is a public data set[5], which contains six kinds of typical surface defects of the hot-rolled steel strip. It has 300 images per surface defect type and 1800 greyscale images in total, where the size of each image is 200x200 pixels. This image size is very suitable for the application of CNNs. Examples of images are shown in Figure 5.24 (left). A new data set named MTEXTURE was also collected. Example images are shown in Figure 5.24 (right).This data set has seven texture attributes, which are related to defect textures on metal surface. The size of images is 100 $\times$ 100 with greyscale. Table 5.9 below shows the attributes of each data set.

5.5.3.2 Experiments

The AlexNet [Krizhevsky et al., 2012] CNN has five convolutional layers with Leakly-ReLU as activation function. The Leakly-ReLU is defined as:

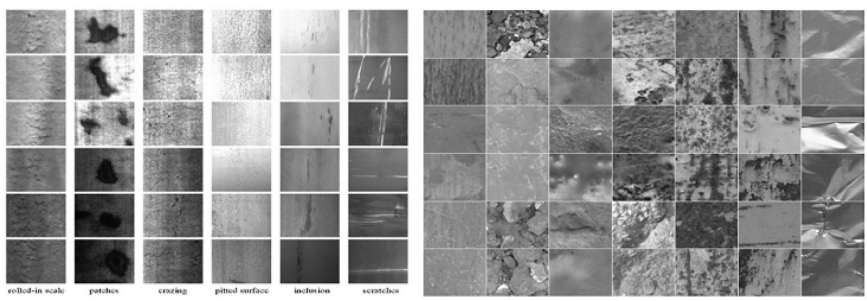


Figure 5.24 Example of images in the NEU data set (left) and the newly collected MTEXTURE data set (right).

Table 5.9 Attributes in the two data sets

Database	Attribute						
	1	2	3	4	5	6	7
NEU	Rolled-in scale	Patches	Crazing	Pitted	Inclusion	scratches	
MTEXTURE	Worn	Debris	Dented	Rough	Rust	Scratched	Crumpled
Attribute Database							

[5]http://faculty.neu.edu.cn/yunhyan/NEU surface defect database.html

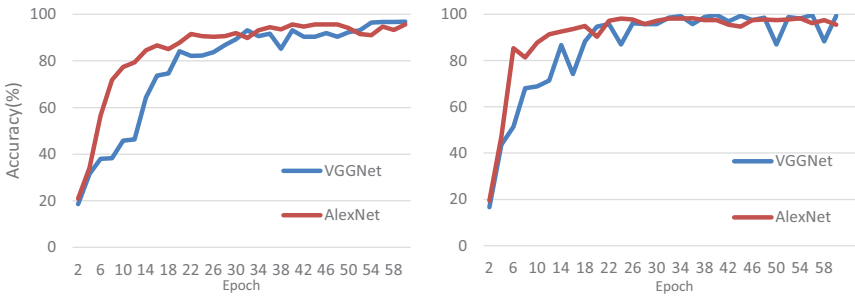


Figure 5.25 Performance of VGGnet and AlexNet on NEU[6] (left) and the collect set (right) with different epochs.

$y = \max(x, \lambda x)$, where x is the input, y is the output and λ is a constant parameter which is set to 0.25. After each convolutional layer, a max-pooling layer is used with kernel size 2x2 and stride 2. Three fully-connected layers are applied on the output and the cross-entropy softmax loss is used to train the model. The total number of trained parameters for AlexNet is 26.4M. VGGNet [Simonyan and Zisserman, 2014] is deeper than AlexNet. It consists of sixteen convolutional layers, five max-pooling layers and three fully-connected layers. The training procedure is the same as AlexNet. There are 50.8M parameters in VGGNet. The performance of AlexNet and VGGNet on the two datasets with different epochs is shown in Figure 5.25. The performance looks promising and the precision of metal texture recognition is higher than 95%. The performance of AlexNet and VGGNet has no significant difference, but the number of parameters of AlexNet is much smaller and therefore it is easier to be trained and applied in real-world product lines. The defect textures in metal surface can be detected by a sliding-window strategy. Each patch on the metal image is classified into different defect textures. Once the texture attributes are detected, the related root causes may be found, such as an external (metal) element, which is affecting the surface of the tool. This is an exploratory result and more wear and tear textures can be added if available.

