Predicting harmful conditions with Hidden Markov Models

Vorhersage von Schadzuständen durch Nutzung von Hidden Markov Modellen
Master-Thesis von Zahra Fardhosseini
Tag der Einreichung: 15. Mai 2017

1. Gutachten: Prof. Johannes Fürnkranz
2. Gutachten: Sebastian Kauschke, M.Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department
Knowledge Engineering Group

Predicting harmful conditions with Hidden Markov Models
Vorhersage von Schadzuständen durch Nutzung von Hidden Markov Modellen

Vorgelegte Master-Thesis von Zahra Fardhosseini

1. Gutachten: Prof. Johannes Fürnkranz
2. Gutachten: Sebastian Kuschke, M.Sc.

Tag der Einreichung: 15. Mai 2017

Ehrenwörtliche Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. Mai 2017

Zahra Fardhosseini

Abstract

Predictive Maintenance is a technique used to predict the condition of in-service equipment for adapting a maintenance schedule. Techlok is a project of DB Cargo, benefiting from Predictive Maintenance in order to increase the trains availability and cost reduction. The trains are equipped with sensors to produce continuous log file of diagnostic data. Based on these diagnostic data, a scenario is employed to create a predictor of failure trains. Hidden Markov Model (HMM) is a Machine Learning algorithm rest on Markov Model with hidden states. The model is applied in different fields such as speech recognition and gen techniques. In this work dealing with lots of diagnostic data, Hidden Markov Model is used in order to develop a failure predictor model. The results show that using HMM to predict the failure with respect of the incoming system data is possible. The model classifies with an accuracy of 96%. The classifier can predict well, but it is strongly dependent to the data stabilities. Although here a subset of the whole diagnostic codes is examined, the results encourage generalizing the model forward all features. More research and development is called for.

1. Contents

List of Figures

List of Tables

2. Introduction

Predictive Maintenance is a technique to predict the conditions of the equipment [1]. By applying this technique, it can be predicted, when a failure of in-service equipment is happening. The aim is to predict the failures at early stages to be able to avoid them. In this way, a maintenance schedule could be created. This results in cost savings and a reduction of maintenance time. Performing maintenance on systems is required to avoid the unplanned breakdowns. Consequently, the time the system is out of order would be minimized. This results in increasing of production time. The advantage would be, that the system is more reliable. In big machinery scenarios, the equipment and their in-use time are very costly. Being able to predict the failure and efficient maintenance planning is highly relevant.

DB. Cargo is one of the greatest European railway cargo carriers. The company has 2869 locomotives beside 87 264 boxcars [2]. In such large companies, the effort is to have an overview of the in-service systems. By unforeseeable breakdown of locomotives or boxcars, the company makes great losses. To avoid unpredicted breakdowns, the systems are ordinary send to workshops to be checked. However, this program could not prevent the sudden breakdowns. An alternative solution is to employ Predictive Maintenance to benefit from the advantages. Therefore, the company started at 2011 a predictive maintenance project, called Techlok. For this purpose, the systems (locomotives boxcars) were equipped with sensors. The sensor's data were saved as log data from the components of the systems. The diagnostic data were collected in form of log data by monitoring the systems. These data were used to be applied in the Predictive Maintenance. The aim in this work is to design a failure prediction model based on the log data. The Knowledge Engineering group of Technische Universität Darmstadt cooperates with this company to design the failure prediction model. This thesis is a part of this cooperation. This thesis focuses on development and evaluating a Hidden Markov Model to predict the power converter failures with support from diagnostic data. The power converter converts high-voltage to power lines. This is a component, which is responsible for the electrical supply of the drive motors.

2.1. Goal of this Work

In this thesis, Hidden Markov Model is applied to predict the system failures based on diagnostic codes which are derived from log data. The model is trained and evaluated on a subset of real-world log data from DB. Cargo's systems. To achieve this, the following challenges are addressed:

- Determining a proper feature selection and model configuration.
- Learn suitable parameters to adapt it to real-world situations.

- Find out the optimal training data for model.
- Developing useful evaluation method to estimate the performance.
- Analyze the influence of scaling factors and training data on performance.

2.2. Structure

This thesis is structured as follows. First, an introduction to Machine Learning and its theoretical background and applications is given. Second, the utilizing models and methods are introduced. Third in Chapter 4, the related works based on two parts of this project are mentioned. Later in chapter 5, the procedure of designing the datasets and preprocessing steps are described. Afterwards Implementing algorithms and issues are depicted in detail in chapter 6. In addition, some faced difficulties, and the applied solution are presented. Furthermore, the final results and evaluation are mentioned. Finally, the work is completed with conclusion remarks.

3. Theoretical Background

This work is firmly based on the application of Machine Learning algorithms it utilizes. Therefore, an introduction to the models and learning methods is given in this chapter. First, an overview on general Machine Learning methods are provided, supervised as well as unsupervised. Further evaluation aspects are elaborated, followed by describing the specific aspects of Hidden Markov Model.

3.1. Machine Learning

The field of Machine Learning is closely related to Data Mining and Statistics. The goal is to develop means for discovering meaningful patterns in datasets. In statistics, this is known as Pattern Recognition. The goal of all three is to predict data, but through different approaches. Pattern recognitions come from engineering, where the statistic part is about numbers and quantifications. Mathematical tools and computation are applied in Pattern Recognition. In contrary, Machine Learning is part of computer science, which improves its performance with experiences. That means the performance improves with experience according to some performance measures. The goal is to find, learn, and recognize patterns in datasets, with support from massive Pattern Recognition methods and Statistics. The procedure of Data Mining applies Machine Learning algorithm to find appropriate patterns in large databases. In other words, within Data Mining, data sets are converted to knowledge using programing methods and algorithms. In this work, the major related parts are explained.

Databases are created to store the data, but not to analyze them. Being able to analyze the data in Data Mining, a knowledge discovery cycle is created [3]. Basically, it is a process to transform row data to useful knowledge. These steps are depicted in Figure 3-1.

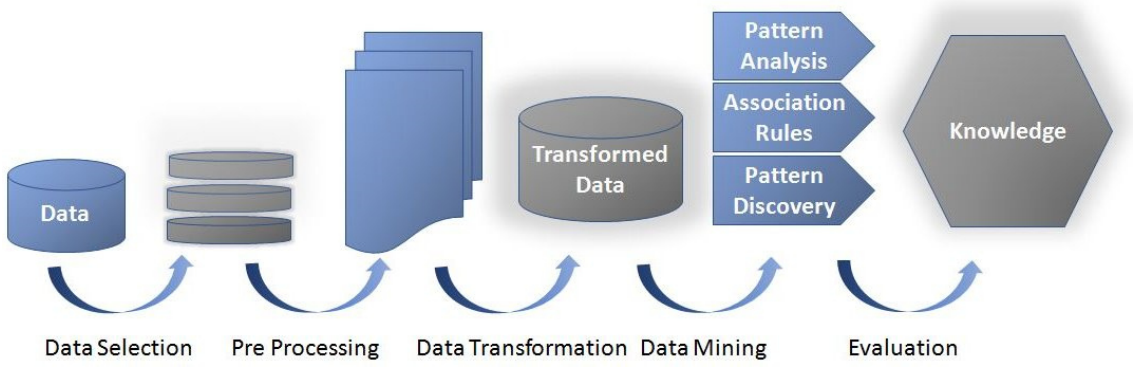

Figure 3-1: Knowledge discovery cycle [1]

In the figure, the steps required to produce knowledge from data are shown. First in data selection, the relevant data parts are chosen. In the next step, the data should be cleaned from noises and inconsistence. Now, the clear relevant data should be analyzed to export features from them. The features can be just one or a set of different feature set. Features must adapt the destination format. These steps are also named Feature Engineering. In addition, data must be allocated to tables and aggregation. This process is done in Data Transformation. Machine Learning algorithms and knowledge discovery techniques are used to create and find useful patterns in Data Mining part. Finally, the detected patterns can be evaluated. The evaluation is based on whether the output data is interesting and useful or not. Thus, the output is the knowledge which should be properly visualized.

Following, some major basics of Machine learning and more specific parts of Hidden Markov Model are described.

Supervised vs. Unsupervised

The observed data sets could be defined without arranging them in groups [4]. Defining group for each data is defined as labeling. Setting a label for data means allocate it to a class. In Machine Learning models, the labels are important. These labels determine the type of learning models. The learning types are divided into four subcategories:

- Supervised: The labels are included in data sets. So, the data belongs to specific classes. These data are mostly used for training. Thus, classification and regression algorithms are suggested.
- Unsupervised: The labels are not clarified in data sets. Here, the Machine Learning algorithms which subgroup the data in clusters are suggested. Each cluster defines a label.
- Reinforcement Learning: The learning procedure is done by evaluation function through feedback from environment.
- Semi-Supervised Learning: In this model, only some of the instances are labeled.

A learning process could be like the process depicted in Figure 3-2 [5]. A criterion in Machine Learning is to predict the labels. Based on the labels format and the learning types, classification, regression, or clustering can be used.

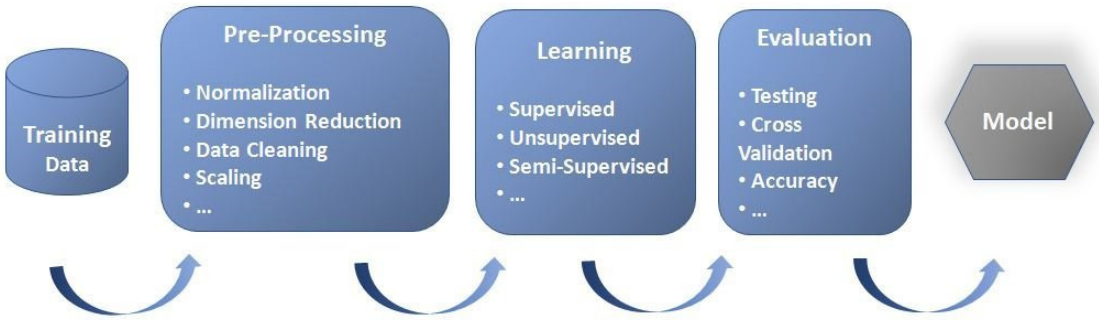

Figure 3-2: Machine Learning process [1]

Classification, Regression, and Clustering

Classification or Regression are methods for supervised learning. The goal is to aggregate the same labels in classes. Choosing an appropriate method between Classification and Regression depends on the label type. Label type can be numeric or non-numeric data. For example, having Yes and No as labels, the algorithm is a binary classification with Yes and No as labels. Numeric labels should be regressed and non-numeric should be classified into number of discreet labels. However, clustering uses the similarity basics, which means the affine instances are grouped together. The goal is to cluster the examples in same groups, so that both the intra-class similarity on data and inter-class dissimilarity on classes are increased. An example for unsupervised learning is K-means.

K-means Algorithm

K-means is a clustering algorithm, which learns from Instances [6]. The data is numeric domain and the procedure is like K-nearest neighbors. Centroids represent the clusters. The process starts by setting *K* as the number of clusters. These points are initially set as centroid. Each data is allocated to the closest centroid. Then, the centroids are computed again. These steps are repeated till the centroids stay the same and the clusters stays stable. K-means Algorithm is simple and efficient, but in some cases, it won't give the best clusters.

3.1.1. Maximum Likelihood

Maximum Likelihood is a statistical method. Here, the aim is to find parameters, which maximize the likelihood function, by given observations [7]. The maximum parameters describe the datasets as

best. This method is called Maximum Likelihood Estimation (MLE). In other words, the best parameters are to be found to match the data to the function. For random variables, MLE maximize the probability of observed data to adjust the distribution function. Assume X has n independent identical distribution (i.i.d data) random observed data $X= \{x_1, x_2, x_3..., x_n\}$, where it comes from a density function $p(x|\theta)$ and θ is a set of parameters. MLE shows how possible it is to observe x_i in the density function by setting θ as parameters. So, the Likelihood would be as shown in Equation(3-1):

$$\mathcal{L}(\theta; x_1, x_2, x_3, \dots, x_n) = p(x_1, x_2, x_3, \dots, x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \qquad (3\text{-}1)$$

Since working with sums are easier than products, the logarithmic form of Likelihood is computed. So, the projection is converted to a sum, as shown in Equation (3-2):

$$\ln \mathcal{L}(\theta; x_1, x_2, x_3, \dots, x_n) = \sum_{i=1}^{n} \ln p(x_i|\theta) \qquad (3\text{-}2)$$

Here the idea is to set the parameters so that the likelihood gets maximized. The best parameter set θ^* should match data to density function, see Equation(3-3).

$$\theta^* = \arg max_{\theta}\, \mathcal{L}(\theta|X) \qquad (3\text{-}3)$$

The equation above is an optimization problem. Optimization problem is the process to find the optimal solution from all feasible solutions. Maximum Likelihood Estimation has convergence properties by increasing the size, even though they are one of the simplest alternative techniques.