5.5.3.3 Discussion

Hunting for root causes of metal surface defects can be performed by an analysis of the textural attributes of metal surfaces. Textural attributes can be learned from an annotated data set. This experiment gives initial results useful for RCA where damage of metal surfaces is involved. From the initial results of two small data sets, it is clear that textural analysis on metal surface

can provide information for maintenance of product lines with classification of 95%.

Several issues should be solved to apply this method in the real-word:

- Real data set collection and annotation. The challenge of using deep learning in general on metal surfaces concerns the labelled data set. The number of parameters in the network is very large and thus the system provides higher performance when more data is available or the network is pre-trained on other large-scale databases;
- Running the complex deep neural network takes computing time and memory. Networks that are more efficient might be investigated, such as quantizing the parameters of the network;
- Continuous learning is also very important because unseen causes are coming in over time. The system should learn new things when new labels are added and operators may need to perform this additional task before valid results can be obtained.

References

Aghasaryan, A., Fabre, E., Benveniste, A., Boubour, R., and Jard, C. (1997) 'A Petri net approach to fault detection and diagnosis in distributed systems,' in *36th IEEE Conference on Decision and Control, IEEE CDC'97*, San Diego, California, USA.

Agresti, A. (1992) 'A survey of exact inference for contingency tables,' *Statistical Science* 7(1), pp. 131–153.

Ali, J. B., Chebel-Morello, B., Saidi, L., Malinowski, S. and Fnaiech, F. (2015) 'Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network,' *Mechanical Systems and Signal Processing*, 56–57, pp. 150–172.

Aminikhanghahi, S. and Cook, D. J. (2017) 'A survey of methods for time series change point detection,' *Knowledge and information systems* 51.2, 339–367. PMC.

Andrews, R., Diederich, J., and Tickle, A. B. (1995) 'Survey and critique of techniques for extracting rules from trained artificial neural networks,' *Knowledge-Based Systems*, 8(6), pp. 373–389.

Barnard, G. A. (1959) 'Control charts and stochastic processes,' *Journal of the Royal Statistical Society. B (Methodological)* 21(2), pp. 239–271.

Bashivan, P., Rish, I., Yeasin, M., and Codella, N. (2015) 'Learning representations from EEG with deep recurrent-convolutional neural networks,' *International Conference on Learning Representations*.

Batal, I., Valizadegan, H., Cooper, G. F., and Hauskrecht, M. (2013) "A temporal pattern mining approach for classifying electronic health record data." *ACM Trans. Intell. Syst. Technol.*, 4(4), pp. 63:1–63:22.

Baxt, W. G. (1990) 'Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion,' *Neural Computation*, 2, pp. 480–489.

Burnham, K. P. and Anderson, D. R. (2002) *Model Selection and Multimodel Inference: A Practical InformationTheoretic Approach* (2nd ed.). Springer-Verlag.

Cai, M. and Liu, J. (2016) 'Maxout neurons for deep convolutional and LSTM neural networks in speech recognition,' *Speech Communication*, 77, p. 53.

Caruana, A. and Niculescu-Mizil, A. (2005) An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2015) The ucr time series classification archive. www.cs.ucr.edu/eamonn/time series data/.

Chen, Y., Yang, J., Qian, J. (2016) 'Recurrent neural network for facial landmark detection,' *Neurocomputing*.

Chilimbi, T. M., Suzue, Y., Apacible, J. and Kalyanaraman, K. (2014) 'Project Adam: Building an efficient and scalable deep learning training system,' in *11th USENIX Symposium on Operating Systems Design and Implementation*, Broomfield, CO, USA.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014) 'Learning phrase representations using RNN encoder-decoder for statistical,' in *Proceedings of the 2014 Conference on Empirical Methods in Natural*, Doha, Qatar.

Collett, D. (2003) *Modelling Survival Data in Medical Research*, Second Edition, Taylor Francis.