3.1.2. Bayes Theory

Bayes theory is also a statistic method. It has alternative names, also known as Bayes rule, conditional probability or inverse probability [8]. The theory is used to find the conditional probability. Bayes theory is a method to find the probability when other certain known probabilities exist. In other words, the probability of an event given that another event happens. Assuming A and B are events, where $p(A)$ and $p(B)$ are referred as prior probabilities, the conditional probability is shown in (3-4).

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} \qquad (3\text{-}4)$$

$p(A \mid B)$ is the conditional distribution also named as posterior probability. $p(B \mid A)$ is the probability of B given A. $p(B)$ is the normalizing constant. This ensures that there is a probability function. Using this method, the probability of event with prior knowledge of conditions is calculated. These conditions might be relevant or not. The conditions are added to increase the accuracy

3.1.3. Expectation Maximization

Expectation Maximization (EM) is a generally iterative statistic method [9], which optimize the Maximum Likelihood estimating parameters in hidden states or hidden variables. In each iterative step, the likelihood of given data is estimated. Both parameters and hidden variables could be unknown in EM.

This model consists of two processing steps:

- Expectation step (E-step): Expectation value of log-likelihoods depending on the current parameters and the given data.
- Maximization step (M-step): Calculation of the parameters, which locally optimize the function resulting from the E-step.

For next iteration, E-step utilizes the latent variable distribution determined by estimated parameters. This algorithm will find a local optimum, which is not a global solution for the problem. The MLE for a statistical model, which generates an observation data set X, unobserved dataset Z and θ as the vector of unknown parameters, with likelihood function $L(\theta; X, Z) = p(X, Z \mid \theta)$ is computed, see Equation (3-5):

$$L(\theta; X) = p(X \mid \theta) = \sum_{Z} p(X, Z \mid \theta) \qquad (3\text{-}5)$$

In further, the EM tries to find the MLE of the marginal likelihood using E-step and M-step iteratively. E-step is defined in Equation (3-6) with respect of conditional distribution of Z given X and the previously determined θ parameters.

$$Q(\theta, \theta^{\text{old}}) = E_{Z|X,\theta^{\text{old}}}[\ln L(\theta; X, Z)] \quad (3\text{-}6)$$

M-step tries to find the optimal parameter which maximizes expectation shown in Equation (3-6). This is given in Equation (3-7):

$$\theta = \arg\max_{\theta} \sum_{J \varepsilon J^n} Q(\theta, \theta^{\text{old}}) \quad (3\text{-}7)$$

EM is widely used for learning the presence of unobserved variables like class labels and features. Therefore, it is used as the basis of many unsupervised clustering algorithms. This method is also used in Baum Welch algorithm and Forward Backward algorithm in Hidden Markov Model (more detail in the section 3.2.5).

3.1.4. Multivariate Gaussian Distribution

Multivariate Gaussian distribution [10] is an extension of one-dimension Gaussian distribution. Multivariate Gaussian distribution is defined as a *k*-variate normal distribution, if any linear combination of *k* component has a univariate normal distribution. It refers to the joint distribution of at least two variables. When this joint distribution is Gaussian, the parameters are set as a vector of means for each variable and a variance-covariance matrix. The matrix contains the variance of each single variable and the covariances between pairs of variables. Equation (3-8) is a n-dimensional Gaussian distribution for a continuous variable *x* with *n* values. It is specified with μ as mean vector in length *n* and Σ as covariance $n \times n$ matrices. In this equation $|\Sigma|$ is the matrix determination.

$$p(x|\mu, \Sigma) = \frac{1}{2\pi^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\} \quad (3\text{-}8)$$

For Maximum likelihood, the equations are like univariate state with the difference that the mean vector is the empirical average of the data vector *x*. This is shown in Equation (3-9):

$$\hat{\mu} = \frac{1}{m}\sum_{j} x^{(j)} \quad (3\text{-}9)$$

The covariance matrix will also be the empirical average of $n \times n$ matrices, see Equation (3-10):

$$\hat{\Sigma} = \frac{1}{m}\sum_{j}(x^{(j)} - \hat{\mu})^T(x^{(j)} - \hat{\mu}) \qquad (3\text{-}10)$$

3.1.5. Evaluation

To be aware of productivity and to check the quality of the model, it should be evaluated. Evaluation helps to find the best model representing our data and how well the chosen model will work in the future. Evaluation techniques are about validation the model and score it. In validation, the data are divided in subgroups and performing the model on test data. The model does not see test data to evaluate their performance. An example of this is Cross-validation, which is based on dividing the limited labeled data into equal sized folds. After validation, a quantitative method must be used to evaluate the model. The confusion matrix is a good way to measure whether the model assigns the correct class value to the test instances. The basic parameters are true positive, false positive, true negative and false negative. These parameters are demonstrated in confusion matrix shown in Table 3-1.

Table 3-1: Confusion matrix, binary classification

Confusion Matrix	Classified as +	Classified as -	
Class +	True Positive (tp)	False Negative (fn)	tp + fn = P
Class -	False Positive (fp)	True Negative (tn)	fp + fn = N
	tp + fp	tn + fn	|E| = P + N

Some common evaluation techniques are listed below:

- True positive rate: The proportion of positives that are correctly classified.
- True negative rate: The proportion of negative that are correctly classified.
- False positive rate: The proportion of positives that are incorrectly classified.
- False negative rate: The proportion of negative that are incorrectly classified.
- Accuracy: The percentage of correct classified, see Equation (3-11):

$$acc = \frac{tp + tn}{P + N} \qquad (3\text{-}11)$$

- Precision: The number of true positive of all positive classified data. In other words, the count of relevant positive classified data.

$$precision = \frac{tp}{tp + fp} \quad (3\text{-}12)$$

- Recall: The number of true positive data of all positive data. It can also be described as the count of selected relevant positive data.

$$recall = \frac{tp}{tp + fn} \quad (3\text{-}13)$$

- F-measure: The precision or recall is separately not so useful. Dependent to the problem definition they can be used separately. The relation between them is defined by F-measure or F1 score, see Equation(3-14):

$$F1 = 2\,\frac{precision\,.recall}{precision + recall} \quad (3\text{-}14)$$

- Mean Square Error (MSE): Is a risk function which computes the squared mean of the difference between actual value and the estimated value. The Equation is seen in (3-15), with respect of Forecasted value F_t at time *t* and Actual value *A* at time *t.*

$$mse = \frac{1}{n}\sum_{t=0}^{n}(F_t - A_t)^2 \quad (3\text{-}15)$$

- Mean Absolute Percentage Error (MAPE): The percentage of accuracy of the forecasted data to the actual data is computed. MAPE is defined by the sum of the difference from actual data and forecasted data divided by the number of fitted points.

$$mape = \frac{100}{n}\sum_{t=0}^{n}\frac{A_t - F_t}{A_t} \quad (3\text{-}16)$$

The most common evaluation techniques in Predictive Maintenance are accuracy, precision, mean square error and mean absolute percentage error [11]

Cross validation

Cross validation is an evaluation technique used to improve the performance. This model is used when training data is not enough. The process is an iterative process on training data. The training data

is divided into equal size *n* folds, where *n* is defined according to the use case. At each iteration, the model learns of (*n*-1) folds and evaluates on the last fold. At the end, the mean value of all evaluations is taken as model performance [4]. For example, by selecting a 10-fold cross validation, the data is

Figure 3-3: Cross validation illustration

divided into 10 equal size partitions. In every iteration, one partition is used for testing and all others for learning. The procedure will be done 10 times, till each partition has been once set for the test. An illustration of Cross validation is shown in Figure 3-3.

3.2. Hidden Markov Model

For explaining the Hidden Markov Model, a simple example is described at first. The goal is to predict tomorrow's weather with the support of observing the weather from the previous day. The weather types are sunny and rainy. These are defined as states which can be observed {sunny, rainy}. For more simplicity, it is assumed that the weather does not change in the middle of the day. An Observation could be O = {sunny, rainy, sunny, rainy, sunny, sunny}

Another assumption is that tomorrow's weather is just dependent on today's weather. So, if today is a sunny day, tomorrow's weather will be predicted as $p(w_t|w_{t-1} = \text{sunny})$. This is the Markov chain property. This is nothing else than computing the probability of weather with support of previous weather observations. Markov model takes this probability and creates a probability state automation. In the weather example, there exist 2 states and the probability to transit from a state to another state is written on the axes. An illustration of the model is shown in Figure 3-4. Here the transition

probabilities are shown. The probability of observing a sunny day tomorrow, if today is sunny, would be 0.7.

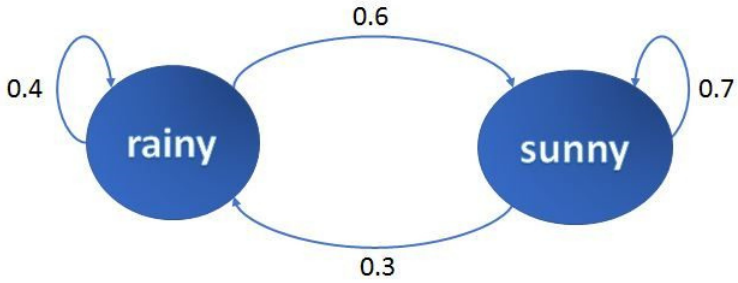

Figure 3-4: Markov Model automation

The weather from the next days could be computed by knowing the transition probabilities. This would be the joint probability of previous day. So, the following probabilities can be set:

$$p(\text{sunny}|\text{rainy}) = 0.3$$

$$p(\text{sunny}|\text{sunny}) = 0.7$$

$$p(\text{rainy}|\text{sunny}) = 0.6$$

$$p(\text{rainy}|\text{rainy}) = 0.4$$

It is assumed that the states have an initial probability like $p(\text{sunny}) = 0.6$ and $p(\text{rainy}) = 0.4$. That means with which likelihood the observation sequence starts with a sunny day. Now the likelihood of observing $O_1 = \{\text{sunny}, \text{sunny}, \text{rainy}, \text{rainy}\}$, an observation sequence can be computed as:

$$p(\text{sunny}, \text{sunny}, \text{rainy}, \text{rainy}) = p(\text{sunny}) * p(\text{sunny}|\text{sunny})$$

$$* p(\text{sunny}|\text{rainy}) * \ p(\text{rainy}|\text{rainy}) = 0.6*0.7*0.6*0.4$$

Hidden Markov Model is a Markov model where the states are hidden. Assume in the example the weather is observed directly. But the model wants the temperature feeling as high or low. An illustration is depicted in Figure 3-5.

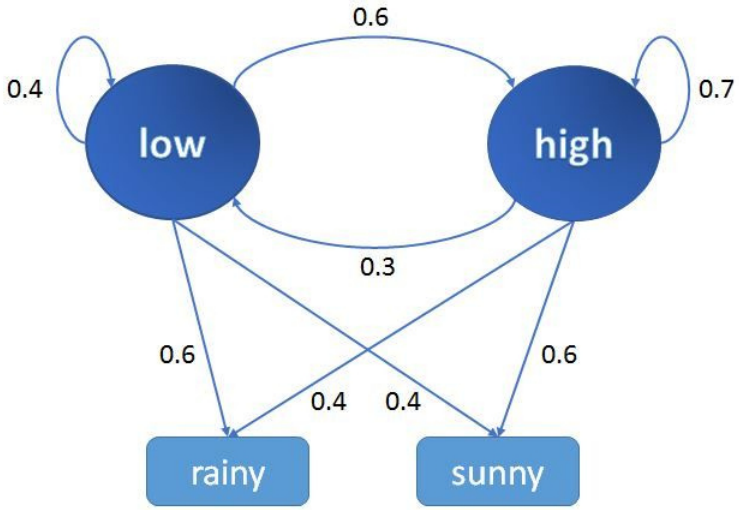

Figure 3-5: Hidden Markov Model, weather example

Here we have two hidden states with low and high. The observation types are sunny and rainy. Like Markov model, the observation sequence could be a subset of any combination of observations. The transition probabilities are as followed:

$$p(\text{low}|\text{high}) = 0.6$$

$$p(\text{high}|\text{high}) = 0.7$$

$$p(\text{high}|\text{low}) = 0.3$$

$$p(\text{low}|\text{low}) = 0.4$$

Another probability in HMM is the emission probability. Emission probability is the likelihood of observation in a state. That means any observation must assign to a state. In another world by a low temperature what would be the probability of observing a sunny day. Here these probabilities are set as follows:

$$p(\text{sunny}|\text{high}) = 0.6$$

$$p(\text{rainy}|\text{high}) = 0.4$$

$$p(\text{rainy}|\text{low}) = 0.6$$

$$p(\text{sunny}|\text{low}) = 0.4$$

The initial probabilities are set to $p(\text{low}) = 0.4$ and $p(\text{high}) = 0.6$. The same as in Markov model. For computing the probability of observation sequence $O_2 = \{\text{sunny}, \text{rainy}\}$ the states probabilities must be included. Since the relevant state is not defined, every possibility has to be checked. The calculation is like followed:

$$p(\text{sunny}, \text{rainy}) = p(\{\text{sunny}, \text{rainy}\}, \{\text{low}, \text{low}\}) + p(\{\text{sunny}, \text{rainy}\}, \{\text{low}, \text{high}\}) +$$

$$p(\{\text{sunny}, \text{rainy}\}, \{\text{high}, \text{low}\}) + p(\{\text{sunny}, \text{rainy}\}, \{\text{high}, \text{high}\})$$

That means besides computing the probabilities of observation, the likelihood of the state must be also computed. This is calculated like below:

$$p(\{\text{sunny}, \text{rainy}\}, \{\text{low}, \text{low}\}) = p(\{\text{sunny}, \text{rainy}\} \mid \{\text{low}, \text{low}\})\, p(\text{low}|\text{low}) =$$

$$p(\text{sunny}|\text{low})\, p(\text{rainy}|\text{low}) p(\text{low}) p(\text{low}|\text{low}) = 0.4 * 0.6 * 0.4 * 0.4$$

The 3-main problems that can be solved with HMM are Evaluation, Decoding and Learning. The probability of generating of O_2 by the Model is computed in Evaluation. A list of weather observations shows how the weather has changed in earlier days. To find out a list of temperatures from earlier days, Decoding must be used. Here with the support of the weather list and the parameters, these can be found out. To be able to predict, first the model must learn the parameters. For learning, the weather observation list must be given as training data. The learning process computes the initial probability, emission and transition probabilities.

Following are the description of Hidden Markov Model and other methods needed in this work.

3.2.1. Time Series Data

As the name is explaining, time series data are data which comes in order of time, mostly by monitoring the system. Time series data enables to distinguish between activities and events happened [12]. These data are sequential with high-dimensionality, which could be discreet or continuous. The speed of generating them and their volume are extraordinary. This data model is used in many fields such as finance, medical, oil and gas industry [13]. The possible internal hidden structure of data is considered by time series methods [14]. To figure out the internal structure of time series data, various classification and clustering techniques are developed such as ARIMA and Linear Regression [12].

The goal is to find methods which can extract the hidden internal structures from time series data. By applying Hidden Markov Model on time series data, the accuracy of the data is improved. The model tries to predict how the system, which produces time series data, works.

3.2.2. Markov Model

Markov model is a stochastic model which consists of states. It can transit among these states with a specific probability. The probability of being in state at the time=0 is defined as the initial probability. The main property in Markov Model is that the next state is just dependent on the current state, not to the previous ones. In other words, the model is memoryless. The simplest Markov model is Markov Chain. Markov Chain is a statistical model, which moves sequentially through states in time. It is used when an uncontrolled system observes the whole states. Another type of Markov Model is the Hidden Markov Model. This is based on Markov Chain, but the difference is that in Hidden Markov Model the autonomous system observes the states partially. This is described more precisely in the next section. There are also other Markov types such as Markov Decision Process or partially observable Markov Decision Process, which are not relevant to this work. An illustration of this model was shown in Figure 3-4.

3.2.3. Hidden Markov Model Definition

Extending the Markov Model, each observation is a probabilistic function of states. The resulting model is Hidden Markov Model (HMM). HMM is a stochastic model of temporal and sequential time series data. The non-observable states correspond to an observation. Therefore, it is defined as hidden states. Hidden states can only be observed by another stochastic process, which produces the observations. HMM characterizes just the statistical properties to Signal Model. Signal Model is the ability to find output characters as signals, such as speech samples, temperature measurements, and physical alarms. The model is rich at mathematical structures. A mathematical structure is a set (or several sets). In this set, various mathematical objects are associated. These objects could be subsets, set of subsets, operations, relations etc. Therefore, it can be used in many applications [15], such as speech recognition, bioinformatics, handwriting recognitions and gen techniques. HMM is used to model large quantity of data like weather or finance data. As it is shown in Figure 3-6, the states are connected to each observation. This defines that any state is accessible from each observation.

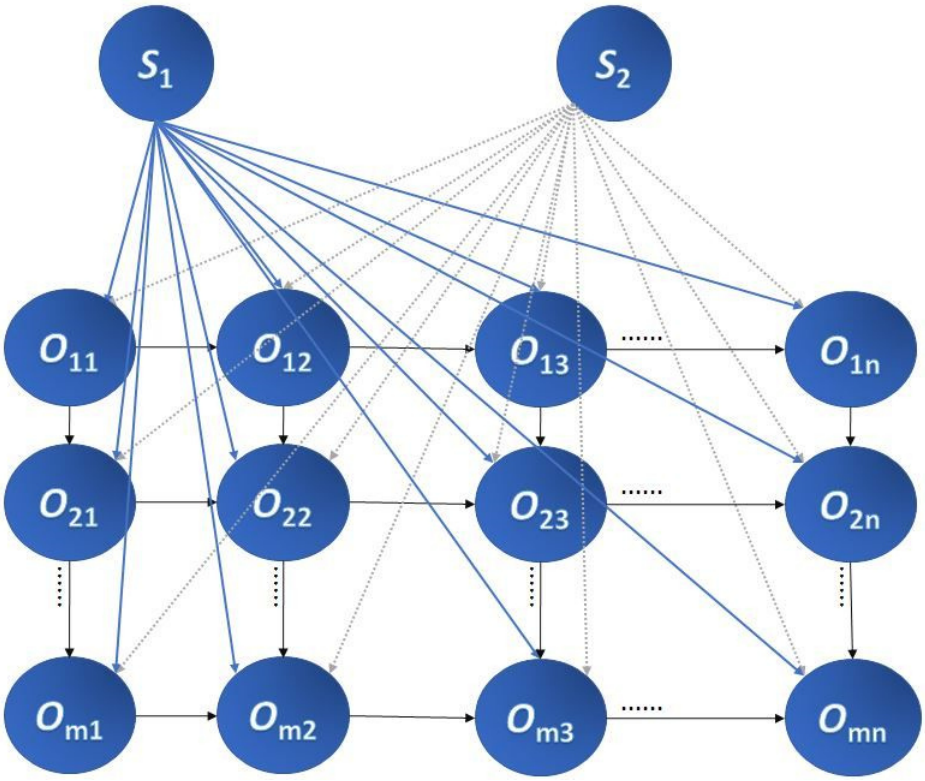

Figure 3-6: Markov Model: States = {S1, S2}, observations = {O1, O2, O3}

HMM use time series data. These data develop a state model, which is dependent only on the previous state. These observations vary in the dimensions and lengths. Assume there is N distinct states $S_1, S_2, S_3, \ldots, S_N$ based on some probabilistic at time t at S_i . This is illustrated in Equation (3-17):

$$P[q_t = S_i \,|q_{t-1} = S_i, q_{t-2} = S_k, \ldots] = P[q_t = S_j \,|q_{t-1} = S_i] \quad (3\text{-}17)$$

Applying perfectly the model has many advantages. But the difficulties by HMM is how to apply the model perfectly. To do this, Maximum likelihood is utilized. So, the optimal parameters should be set. By setting the parameters ideally, the model's likelihood is increased and more proper outputs are achieved. Defining the optimal parameters is in more complex models very difficult. A well-trained HMM should be able to generate observation with the support of the hidden states and the parameters. A sequence of observation is observed. The problem is how to figure a sequence of states, which correspond to these observations. For this, we should find the number of hidden states and the interpretation of them. Theoretically, as the number of states is increased, the degree of freedom is also

increased. Then, the HMM would be larger and could be more capable of modeling. But in practice, the amount of hidden state have limitations [15].

HMM mostly predicts sequences as in speech recognition or text or in DNA for classifications. Therefore, it can be used to predict labels (binary or others) of tokens in sequences. On the other hand, by longer sequences, the probabilities to predict the whole sequence of observations are close to zero. When the sequence length is increased, smaller probabilities of the sequence are achieved. This result is a mathematical limitation. For this reason, the output of HMM could be adapt in other classifiers to evaluate.

HMM also calculates the probability of a specific sequence of observations. In other words, what is the probability of the observed sequence. For example, assume O is the observed sequence as $O = \{S_1, S_1, S_4, S_3\}$ correspond to time=1,2,3,4. The model computes the probability of observing O [5].

In this work, the ergodic or fully connected HMM is applied. Ergodic HMM specifies that every single state can be reached with a single step to other states. This is because of the verity of used states and there is no order of the states. Thus, the ergodic form of HMM came into play.

3.2.4. HMM Parameters

In this section, the Hidden Markov Model parameters are introduced. The main parameters are listed as below:

- N: The number of states. The amount of hidden states is not always easy to determine. By considering the physical characteristics of the model, this can be set individually. An alternative would be by utilizing BIC or AIC described in the section 3.2.6. This set is shown as $S = \{S_1, S_2, S_3, \dots, S_N\}$ and at time t the state is at q_t.
- M: The number of distinct observations in each state. This is an output format of the physical modeling that it has done. The symbols are denoted as a set $V = \{ v_1, v_2, v_3, \dots, v_M\}$.
- **Transition Matrix:** The probability of transition from state i to state j. Transition probabilities are demonstrated in matrix, see Equation (3-18):

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \qquad (3\text{-}18)$$

Each transition probability is calculated with the probabilities of getting from one state to another, see Equation (3-19).

$$a_{ij} = P\left[q_t = a\, S_j \,\middle|\, q_{t-1} = S_i\right], \quad 1 \le i, j \le N \qquad (3\text{-}19)$$

For a special case that each step is accessible from other steps the following conditions should be qualified.

$$a_{ij} \ge 0 \qquad (3\text{-}20)$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad (3\text{-}21)$$

If the state is unreached, then it should be set to zero, see Equation (3-22):

$$a_{ij} = 0 \qquad (3\text{-}22)$$

- **Emission Probability:** This is the probability of the observation symbol at state *j*, shown in Equation (3-23). The emission probability could be a number or a distribution like Gaussian. Here the mean and variance are computed. This results in definition of the emission probability as vectors.

$$B = \{b_j(k)\}, b_j(k) = P[v_k \; at \; t \,|q_t = \; S_j], \quad 1 \le j \le N, 1 \le k \le M \qquad (3\text{-}23)$$

- **Initial State Probability:** The probability at state *i* in time zero *t* = 0, see Equation (3-24):

$$\pi_i = P[q_0 = \; S_i], \qquad 1 \le i \le N \qquad (3\text{-}24)$$

By setting the *N, M, A, B* and π, the HMM can explain the observed sequence. For more simplification, the parameter set is shown in Equation (3-25):

$$\lambda = (A, B, \pi) \qquad (3\text{-}25)$$

The transition and emissions probabilities are visualized in Figure 3-7.