Collobert, R. and Weston, J. (2008) 'A unified architecture for natural language processing: Deep neural networks with multitask learning,' in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland.

Daniel, Wayne W. (1990) 'Kolmogorov–Smirnov onesample test,' *Applied Nonparametric Statistics* (2nd ed.). Boston: PWSKent. pp. 319–330. ISBN 0-534-91976-6.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009) 'ImageNet: A large-scale hierarchical image database,' *IEEE Conference on Computer Vision and Pattern Recognition*.

Domhan, T., Springenberg, J. T. and Hutter, F. (2015) 'Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves,' in *Proceedings of the 24th International Conference on Artificial Intelligence*, Buenos Aires, Argentina.

Ebrahimi, M., Suen, C. Y., and Ormandjieva, O. (2016) 'Detecting predatory conversations in social media by deep convolutional neural networks,' *Digital Investigation*, 18, p. 33.

Egmont-Petersen, M., deRidder, D., and Handels, H. (2002) 'Image processing with neural networks-a review,' *Pattern Recognition*, 35, pp. 2279–2301.

Facchinetti, S. (2009) 'A procedure to find exact critical values Of Kolmogorov-Smirnov test,' *Statistica Applicata – Italian Journal of Applied Statistics* 21(3–4), pp. 337–359.

Fisher, R. A. (1922) 'On the interpretation of $\chi 2$ from contingency tables, and the calculation of P,' *Journal of the Royal Statistical Society*, 85(1), pp. 87–94, doi:10.2307/2340521.

Fisher, R. A. (1954) *Statistical Methods for Research Workers*, Oliver and Boyd. ISBN 0-05-002170-2.

Fisher, L. D. and Lin, D. Y. (1999) 'Time-dependent covariates in the Cox proportional-hazards regression model,' *Annual Review of Public Health*, 20, pp. 145–157.

Fox, J. and Weisberg, S. (2010) Appendix to An R Companion to Applied Regression Appendix to An R Companion to Applied Regression, SAGE Publications.

Friedman, J. (1999) 'Stochastic gradient boosting,' *Computational Statistics and Data Analysis*, 38, pp. 367–378.

Gebraeel, N. Z. (2003) 'Real-time degradation modelling and residual life prediction for component maintenance and replacement,' Ph.D. Dissertation. School of Industrial Engineering.

Gebraeel, N., Lawley, M. A., Li, R., and Ryan, J. (2005) 'Residuallife distributions from component degradation signals: A Bayesian approach,' *IIE Transactions 37*, 37, pp. 543–557.

Gers, F. A., Schmidhuber, J., Cummins, F. (2000) 'Learning to forget: Continual prediction with LSTM,' *Neural Computation*, 12(10).

Guo, C. and Morris, S. A. (2017) 'Engineering cell identity: Establishing new gene regulatory and chromatin landscapes,' *Current opinion in genetics & development*, 46, pp. 50–57.

Gustafsson, F. (2000) *Adaptive Filtering and Change Detection*. Wiley.

Haeusser, P. (2017) 'How computers learn to understand our world,' [Online]. Available: http://www.in.tum.de/forschung/forschung shighlights/how-computers-learn-to-understand-our-world.html.

Hajinoroozi, M., Mao, Z., Jung, T., Lin, C., Huang, Y. (2016) 'EEG-based prediction of driver's cognitive performance by deep convolutional neural network,' *Signal Processing: Image Communication*, 47, p. 549.

Hasan, M., Kotov, A., Carcone, A. I., Dong, M., Naar, S., Hartlieb, K. B. (2016) 'A study of the effectiveness of machine learning methods for classification of clinical interview fragments into a large number of categories,' *Journal of Biomedical Informatics*, 62, p. 21.

Hijazi, S., Kumar, R., and Rowen, C. (2015) *Using Convolutional Neural Networks for Image Recognition*, Cadence.

Horster, H., Kauer, E., and Lechner, W. (1971) 'The Burn-out mechanism of incandescent lapms,' *Philips tech Rev*, 32(6), pp. 155–164.

International Institute of Welding. (2008) Recommendations For Fatigue Design Of Welded Joints And Components.