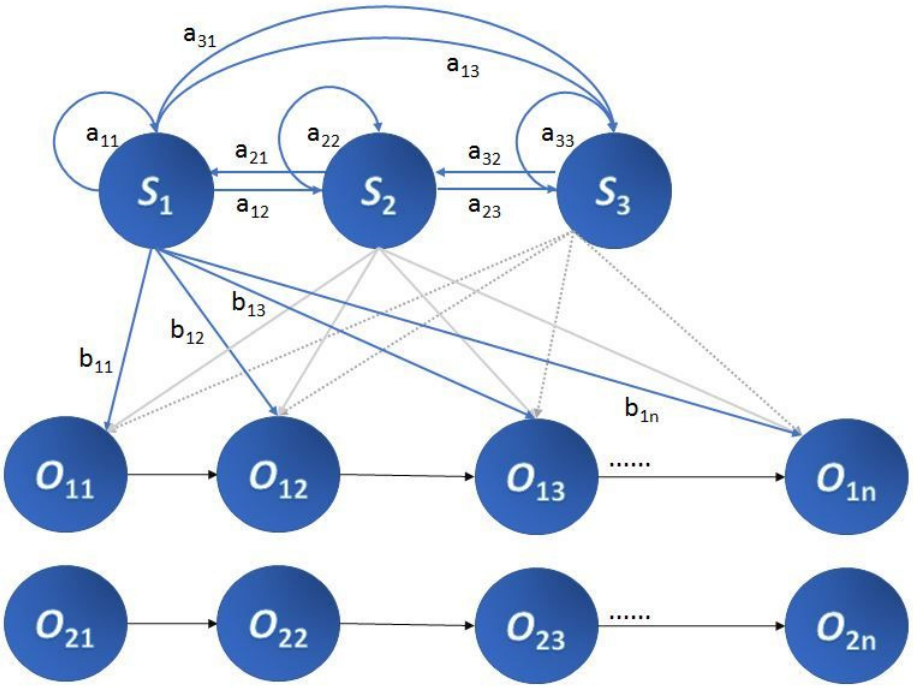

Figure 3-7: The transition and emission probabilities between states and observations

3.2.5. HMM Problems

By knowing the parameters described in the section 3.2.4, a Hidden Markov Model can be initialized. This model answers to three main questions. These questions are dependent to each other based on probabilistic. The questions with the relevant mathematical solution are introduced in this section.

1.What is the probability of an observation sequence $O = O_1, O_2, \dots, O_T$ (T is the total time) by a given λ?

This is also known as Evaluation problems. The goal is to find the probability of generation of this observation by the model. Proposed solution is to use Forward Backward algorithm [2]. Forward Backward algorithm computes the posterior probability for all hidden states given an observation. The method consists of two phases, the forward phase and the backward phase. In forward phase, the methods go forward in time an in Backward it goes backward in time on the observations. The forward variable α_t is defined by giving the model parameters λ. Since the forward model goes forward till time *t,* just a part of the observations is analyzed. So, the probability of partial observations till time *t*, staying at state S_i at time *t* is computed. This can be shown by Equation (3-26):

$$\alpha_t(i) = P(O_1, O_2, \dots, O_T, q_t = S_i|\lambda) \quad (3\text{-}26)$$

The forward variable for the first state is computed by the product of the initial and emission probabilities of the state, see Equation (3-27):

$$\alpha_t(i) = \pi_i b_i(O_1) \quad (3\text{-}27)$$

The next step is to calculate forward variable for the next states α_{t+1} by knowing the transition matrix, see Equation (3-28):

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}]\, b_j(O_{t+1}), 1 \le t \le T-1\,, 1 \le j \le N \quad (3\text{-}28)$$

Finally, the probability of observation given the parameters is defined by the sums of the forward variables, shown in Equation (3-29):

$$P(\mathrm{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (3\text{-}29)$$

The Backward procedure is similar to Forward procedure with minor changes. Instead of calculation partial observation from beginning to time *t,* the second part is analyzed. So, probability of the partial observation from *t*+1 till end is computed. $\beta_t(i)$ is the probability of the partial observation sequence from *t*+1 to the end, given state S_i at time *t* and the model parameter λ. This is calculated as shown in Equation (3-30). This work deals with Forward Backward algorithm, which is the combination of these two.

$$\beta_t(\mathrm{i}) = P(O_{t+1}, O_{t+2}, \dots, O_T|q_t = S_i, \lambda) \quad (3\text{-}30)$$

2.What is the most likely sequence of states $Q = q_1 q_2 \dots q_T$ by given λ and observations?

This problem is also known as the Decoding process. This is used to resolve the hidden states. Here, there is not a best correct sequence, but the most optimal one from a list of correct sequences. The Viterbi algorithm [15] is the proposed solution. The goal is to find the most likely state sequence as $\delta_t(\mathrm{i})$. Maximizing the probability of state sequences given the observation and the model parameters λ has to be achieved. Maximization is shown in Equation (3-31):

$$\max P(Q|O,\lambda) \qquad (3\text{-}31)$$

At time *t*, the following equation computes the highest probability of an observation.

$$\delta_t(\mathrm{i}) = \max_{q_1,q_2,\ldots,q_{t-1}} P[q_1, q_2, \ldots, q_t = i, O_1, O_2, , , \ldots O_t|\lambda] \qquad (3\text{-}32)$$

The next state is only dependent to the previous state. Therefore, for calculating the next state is the product of probability of being in state *i* (previous state) times transition probability from state *i* to state *j*. This is multiplied by the emission probability at the state *j*. So, the hidden state sequence is recognized, see Equation (3-33):

$$\delta_{t+1}(\mathrm{i}) = [max_i \delta_t(\mathrm{i})\ a_{ij}]\ b_j(O_{t+1}) \qquad (3\text{-}33)$$

The Viterbi algorithm is comparable to Forward Backward algorithm, where the backward steps are not computed. Comparing Viterbi with Forward algorithm, the difference is that in the computation of forward procedure, the probabilities are computed by addition. In contrary, Viterbi takes maximum of the previous state.

3.How to set the best parameters to maximize the probability of observations?

This is also known as the learning problem. The crucial part of HMM is to find parameters, which produce the model. Given a sequence of observations as trainings data, the parameters should be set, so that the observations be the output. There are two solutions for this problem, with iterative process like Baum Welch [15] or with gradient techniques [16]. Here, for recognizing the parameters $\lambda = (A, B, \pi)$ the Baum Welch algorithm is applied. Baum Welch is a specific type of EM (Expectation Maximization algorithm). In Equation (3-30) the parameter β for the backward process is shown. The probability at being in the state S_i at time *t* given the observation and HMM is indicated by γ, see Equation (3-34). Here the relation between Forward Backward and γ has been illustrated.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \qquad (3\text{-}34)$$

This has a normalization factor, therefore $\sum_{i=1}^{N} \gamma_t(i) = 1$. Now the probability of being in the state *i* at time *t* and being in the state *j* at time *t*+1 is calculated as $\xi_t(i,j)$, shown in the following equations.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (3\text{-}35)$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)} \quad (3\text{-}36)$$

By summing the $\gamma_t(i)$ over t=1 to t=T-1 ($\sum_{t=1}^{t=T-1}\gamma_t(i)$), the expected number of the transitions from S_i is determined. Doing the same for $\xi_t(i,j)$ ($\sum_{t=1}^{t=T-1}\xi_t(i,j)$) the expected number of the transitions from S_i to S_j is measured. Using these two quantities, the parameters could be re-estimated as below:

$$\pi_i = \gamma_1(i), \quad (3\text{-}37)$$

$$a_{ij} = \frac{\sum_{t=1}^{t=T-1}\xi_t(i,j)}{\sum_{t=1}^{t=T-1}\gamma_t(i)} \quad (3\text{-}38)$$

$$b_j(k) = \frac{\sum_{t=1\ \text{s.t}\ O_t=v_k}^{t=T}\gamma_t(i)\,,}{\sum_{t=1}^{t=T}\gamma_t(i)\,,} \quad (3\text{-}39)$$

The initial probability is the expected frequency (number of time) in the state *i* at time $t = 0$. The transition matrix would be the expected number from the state *i* to the state *j* divided by the expected number of times in the state *j*. The emission probability is the expected number of times in the state *j* and observing symbol v_k divided by expected time in the state *j*. It has been proven that it is better to use the current model to re-estimate λ for calculation of the right hand-side of the equations (3-37) and (3-39). Then by iteratively using the new parameters, the probability of observing *O* from model will increase (till an optimum) [15]. The end result would be the Maximum Likelihood Estimate of HMM. As mentioned before, the Baum Welch is a specific type of EM. The E-step can be a function $Q(\lambda, \bar{\lambda})$, and the M-step would be the maximization of $\bar{\lambda}$. So, this problem can be seen as an EM problem with the new E- and M-step.

3.2.6. Implementation Issues

In the previous section, the theoretical part of HMM is defined. Regarding to the use case it is used, HMM has some implementing issues. These are listed as: Scaling, multiple observation sequences,

initial parameter estimates, missing data and choice of model size and type. Some of the relevant issues are mentioned here. These parts are mostly with respect of Rabiner's paper [15].

Scaling

Scaling is an important part of re-estimation procedure. Baum Welch and Forward Backward algorithm are iterative methods. Their parameters should be re-estimated until a desired level of convergence is reached. Considering the definition of α_t in the equations (3-28) and (3-19), this has several terms. It can be shown in following equation. From previous part, *a* and *b* are very small numbers, significantly less than 1.

$$\prod_{s=1}^{t-1} a_{q_s,q_{s+1}} \prod_{s=1}^{t} b_{q_s}\ (O_s) \qquad (3\text{-}40)$$

As the number of observations become greater than 10 [15], the $\alpha_t(i)$ tends to zero exponentially. Thus, for much larger observations (e.g., 100 or more) the machines could not support such small numbers. Therefore, the scaling procedure must be used. The goal is to keep α in a dynamic range of $1 \leq t \leq T$. For that reason, a scaling factor is multiplied to the $\alpha_t(i)$. The scaling factor is dependent on *t* (not relevant to *i*, sequence length not to state). The mathematical structure is defined in the following equations. The re-estimation formula for transition matrix is regarding to the forward and backward variables, see Equation (3-41):

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)\ a_{ij}\ b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i)\ a_{ij}\ b_j(O_{t+1})\beta_{t+1}(j)} \qquad (3\text{-}41)$$

According to Equation (3-29), $\alpha_t(i)$ is each time computed based on *t*. This is multiplied to the scaling factor defined in (3-42):

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \qquad (3\text{-}42)$$

Knowing the scaling factor the scaled coefficient is computed by Equation (3-43):

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\ a_{ij}\ b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\ a_{ij}\ b_j(O_t)} \qquad (3\text{-}43)$$

Defining the induction, $\hat{\alpha}_{t-1}(j)$ is determined by the product of the initial probabilities and scaling factor, see Equation (3-44):

$$\hat{\alpha}_{t-1}(j) = \left(\prod_{\tau}^{t-1} c_\tau\right)\alpha_{t-1}(j) \qquad (3\text{-}44)$$

By inserting $\hat{\alpha}_{t-1}(j)$ from above equation in Equation (3-43), the scaled coefficient set based on α can be computed, see Equation (3-45):

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N}(\prod_{\tau}^{t-1} c_\tau)\alpha_{t-1}(j)\; a_{ij}\; b_j(O_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N}(\prod_{\tau}^{t-1} c_\tau)\alpha_{t-1}(j)\; a_{ij}\; b_j(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N}\alpha_t(i)} \qquad (3\text{-}45)$$

So, α is efficiently scaled by the sum over all states. A similar procedure is needed for the backward process. β tending to zero has to be scaled. This process is the like previous one. The difference is that the scaling factor is the inverse of sum of α. The magnitudes of α and β are comparable. This is shown in Equation (3-46):

$$\hat{\beta}_t(\mathrm{i}) = c_t\beta_t \qquad (3\text{-}46)$$

However, in some cases the scaling factor is not set correctly and is set to 1 [15]. Using the scaling terms causes to change of computing the $P(O|X)$, since $\hat{\alpha}_t(i)$ cannot be added together, because they are scaled. So instead, the sum of logarithm of coefficients is used in Equation (3-47). The probability itself would be so high and out of the machine dynamic range. As the number of observations increases, the scaling method, tend to his limitations.

$$\log P(O|X) = -\sum_{t=1}^{T}\log c_t \qquad (3\text{-}47)$$

Multiple observation

With one single observation, the model cannot re-estimate the parameters. The need of multiple observations is here clarified. Consider the model needs a small number of observations to calculate the transition to the successor state. The modification of re-estimation is simple, goes as follow:

$$O = \left[O^{(1)}, O^{(2)}, O^{(3)}, \ldots, O^{(K)}\right] \qquad (3\text{-}48)$$

Since $O^{(K)}$ is the set of Kth observation sequences so that $O^{(K)} = [O_1^K, O_2^K, O_{13}^K, \dots, O_{\tau t}^K]$ is a set of observations. The observations are independent to each other. The aim is to maximize the parameters such as the probability shown in Equation (3-49):

$$P(O|\lambda) = \prod_{K=1}^{K} P\left(O^{(K)}|\lambda\right) = \prod_{K=1}^{K} P_k \qquad (3\text{-}49)$$

The re-estimation procedure is based on the frequency of occurrence from various events. By adding the individual frequencies of occurrence, the parameter estimation based on multiple observations, is modified. For further details, refer to [15].

Multi Observation Continuous Density Hidden Markov Model

The standard model according to Rabiner [15] is designed for one-dimensional discreet observations and a single observation sequences. Later multi observation has been defined, completed with scaling for long sequences. But this project is a Multi Observation Continuous Density Hidden Markov Model (MOCDHMM). So, instead of applying standard HMM, a Multi-Dimensional Continuous Density Hidden Markov Model(MOCDHMM) must be applied [17]. MOCDHMM is a form of HMM which models a training dataset more precisely than the simple HMM. The MOCDHMM have been described in theory, but in practice they are seldom applied. Mostly these complex projects get simplified to adapt the standard HMM.

This project consists of various multi-dimensional observations. They are continuous observation vectors. Therefore, the work is more complex. So, high-dimension, long sequence, and continuous observation vectors are combined. A solution would be to decrease the dimensionality, discretize the vectors and concatenate the sequences. This causes a huge information loss, which is not the desired result. The dependencies between variables or covariances among observation vectors are required.

3.3. Model Selection

Model selection method should find simple, high quality model from a set of models. Different methods are developed such as: Akaike information criterion (AIC), Bayesian Information Criterion (BIC) [18], least squares, Cross validation and Mallows's Cp. In this work, the BIC and Cross validation are used. These procedures are defined in detail as follows.

3.3.1. Bayesian Information Criterion

Bayesian Information Criterion (BIC) or Schwarz Criterion is a model to measure the relative quality of parametric models with different numbers of parameters. It was developed by Schwarz adapting a Bayesian argument [19]. Each quality is computed based on the relevant data. Comparing two estimated models, the model with the lowest value of BIC is the one to be preferred. Using additional parameters for Maximum Likelihood model estimation may result in over-fitting. As a solution, BIC introduces a penalty term for the number of parameters. The formula described in [20], is given by Equation (3-50):

$$BIC = L - 0.5 * v * \log(n) \qquad (3\text{-}50)$$

In this equation, L is the log-likelihood of estimated model, where v is the number of free parameters to be estimated in the model. The number of samples is given by n. BIC is very close to AIC with the difference that the penalty of additional parameters is stronger than in AIC. Thus, it is a trade between how good it fits to the model and complexity of the model. BIC has also two main limitations. When the sample size is much larger than number of parameters, the BIC is valid. The second limitation is that a complex collection selection cannot be done with it. It has a problem with high-dimensionality.[21]

3.3.2. Cross validation

Determining the number of components in mixture models encounters theoretical difficulties. A useful model is proposed [22] to choose the number of components in a mixture model for independent data by maximizing the cross-validated likelihood. Cross validation is used here as a model selection method. It estimates the performance prediction for a model. The Advantage of Cross validation to BIC and AIC is that, Cross validation dose not assume that the actual distribution of the data belonging to other models. In contrary, in BIC and AIC this assumption is valid [23].The other advantage is using penalized likelihood criteria to prevent the technical difficulties. Using Cross validation in HMM to determine efficient number of states has potentially some difficulties. Choosing random observation from data to train results in breaking Markovian dependence between the states of the hidden Markov process. A solution is to partition the original sequence into long contiguous blocks and treat them as independent sequences [24]. By having various sequences a straightforward Cross validation approach can be implemented. Here the Cross validation procedure on sequences is used instead of observations. The training data is a multi-fold Cross validation. Forward Backward algorithm is used to compute the likelihood of the observations of each fold. The only problem here is by increasing the sequence size, it is subjected to underflow. This causes to unobservable probabilities and the model cannot rely on it.

4. Related Work

In this chapter, the works which are partially relevant to this work are described. For more simplicity, this chapter is divided into two sections. At the first part, Techlok project a Predictive Maintenance railway scenario is mentioned. Second part demonstrates the application of Hidden Markov Model in other projects. These projects intended to be as an inspiration for methods or procedures in the scenarios defined in this work.

4.1. Techlok

German National Railway Company "DB Schenker Rail" works on a predictive maintenance project named Techlok. The aim is to predict failures and preventative maintenance [25] with the support of diagnostic data. In some parts, the need of expert knowledge supports these data. The Knowledge Engineering group of Technische Universität Darmstadt cooperates with this company to design a failure prediction model. This project is a part of this cooperation. Therefore, it rests on earlier achievements and tries to predict the power converter failure with a specific model. Here, the major works done regarding to this project are explained.

In [26] different Machine Learning algorithms are evaluated to predict the train failure regarding to condition based maintenance. Predictive models work with the diagnostic data. In this paper, two procedures are introduced. In the first procedure, the failure is analyzed by the frequency of specific diagnostic code. The second process is about predicting the central screw breakage, which is not relevant for this work.

In [27], Kauschke et. al. analyzed the frequency of diagnostic codes in time based window to forecast the future component failures. This hypothesis defines that the incidence of specific error code increases near a failure occurrence. Therefore, an evaluation of these Machine learning algorithm, J48, JRip, Random Forest and SMO with 5-fold Cross validation has been done. Failure codes are collected and a labeling model is introduced. Precision and recall from the warning call were used to get the performance. For this, five experiments have been investigated. These are listed as below:

- Predicting the motor control failure.
- Predicting the guiding system failure.
- Predicting the guiding system antenna failure.
- Predicting the combination of the earlier failures.
- predicting failures with support of extended timeframes

The results verified the hypothesis. Predicting the failures is possible although the accuracy is still not acceptable. This is because of low data quality. From this work, the window framing idea is utilized in the project.

In [28] Kauschke et. al. developed a process to convert the PM problem to a classification problem. In this classification, all failure types are included. This thesis uses this process and develops a model based on this classification problem.

In [29] the report workshops information and the failure reports from hotline are combined with the diagnostic data. This causes an improvement of data quality to distinguish between various failure types. Guiding system failure and main air compressor failure are recognized by the incidences with the support of domain experts. To do this, the instances should be separated. But because of misleading, it cannot be directly used. Therefore, the criteria are applied to improve the labeling day of failures. ReliefF and CFSSubset are used for attribute selection. The classifying algorithms JRip, BayesNet and Random Forest were chosen for classification problem. A prediction experiment and an unnecessary layover experiment are attended. This is done to classify instances from warnings. Beside, it determines an instance that belongs to a layover with a repair (normal), or to an unnecessary layover. Unfortunately, the experiment results were not encouraging. However, useful features from the diagnostic data were extracted.

Using a layered model on the diagnostic data for predicting the power converter failure (PCF) regarding to the existing technology of locomotive series 185 is described by Kauschke et. al. in [30]. A classification model trained on the pattern structure of diagnostic codes has been developed to predict anomaly detector of instances. The prediction procedures are by a meta classifier on top of this anomaly detector. The paper is a successor of relevant paper [28]. In which, a specific failure type is predicted. Here the failure type is explicit set to power converter failures. A two-layer classifier is designed. At first, the instances are classified with support of decision trees based on supervised learning and fixed windows. The second approach is a meta classifier with combination of each instance prediction. This thesis rests on these achievements and tries to predict the PCF with separate learning model.

The above works were the inspiration for using partially their procedures and methods to handle the diagnostic messages and classification problems.

4.2. Hidden Markov Model

HMM is a generative model that can classify and reconstruct new instances. Therefore, it is appropriate for many scenarios, such as computer vision and scene interpretation.

In [31] Remagnino et. al. developed a cap-park scenario on video streams. This scenario is used to classify persons and vehicles in a video stream system, in order to figure out the entrance or exit of cars. The training data is divided into 4 subsets to be the training set for each class. The object classification is done by computing the forward procedure with computing the maximum posterior on an observation. The number of states changing between 5, 10, 15 and 20. Setting states greater than 20 results in over trainings. As the number of states increase, the accuracy will become greater.

Mirceva used Hidden Markov Model for protein classification in [32]. 61695 protein structures stored in repositories are classified here. The number of structures are exploded regarding to knowledge growth. So, the earlier algorithm gets unsuitable for the huge structure. Therefore, an alternative is to use HMM because of the speed effect on classifications (even reducing computation to minutes). Classification is regarding to the vary of single structure to multi-dimensional structures with a high complexity related. For this reason, for every class a separate HMM is trained. The probability of belonging to class is computed for each HMM. The next observation can be predicted using the similarity score which is calculated by the product of emission and transition probabilities. In this work, the Viterbi algorithm is used to estimate the most likely states by backtracking. The sequence length of observation should be equal. The number of hidden states vary between 16, 20 and 30. Although all configurations decrease the classification time, the accuracy with 20 states is maximum. This work partially rests on the handling method for long sequences.

The Hidden Markov Model described in [15] is a standard approach with one-dimensional discreet observations and singular observation sequences. Most related works decrease the complexity of the model to adapt the standards by decreasing the observations or quantizing multi-dimensional vectors in one dimension or both together. Since the present project is a Multi Observation Continues Density Hidden Markov Model (MOCDHMM), the following projects are introduced to get an inspiration how to handle the project.

Multi-dimensionality makes the HMM more complex. To overcome this, in several image processing projects, researchers reduce the dimensionality to one. Consequently, instead of the whole image, just a slice is observed. As in [33, 34], two dimensions are converted to one dimension by considering a set of images at a time. Thus, HMM processes two-dimensional images as an one-dimensional observation in mean of the gray scaled value. Other criterium changed in [34] was that the Viterbi algorithm is changed to produce Network Algorithm (NA). NA method is to traverse the

images pixel by pixel. This increase the likelihood of creating the image by the learnt HMM. The idea of dimensionality reduction to index bit comes from this work.

In gesture recognitions project [35], the multi-dimensional output of Probability Density Function (PDF) is reduced to the smallest number required to classify it by HMM. Gesture which could be reduced to one dimension were analyzed. Huang et. al. developed a similar hand gestures recognitions approach in [36]. Each finger is a variable as 5-long observation vector. Fingers positions are discretized in just three forms: Bending, half-bending, or straight. In this work, Vector Quantization (VQ) is used. VQ reduces the high-dimensional continuous vector to one-dimensional discreet observation using a code book. This procedure could be an alternative for the present study. Unfortunately, the dependencies between the feature get lost by dimensionality reduction.

A theoretical framework for multi-space continuous HMM application is introduced in [37]. The goal was to find a solution for having discreet one-dimension observation plus multi-dimension continuous observations simultaneously. The proposed technique uses a generalized continuous density Hidden Markov Model, in which the length of the observation vectors are not constant. The various dimensionality makes this work relevant to the present project.

Parallel HMM is a method introduced in [38]. This method trains different HMM for each component in observation. So, eight HMM are required to model an 8-length observation vector. Results show that the method is efficient for the analyzed scenario: Automated recognition of American Sign Language. But as each observation is modeled separately, the relations between variables are not modeled.