Ijjina, E. P., Chalavadi, K. M. (2016) 'Human action recognition using genetic algorithms and convolutional neural networks,' *Pattern Recognition*, 59, p. 199.

ISO 13381-1, "Condition monitoring and diagnostics of machines-prognostics-part 1: General guidelines,' *International Standards Organization*.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2014) *An Introduction to Statistical Learning: With Applications in R*, Springer.

Jolliffe, I. T. (2002) *Principal Component Analysis*, 2nd edition, Springer.

Kaushik, M. and Mathur, B. (2014) 'Comparative study of K-means and hierarchical clustering techniques,' *International Journal of Software and Hardware Research in Engineering*, pp. 93–98.

Kingma, D. P. and Jimmy, B. (2014) 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012) 'Imagenet classification with deep convolutional neural networks,' *Advances in neural information processing systems*, pp. 1097–1105.

Larsen, R. J. and Stroup, D. F. (1976) *Statistics in the Real World: A Book of Examples*, Macmillan.

LeCun, Y., Bengio, Y., and Hinton, G. (2015) 'Deep learning,' *Nature*, 521, pp. 436–444.

Lee, H., Largman, Y., Pham, P., and Ng, A. Y. (2009) "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *22nd International Conference on Neural Information Processing Systems (NIPS'09)*, USA.

Levina, E. and Bickel, P. (2001). "The earthmover's distance is the mallows distance: Some insights from statistics,' *Proceedings of ICCV 2001*, Vancouver, Canada, pp. 251–256.

Li, J., Deng, L., Gong, Y., Haeb-Umbach, R. (2016) 'Robust automatic speech recognition – a bridge to practical applications.

Li, X. and Wu, X. (2015) 'Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition,' *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Lipton, Z. C., Kale, D. C., Elkan, C., and Wetzell, R. (2016) 'Learning to diagnose with LSTM recurrent neural networks,' *International Conference on Learning Representations*.

Ljung, L. (1999) *System Identification: Theory for the User*, Prentice Hall, 2 edition.

van der Maaten, L. and Hinton, G. (2008) 'Visualizing high-dimensional data using t-SNE,' *Journal of Machine Learning Research*, 9, pp. 2579–2605.

Maier, H. R. and Dandy, G. C. (2000) 'Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications,' *Environmental Modelling Software*, 15, pp. 101–124.

Mann, H. B. and Whitney, D. R. (1947) 'On a test of whether one of two random variables is stochastically larger than the other,' *Annals of Mathematical Statistics*, 18(1), pp. 50–60.

Mann, P. S. (1995). *Introductory Statistics* (2nd ed.). Wiley. ISBN 0-471-31009-3.

Mittelhammer, R. C., Judge, G. G., Miller, J., and Douglas, J. (2000). *Econometric Foundations*, Cambridge University Press. pp. 73–74. ISBN 0-521-62394-4.

Moghaddam, A. H., Moghaddam, M. H., and Esfandyari, M. (2016) 'Stock market index prediction using artificial neural network,' *Journal of Economics, Finance and Administrative Science*, 21, pp. 89-93.

Nielsen, M. (2017) 'Using neural nets to recognize handwritten digits,' Available: http://neuralnetworksand deeplearning.com/chap1.html.

Parzen, E. (1962) 'On estimation of a probability density function and mode,' *The Annals of Mathematical Statistics* 33(3), p. 1065.

Pearson, K. (1901) "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, 2(11), pp. 559–572. doi:10.1080/14786440109462720.

Refenes, A.-P. N., Zapranis, A., and Francis, G. (1994) 'Stock performance modeling using neural networks: A comparative study with regression models,' *Neural Networks*, 7, pp. 375–388.

Rice, R. (1997) *SAE Fatigue Design Handbook*, Warrendale, Penn.: Society of Automotive Engineers.

Rifai, S. and Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011) 'Contractive auto-encoders: explicit invariance during feature extraction', *Proceedings of the 28th International Conference on Machine Learning*.

Rosenblatt, M. (1956) 'Remarks on some nonparametric estimates of a density function,' *The Annals of Mathematical Statistics*, 27(3), p. 832.