Mann used in [39] a more stable model to deal with multiple and longer observations. Here the logarithmic method is used instead of scaling. The logarithmic approach is also introduced by Rabiner [15], but only in Viterbi algorithm. Mann applied the logarithmic approach on other algorithms used in HMM. As the observation length in HMM increase, the probabilities decrease extremely. This results in numerical instability. An alternative solution was the scaling process defined in the previous chapter. But works not always as expected. The probability of observing sequence is used in Viterbi algorithm, Maximum Likelihood and Baum Welch algorithm. Therefore, this probability is a critical forecasting in HMM. Alternative to Scaling, the logarithms of the conditional probabilities can be computed. The advantages are: The code get simpler, debugging get easier and the need for scaling constant is dropped. So, Baum Welch learning parameters do not need to be re-driven. In absolute, the computation gets more simple and easier to test, but they are less efficient than scaling. Mann developed a procedure to bound the outlying conditional probabilities with the support of exponential logarithms. The small values of exponentials are heavily related to the sum of natural logarithms. Since the machine derives the small numbers as e^{-60} to underflow, while e^{70} goes to *Infinity*. So, the magnitude

of two numbers has to be equal unless the usable result change to overflow error. For a complete computation, the calculation of α and β and all relevant algorithms such as Forward Backward, Baum Welch and the calculation of ξ_t should be adapted to logarithmic form. For more detail the pseudo codes are described in [39]. However, the scaling approach in [15] is more efficient because of computing the logarithmic and exponential of each addition in logarithm approach. So, it is more useful for HMMs with small states and small feature sets. The use of logarithmic combined scaling method was an inspiration from this work.

5. Data Sets

This chapter gives an overview of the dataset and the algorithm applied to preprocess the data for Power converter failure. In the first section, history of dataset and methods applied on them are described. At the next section, the data form employed in this work is defined. The last section defines observation vectors and sequences.

5.1. Row Data

To benefit from Predictive Maintenance, DB-Cargo started the Techlok project in 2011. The goal was to find a specific hidden failure process to implement corrective processes. For this reason, the on-board computers monitored systems status and recorded them constantly. The history dataset from locomotive series 185 as log data are present. The recording system is event based. The event do not have any specific order to occur. By activating hardware relevant to power-converter, the system starts to record the system's status as log data. So, the occurred event is logged as a diagnostic message. Diagnostic messages are an encoded form of an event. The duration of this message is dependent to the duration of the event. It can be just a timestamp or it takes several days. For this procedure, an expert support is used to transfer the log data from each locomotive manually at the workshop entrance. Different events such as Errors, Warnings, Protocol messages are recorded as diagnostic data. Generally 6900 diagnostic messages exist [27]. The diagnostic messages consists of multiple parts such as diagnostic code, environment data, start and end timestamps and the locomotive series. An illustration of diagnostic message can be seen in Table 5-1.

Each event is assigned to a diagnostic code. For example, the diagnostic code 3985 is an encoded form of "ASG2 freigegeben". This means controller 2 of Powertrain got enabled. The start and end timestamp show when this message get activated and when it is deactivated. This enables the computation of the message duration. Starting a code does not lead to deactivating of other codes, which could be active parallel. Environmental data give extra information about system behaviors like motor temperature or coolant pressure. In this work, the environment data are irrelevant and will not be considered.

The diagnostic messages from 200 locomotives, model series of 185 (BR185), starting from 185001 till 185201, are analyzed. Every time a locomotive send the message "main engine powering on", the locomotive start to move. So, the logging procedure is started. Receiving the message "engine switching off", then locomotive stops moving. Logged data bounded in these two messages are defined as a tour. Thus, each locomotive has many tours. Tours are defined based on specific characteristics such as the duration of the tour and type of the tour. Tour types are either failure (breakdown) or

normal. Each tour has a set of diagnostic codes. This work focuses on the tours with at least 100 km distance covering and a minimum time duration of 2 hours. In the whole tours, there exist 40 tours of different locomotive series where PCF happened. These are defined as failure tours. The remaining ones are normal tours.

Table 5-1: Example diagnostic message

Train 185001			
Diagnostic Code	From	To	Environment
3992	14.03.2012 13:30:27	14.03.2012 13:31:47	A4CD120050000300000000B2000000005B150513
3984	14.03.2012 17:00:06	14.03.2012 17:01:11	A4CD12005000841C180100B2000000005B150513
3985	14.03.2012 17:00:00	14.03.2012 17:01:05	A4CD12005000841C180100B2000000005B150513
3993	16.05.2012 13:30:27	16.05.2012 13:53:16	A4CD12005000841C180100B2000000005B150513
3995	02.03.2012 08:02:30	02.03.2012 08:03:07	A5CD120050008502180300CE02080D0D51150E51
4014	24.12.2012 18:45:34	24.12.2012 18:45:34	A5CD1200500084021A0300CE0000000043151453

5.2. Pre-processing and Data Labels

To convert the diagnostic messages to instances, first the relevant parts are selected. This parts are diagnostic codes, start and end time. The algorithm defined in [30] is applied here. Each diagnostic code is converted to a Boolean flag. When the code is activated the relevant flag shows 1. In case of deactivated codes the flag show 0. The code is in between the start and end time activated, see Table 5-2.

Table 5-2: Instance example

Diagnostic Code	Start	End	Code 1	Code 2	Code 3	Code 4
Instance 1	14.03.2012 13:00:00	15.03.2012 10:21:15	0	1	0	0
Instance 2	14.03.2012 13:00:12	15.03.2012 04:20:46	0	1	1	0

In this work, the focus is not on the instances rather on the diagnostic codes. The same algorithm is applied to convert the diagnostic codes. The number of diagnostic codes relevant to PCF are 888 from the total number of 6900 codes. For the feature set (described in the section 6.1) a 14-dimensional Boolean vector is created. This vector consists of the 13 most important diagnostic codes (shown in Table 11-3) and the last instance is for the breakdown bit. The breakdown bit is activated in the case that one of 50 breakdown codes happens. These codes are listed in Table 11-4. In a tour, diagnostic codes are encoded as binary vectors. Each time one of the codes happen, the relevant index changes to 1. In Table 5-3 the feature set is shown. By incoming a code, the related flag changes to 1. Here, the diagnostic codes from locomotive series 185, number 188 is selected and the tour number is 39. Each code is illustrated as a vector showing 13 features of vector. These 13 feature are listed under the observation vector row in Table 5-3. The diagnostic codes, is demonstrated in the right side, are the incoming codes. In front of them would be the encoded form of their observation vectors. Since the vectors are observed in the model, its named as observation vector. The observed vectors and the related diagnostic codes are shown in Table 5-3

Table 5-3: Creating a vector regarding to feature set

Failure Tour Model Series 18518839													
Diagnostic Code	Observation Vector [6172	3995	11272	3985	8921	5240	4081	1417	3984	4004	4014	9999	1000]
8703	0	0	0	0	0	0	0	0	0	0	0	0	0
4081	0	0	0	0	0	0	1	0	0	0	0	0	0
3992	0	0	0	0	0	0	1	0	0	0	0	0	0
3995	0	1	0	0	0	0	1	0	0	0	0	0	0
34496	0	0	0	0	0	0	1	0	0	0	0	0	0
4014	0	0	0	0	0	0	1	0	0	0	1	0	0

The active flags stay 1 till the end of the timestamp. As the duration of the codes are different, the codes can get activated in parallel. In Figure 5-1 the procedure is illustrated. As the code 8703 get activated, the tour starts. This code does not belong to feature set, so components of the relevant observation vector is all set to zero. The next code 4081 arrives and the associated flag changes to 1. The duration of this code is long. As long as this code is activated, each incoming code should set the relevant flag of 4081 to 1 in his observation vector. This is shown in the related vector of code 3995. As the code arrives, the previous codes are still activated. The incoming diagnostic code 8703 does not belong to the feature set. So, it has not a flag. But the code 4081 is activated and the related flag is set to 1. The flag matching to 3995 is also set to 1. The next incoming code is 4004. This happens, when the code 3995 is still activated. So, the flags from the codes 4081, 3995 and 4004 get activated.

However, the code 3995 finished at the middle of 4004. This do not have any influence on the flag. Because at the starting time this code was activated. The next code is 4014. This happens, when the code 3995 and 4004 are deactivated. So, just the flags from the codes 4081 and 4014 are activated in his vector.

Figure 5-1: Creating vectors from failure codes

Each diagnostic code is encoded as a vector of 14 features. As the vectors are observable parts in the model, they are defined as observation vectors. Correspondingly the tours are a list or a sequence of observation vectors. The observation vectors could have 2^{14} situations from all 0 to all 1. The observation vectors can be repeated with respect of active diagnostic codes and timestamps. Thus, a tour can be very long with many duplicated vectors. The observation vectors can also consist of zero vectors. That means none of the 14 indexes happened. In Table 5-4 is shown that the observation vectors could have duplicated vectors and zero vectors. In this Table, a sample tour or a sequence of observation is distinguished.

To clean the data, zero vectors which do not have any information gain for the model, are omitted from the tour list. However, the duplicates could be important for the HMM (described in the section 3.2) to predict the next state based on the emission probability and using Viterbi algorithm (described in section 6.1). Nevertheless, in this scenario, this part is not relevant. Because in this scenario the goal is to classify the observation vectors, not to predict what could be the next observation vector. Therefore, the duplicates are also omitted.

As a summary, there exists tours as sequence of observation vectors. The tours have two types, failure tours and normal tours. For failure tours, 40 tours are labeled with support of expert knowledge. These tours consist of various model series. On the other hand, for normal tours 73000 tours are labeled, from which only 29796 tour are relevant. Every single model series contains several tours.

The sequences can have different lengths, since the tour length and the number of tours in various series are different.

Table 5-4: Failure code binary vectors

Diagnostic Code	From	To	Observation Vector
8703	14.03.2012 13:00:00	15.03.2012 10:21:15	[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
4081	14.03.2012 13:00:12	15.03.2012 04:20:46	[0 , 0, 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
3992	14.03.2012 13:00:33	14.03.2012 13:20:33	[0 , 0, 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
3995	14.03.2012 13:11:43	14.03.2012 13:13:12	[0 , 1, 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
34496	14.03.2012 13:13:01	14.03.2012 13:13:48	[0 , 0, 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
4014	14.03.2012 13:21:36	14.03.2012 13:42:36	[0 , 0, 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0]

6. Implementation

In this chapter, the methods and procedures utilizing in this work are introduced. In the present scenario, the relation between normal tours and failure tours are not the same. The number of normal tours are much larger than the amount of failures. In this project exists 73000 normal tours against just 40 tours for failures. An alternative would be to use Anomaly Detection [40]. But the aim is to benefit from this failure tours to detect the differences. In this scenario, the rare failure tours are utilized to detect the conditions which failure happen. For this reason, a method is developed to classify the tours as normal and failure. The methodic compares the data of both tours to find out the differences. The aim is to figure out the impact of diagnostic codes with support of HMM. This project is a Multi Observation Continuous Density Hidden Markov Model (MOCDHMM). As shown, the project consist of various observations, each in 14-dimensional continuous observation vectors. Therefore, the work is more complex. High-dimension, long sequence, and continuous observation vectors are combined in this project. Applying HMM algorithm to this project causes some errors. These errors and the proposed solutions are described in chapter 7. Log data (described in section 5) is used to train models. The amount of log data usually determines the procedure. The training models can be either standard or nonstandard behavior. To train the model for a nonstandard behavior, there has to be enough data for it [29].

The dataset for this project contains 40 failure tours and 73000 normal tours. But not all of them are relevant. The project analyze 200 locomotives from model series 185. So, the amount of 29796 tours are relevant to the model series 185001 till 185200. Each tour has to take minimum 2 hours long beside passing 100 km distance. The strategy is to train model to be able to classify tours, as normal tours, or failure tours. This is done with support of diagnostic codes. 2 parallel Hidden Markov models are developed, each for a tour type. Each model is trained by his own training data. These training data are the encoded format of tours. In order to classify new observations, the conditional probability of the observation with respect to any Hidden Markov Model is computed. The model with higher likelihood will be chosen. Thus, the new observations will be labels as the model type. According to this procedure, the classes are developed as HMM. The procedure is illustrated in Figure 6-1.

The procedure starts with creating the observation vectors, defined at the chapter 5. Afterwards, a model pool is created to figure out the optimal parameter settings. The best model is regarding to training data, selected for the HMM. Next step is to train two HMM for the two classes. Finally, the evaluation is done.

For implementation, the Jahmm package is used [41]. Jahmm is a Java library for learning and using Hidden Markov Models. To learn the HMM, the sequence of observation vectors are demonstrated as multivariate Gaussian observation probability distribution function.

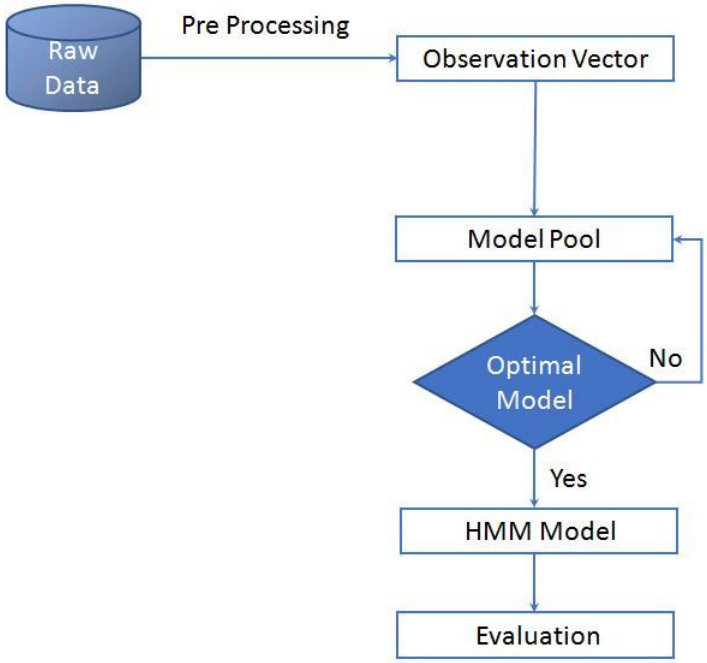

Figure 6-1: Project procedure

To train the HMM the data is transformed to observation vectors as described in the section 5.2. Here HMM is trained with sequences of observation vectors. The training data is selected randomly. Therefore, the successive tour is not in an order. This can be shown in Figure 6-2.the first training set is Tour *i*, follows by a random tour number *j*.

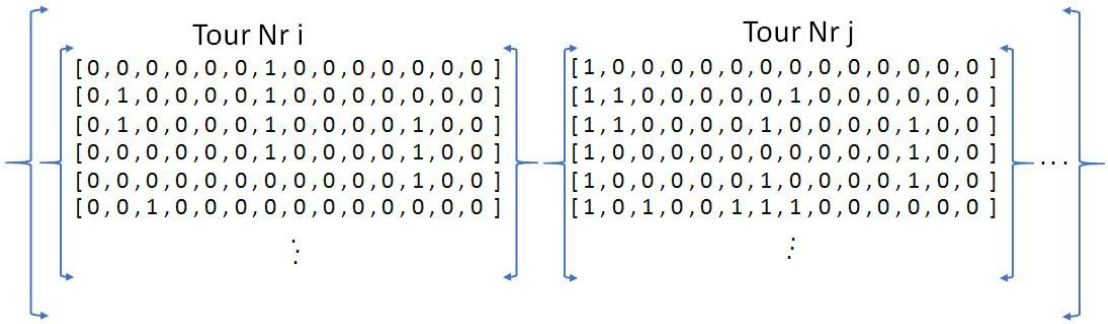

Figure 6-2: A sequence of observation vectors

For learning purpose, one of two algorithms can be chosen, K-means approximation and Baum Welch. The latter is a local optimization algorithm and therefore needs an initial HMM. The initial

HMM is generated by K-means and then, Baum Welch is used to optimize the initial HMM. As the labels are set in the data by expert, the learning method would be supervised learning.

6.1. HMM Model

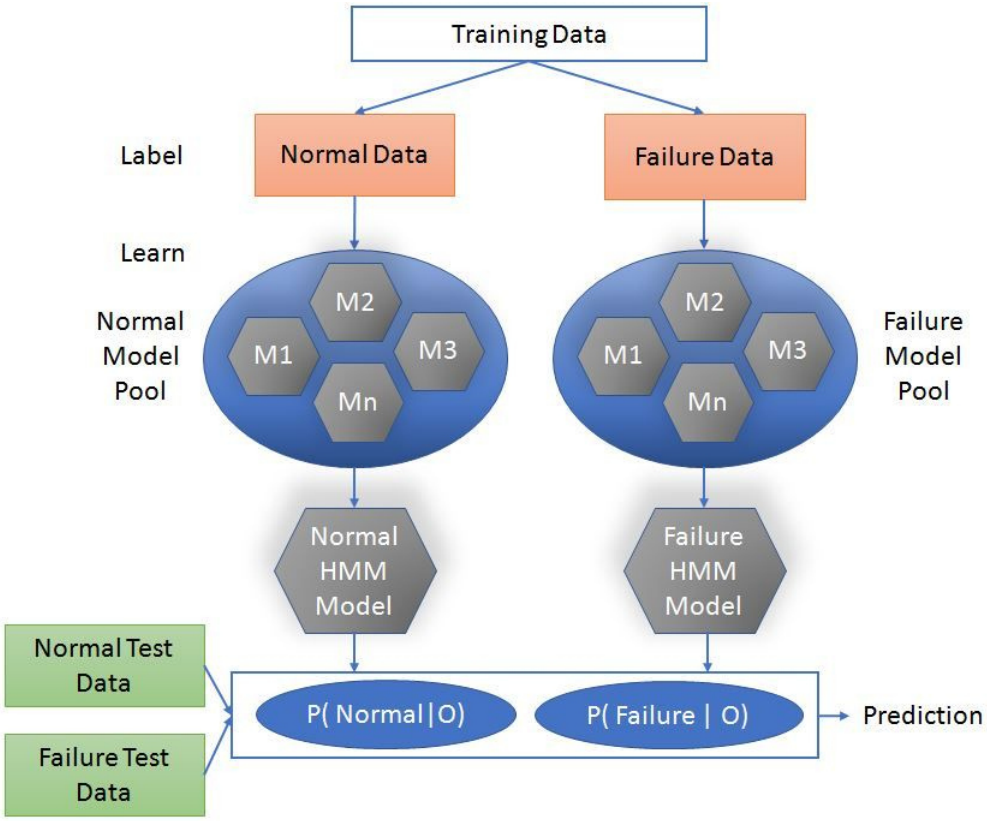

Figure 6-3: HMM procedure

Hidden Markov Model is a statistical tool for modeling generative sequences characterized by a set of observable sequences. The non-observable state of the system or simply the hidden states are governed by a Markov process. As mentioned in the section 5.2, the diagnostic codes are developed as vectors. These are the observation vectors in HMM. The observation sequence for training are sequences of diagnostic codes, here sequences of observation vectors. Beside other employment, HMM is utilized for classification problems. In this work, the goal is to define a binary classifier of tours. The classes are defined as normal and fail. Normal tours are those tours, which the train has successfully ended his tour. In contrary failure tour is not finished and the train tour is broken. In this scenario, different sequences of observations are available. The observations are used to train two Hidden Markov Models with respect of labels in the classifier. Then, given a new observation sequence, the classifier must classify the observation by computing its conditional probability with

respect to each model on the test observation. The model with the highest likelihood is the winner. So, the relevant label is the output prediction. The structure of this chapter is split into two parts, parameters and learning phase. In the parameter part, choosing the structural parameters and their topology beside creating the model pool are described. The second part is the learning phase. The classes will be trained here. In Figure 6-3 the procedure of HMM Model is illustrated. The labeled training data is divided into two sub categories. Each of them are set to relevant class. Model pools are designed to find out the optimal parameter configuration matching to each class. Each class has its own observation vectors in his model pool. After selecting the best match parameters, the HMM for failure and normal classes are trained. The next step is to input the test data. These data are separate for each class. To predict class for the test data, the conditional probability is computed given the sequence of observation vector. In other words, the maximum likelihood classifies a tour. The model with higher likelihood is the result class. The higher the probabilities so the model can generate the observation easier.

6.2. Parameters

Parameters in Hidden Markov Models are two types, structural and contingent. Structural parameters are those which defines the HMM and should be set at the beginning, such as number of hidden states defined in HMM and the feature set, which defines the dimension of the states. These parameters stay constant in the model. Further, the contingent parameters are under influence of the structural parameters. These are learned during the learning phase.

6.2.1. Structural Parameters

Structural parameters are the basic parameters in Hidden Markov Models. These parameters consist of the number of states and the features sets. In this part, these two set of parameters are described.

State Selection

One of the critical parts in Hidden Markov Model is to set the number of the hidden states appropriately. By choosing a large number, the model leads to overfitting. A large model has an enormous increasing of transition matrix values and emission probabilities. On the other side, by small number of states distinguishing between this model and other learned model would be difficult [31]. Therefore, it has been a criterion taking the number of the parameters into account. Although the states are hidden, but in practice this number is highly relevant to the physical details. There are also complicated mathematical techniques like BIC and AIC to predict the number of the states. However, the techniques are not guaranteed to give the optimal solution [35]. In this scenario, labels are failure and normal tours. But the states in each class is difficult to guess of the physical behavior. Therefore,

the BIC and accuracy as determination of the best states are used here. The optimum number for each model can be varying. In model pool this parameter is analyzed (more detains are following). For failure HMM a range from 1 till 8 states is selected, since the number of training data is not much. For normal HMM a range from 1 to 20 is selected, since the normal tours has more data. Therefore, it needs more states. However, to avoid more computation and robustness reason, the complexity should be reduced. Thus, models with smaller states should be chosen. The appropriate number of the states must be set such that a trade between the accuracy and the computational cost of evaluations, and performances is done. By choosing large state numbers the model fit to it. A very common complexity evaluation is to variate the sample amount and compute the performances. By large numbers the model can handle just large samples and could not give appropriate probabilities for smaller sets. Here sequences have different length. This makes learning more difficult. The model should fit to various lengths. The variability of the states has been limited to increase the performance of the model.

The algorithm in Model pool iterates over all possible combination of state configuration. For every configuration, a HMM is trained. Further, the test data is evaluated by two models. The BIC score and accuracy are used to determine the best state configuration. Computing the process for all the states needs a considerable amount of time. So, the search space is limited for each tour type. By analyzing the physical consideration of each model and including the above limitations, the states get bounded as shown in Table 6-1.

Table 6-1: Search space state selection

	Failure Tour	Normal Tours
Search Space	1-6	1-15

Choosing greater number than these numbers cause to bad prediction and overfitting problems. Obviously, because the tour is mostly on the normal model so, the variance is much higher. Therefore, this model should have more states. For this reason, maximum 15 states are assigned to normal HMM. For failure model a maximum of 6 states gives a high inter cluster similarity between observations.

Feature Selection

There exist 1663 diagnostic codes in the log data. The goal is to find the most important ones to predict the tours type. The number of possible combination of them would be too large for finding with Brute Force Method. Therefore, a subset of features is selected. In Table 11-2 the 888 diagnostic codes are shown. These are the important diagnostic codes, where the impact has to be to analyzed. But still the number of possible combinations is too much to try all combinations. There are methods to efficient search like Sequential Floating Forward Search (SFFS) and the Sequential Floating Backward Search

(SFBS) [42] to find the feature set. But we know the most impressive codes as they are most relevant to power converter failures. These are shown in Table 11-3. Finally, 13 diagnostic codes in combination of a set of codes which has the same effect named break down codes are selected. The breakdown codes are more than 50 codes. As mentioned before, the feature set consist of 13 diagnostic codes and the index 14. The algorithm iteratively checks through diagnostic codes. When one of these break down failures happens, the index shows 1. So, feature set has 14 numbers. This causes 2^{14} various features combinations, which the model must learn. The model should cluster 16384 different combinations. This is an extra-large number for HMM Models. The model complexity increases with more features. However, this number of features are the minimum, since the impact of the features on breakdown tours has to be analyzed.