Schmidhuber, J. (2015) 'Deep learning in neural networks: An overview,' *Neural Networks*, 61, pp. 85–117.

Shapiro, S. S. and Wilk, M. B. (1965) 'An analysis of variance test for normality (complete samples),' *Biometrika*, 52(3–4), pp. 591–611.

Si, X.-S., Wang, W., Hua, C.-H., and Zhou, D.-H. (2011) 'Remaining useful life estimation – A review on the statistical data driven approaches,' *European Journal of Operational Research*, 213, pp. 1–14.

Sikorska, J., Hodkiewicz, M., and Ma, L. (2011) 'Prognostic modelling options for remaining useful life estimation by industry,' *Mechanical Systems and Signal Processing* 25(5), pp. 1803–1836.

Simonyan, K. and Zisserman, A. (2014) 'Very deep convolutional networks for large-scale image recognition,' *arXiv preprint arXiv:1409.1556*.

Simpson, J. W. and Sheppard, W. (1998) 'Inducing diagnostic inference models from case data,' in *Frontiers in Electronic Testing*, vol. 13, Kluwer.

Stuart, A., Ord, K., and Arnold, S. (1999) Kendall's Advanced Theory of Statistics: Volume 2A—Classical Inference & the Linear Mode.

Tamilselvan, P. and Wang, P. (2013) 'Failure diagnosis using deep belief learning based health state classification,' *Reliability Engineering System Safety*, 115, pp. 124–135.

Team, R. C. (2014) 'R: A language and environment for statistical computing,' *R Foundation for Statistical Computing*, Vienna, Austria.

Therneau, T. M. (2015) 'A Package for Survival Analysis in S.'

Therneau, T. C. C. and Atkinson, E. (2017) 'Using time dependent covariates and time dependent coefficients in the cox model,' *Survival Vignettes*.

Thomas, L. and Reyes, E. M. (2014) 'Tutorial: Survival estimation for cox regression models with time-varying coefficients using SAS and R,' *Journal of Statistical Software*, 61.

Tran, T. D., Khoa Nguyen, D., Tran, T. A. X., Nguyen, Q. C., and Nguyen, H. B. (2011) 'Speech enhancement using combination of dereverberation and noise reduction for robust speech recognition,' in *Proceedings of the Second Symposium on Information and Communication Technology*, Hanoi, Viet Nam.

Tu, J. V. (1996) 'Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes,' *Journal of Clinical Epidemiology*, 49, pp. 1225–1231.

Tukey, J. (1949) 'Comparing individual means in the analysis of variance,' *Biometrics* 5(2), pp. 99–114.

Vapnik, V. N. (1995) *The Nature of Statistical Learning Theory*, Springer-Verlag.

Vapnik, V. N. and Chervonenkis, A. Y. (1971) "On the uniform convergence of relative frequencies of events to their probabilities,' *Theory of Probability & Its Applications*, 16(2), p. 264.

Wang, Z. and Oates, T. (2015) 'Spatially encoding temporal correlations to classify temporal data using convolutional neural networks,' *CoRR*.

Webers, G., Boosten, M., and Sprong, H. (2016) 'Mantis D3.1 Report on sensors selected and to develop - Appendix 9,' *ECSEL*.

Welte, T. M. and Wang, K. (2014) 'Models for lifetime estimation: An overview with focus on applications to wind turbines," *Advances in Manufacturing* 2(1), pp. 79–87.

Widrow, B., Rumelhart, D. E., and Lehr, M. A. 'Neural networks: Applications in industry, business and science,' *Commun. ACM*, 37, pp. 93–105.

Xu, X., Yan, Z., and Xu, S. (2015). 'Estimating wind speed probability distribution by diffusion-based kernel density method,' *Electric Power Systems Research*, 121, pp. 28–37.

Yuea, W., Xiaojiea, W., and Yuzhao, M. (2016) 'First-feed LSTM model for video description,' *The Journal of China Universities of Posts and Telecommunications*, 23(3), p. 89.

Zachary, J. B., Lipton, C., and Elkan, C. (2015) *A Critical Review of Recurrent Neural Networks for Sequence Learning*, CoRR.