6.2.2. Contingent Parameters

The contingent parameters are those parameters, which should be learned by the model to adjust in according with the structural parameters. In Hidden Markov Model, three base parameters should be set as listed below:

- Prior probabilities: Vector of probabilities of being in the first state of a sequence.
- Transition probability: Matrix describing the probabilities of going from one state to another.
- Emission probability: Probability of getting from the observation to the hidden state.

In this work, continuous vectors are were observed instead of a discreet observation. Thus, in any given state the HMM gives out an observation from a continuous distribution. Then instead of probability mass function, here exist a probability density function. Therefore, the observations are vectors of multiple values.

6.2.3. Train and Test Data

The data for this work is divided into two sub categories, training data and test data. Training data are randomly selected from tours, alike the testing data. Where the training tours are different from testing tours. The amount of normal tours is 50 tours which are randomly selected from 29796 normal tours as normal data and 30 tours from a total of 40 failure tours is randomly chosen for failure data. 10 Failure tours are set to testing. Normal test is done on an amount of 2800 tours randomly selected. More details are described in 8. As the model has limitations of amount of sequences, the tour with less than 10 observation vectors is irrelevant for normal tours. The result is that the model can adapt better to longer sequences. In failure tours the amount of tours are bounded. This causes to reduction of the amount of observation vectors to 2.

Multi-variate Gaussian distribution

The feature sets are multi-dimensional observation vectors. The distribution according to the continuous observation is set to Multi variate Gaussian distributions. The parameters of it should be set. So, the means and variance are also parameters which must be learned and set. The learn procedure is using EM. The first estimation is done, then the m step is repeated until convergence criteria is satisfying. A forward pass and a backward pass is done to associate the parameters in Multi variate Gaussian. Recalculate the HMM and multivariate Gaussian distribution parameters (mean, covariances, and mixture coefficients of each mixture component at each state) is computed.

6.3. Model Pool

To learn a HMM the size must be specified. This involves how many states and how many different observations are there. The number of training data is limited. Either, huge amount of training data will not specify the best approximation or less data results to errors on learning procedures.

Table 6-2: Model pool parameters searching space

Model Pool Parameters					
HMM	States	Training Data	Iteration	Stability	Experiments
Failure	1-6	30	100	1-100	285
Normal	1-15	30-300	1-100	1-200	1770

To avoid these problems a model pool is designed. In this pool, diverse models are set. The models differ in number of states, amount of training data, how many iteration has been done and the Numerical Stability Range. Based on each of them, a HMM is trained. The parameters of the model are shown in Table 6-2.

As mentioned the hidden stated could be different for failure and normal HMM models. More details are described in the section 6.2.1. In structural parameters. choosing an optimal number of iteration, the training data has a plausible declaration.

As the procedure is an EM algorithm, this contains several iterations. Each iteration has its own “estimate” and “maximize” step. In maximize step, each observation vector x is align to a state in the model, so that some likelihood measure is maximized. In estimate step, for each state *s,* two estimations should be set. First the parameters of a statistical model for *x* vectors aligned to *s* and second the state

transition probabilities should be set. The process is repeated till a likelihood measure stops rising significantly, or as the model converges a stable solution. But this does not mean that the algorithm creates the optimal distribution. The type of it is a hill climbing algorithm. In some cases, it converges to some local maximum which is not the best one, could be recognized. So, overtraining is happened. This is a problem which occurs by having too many iterations and large training data. Knowing this, the Model pool iteration parameter is for failure model till 100 and for normal model till 10 for long sequences and till 100 for long sequences.

The amount of training data in failure model is limited. This is revenue with the iteration numbers for training pool. The amount of training data varies from 30 to 300. The first tough was by learning with more training, the model stability get better. However, it is roughly reached the overtraining point. Adding additional training data has no positive effect in this experiment. In the section 5.2 observation vectors are defined. In practice, using them as training data result in very small probabilities, which were not recognizable for the machine. To make them more effective the vectors has become another range. In practice, we chose [1-100] stability range. Setting the stability range to 10, probabilities have been changed to recognizable numbers.

The Model pool procedure is shown in Figure 6-5. The model pool is created as a set of HMMs. Each model gets one of the parameters in Table 6-2. HMM is trained in the Model pool. Every single model in this pool is evaluated by 3-fold Coss validations. The mean of resulting accuracy is set to the model's accuracy. The model with highest accuracy is selected. Generally, 2055 models are trained and tested. The Model with smallest BIC and highest accuracy is selected as the optimal model. To select the training data for model pool, the procedure is as described in the section 6.2.3. The only difference is that for testing data a subgroup of training data is preferred. For training a 3-fold Cross validation is used. At each iteration, the training data is divided into 3 folds. The model is trained with two of them and tested on the remaining one, see Figure 6-4.At the next step, the conditional probability of test data is computed.

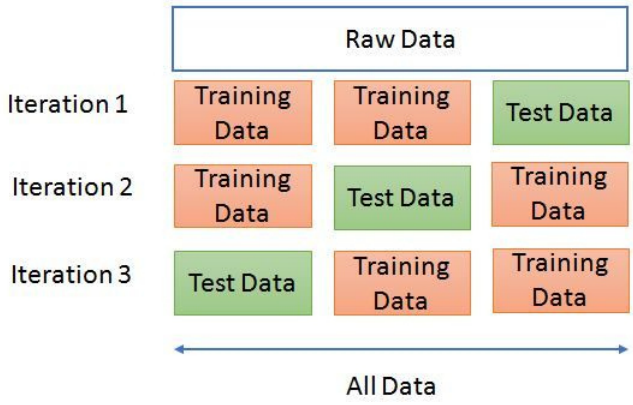

Figure 6-4: Cross validation for training data in model pool

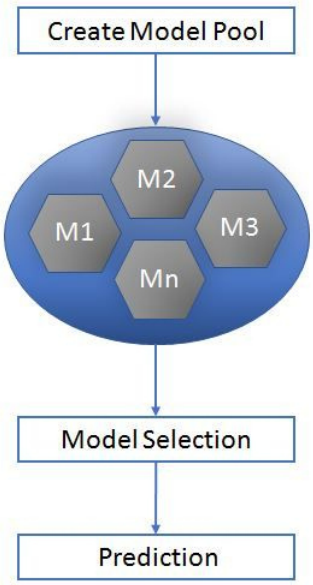

Figure 6-5: Model pool procedure

6.3.1. Training Model

Regarding to the result of the model pool, the parameters λ for each model is set. Algorithm 6 classifies observation sequence O using HMM. The algorithm takes three observation vector sets: The failure training set as $\varepsilon^F$and normal training set ε^N are defined. For the test set observation vector ε^U is decelerated. The classes of tests are undefined (for the machine, but the original label is known for

experts). The output from algorithm 6 is a set of labels $[O_i^U]$ assigning to the sequences $O_i^U \in \varepsilon^U$. After that the accuracy ratio is calculated for each class see Equation (6-1).

$$Classifier\ Accuracy = \frac{Correct\ Classified}{\text{Total Classified}} \qquad (6\text{-}1)$$

The classification method is as mentioned in algorithm 6 regarding to highest likelihood of producing the sequence. The whole process is repeated for normal class. K-means takes a training set as input and produce an initial HMM with respected parameters. This HMM is optimized by Baum Welch scaled algorithm. The log likelihood of a sequence is computed with Forward Backward algorithm for both classes and finally Maximum Likelihood sets the class label for the test sequences. In Figure 6-6 the algorithm is shown.

Algorithm 6: Classify Observation Sequences $(\varepsilon^N, \varepsilon^F, \varepsilon^U)$

$\lambda_N \leftarrow$ Kmeans Learn $(\varepsilon^N$,n)
$\lambda_F \leftarrow$ Kmeans Learn $(\varepsilon^F$, m)
$\lambda_N \leftarrow$ BaumWelch $(\varepsilon^N$,n)
$\lambda_F \leftarrow$ BaumWelch $(\varepsilon^F$, m)

for $O_i^U \in \varepsilon^U$
    $\mathcal{L}(O_i^U|\lambda_N) \leftarrow$ ForwardBackward $(O_i^U|\lambda_N)$
    $\mathcal{L}(O_i^U|\lambda_F) \leftarrow$ ForwardBackward $(O_i^U|\lambda_F)$
    labels$[O_i^U] \leftarrow$ argmax$_{x\in[N,F]}\ \mathcal{L}(O_i^U|\lambda_x)$
return (labels)

Figure 6-6: Classifying algorithm

The dataset for this work is very large, which causes to large sequences and makes a good prediction a truely challenge. An alternative of large sequences is the Scaling procedure defined in [15] (described in 3.2.6). So, a Baum Welch Scaled algorithm is learned. To calculate the likelihood of an observation given the learnt HMM, the scaled Forward Backward algorithm is computed. To focus more on the numbers, the probability is computed as Log likelihood defined in [39] and described in the section 3.2.6. For the initial probability, the very small number one divided by scaling factor(α) is set. This is done to reduce the zero probabilities when in exponential forms the probability get not defined (NaN). This number is so small that has not an effect of other computed probabilities. Otherwise the machine takes it as a number, similar to the step used in [16]. Detailed information about every applied mathematics issue is described in the next chapter.

7. Problems and Solutions

The standard HMM must be generalized to adapt the new condition of multi observation and continuous observations. This needs special training and specified evaluation techniques. Baum Welch learning and Forward Backward algorithm work well for discreet single observations. However, applying them in this scenario faces various mathematical challenges. These challenges are described in 7.1 followed by the solutions applied to them. To generalize the HMM to MOCDHMM, the Baum Welch algorithm and Forward Backward should be modified to fit the new conditions. The observations are transformed from single values to vector of multiple values. Thus, instead of computing the Probability Mass Function (PMF), an observation vector Probability Density Function (PDF) is computed. Following are the challenges faced within this project.

7.1. Continuous Observation Densities in HMM

Two methods exist for training an HMM: Baum Welch and Segmental K-means. The advantage of using K-means is that it is not sensitive to initial HMM parameters. By contrast with Baum Welch an initial HMM should be set. In this project, an initial HMM is exported by k-means and set it to the input HMM for Baum Welch. The Baum Welch algorithm optimizes contingent parameters from initial HMM. The issue is that, the standard probability density is for single discreet observation. But the observations in this project are as multi variable vectors defined. Although a complex solution is to quantize continuous variables via codebooks [43]. But the dependencies between variables get lost. Since the relation between variables are desired, an alternative solution should be found. The standard model should restrict the Probability Density Function (PDF) such that it can re-estimate the parameters constantly. PDF can be re-estimated by a finite mixture of them [15]. This is formulated in Equation (7-1):

$$b_j(O) = \sum_{m=1}^{M} c_{jm}\, \pi\left[O, \mu_{jm}, U_{jm}\right] \quad 1 < j < N \qquad (7\text{-}1)$$

Here O is the observation vector, c_{jm} is the mixture coefficient for the mth mixture in state j (hidden states). This is a positive number and in every state, should be sum up to one (for all *m)*. $\pi\left[O, \mu_{jm}, U_{jm}\right]$ is the log-concave density (a non-negative function which the domain is a convex set). Usually its Gaussians with μ_{jm} as mean vector and covariance matrix U_{jm} for the mth mixture in state *j*.

By using Equation (7-1), any finite continuous density function can be approximated [15].The main disadvantage in continuous HMM algorithm is that the speed and accuracy of the classifier vary

from other parameters in HMM such as sigma and down sample parameters. Where sigma determined how close each observation must match to the trained model to recognize it as a good model. If sigma is too low, the distance get to close to zero (or NaN, since the machine cannot determine it). This is a frequent problem in calculating Kulclback distance (The Kulback distance is used to find the distance between two HMMs). In contrast, by setting sigma too high, it causes matching the observation to all existing models. Sigma is here defined as stability range determined in model pool. The stability range there was chosen between 1, 60, 100 or 200. Lastly it has been set to 60. The observations are encoded in 5.2, to match it to the model. Different experiments have been done to find out the best matching range. Down sample parameter demonstrates the amount of training time series, down sampled for each state in the HMM.

7.2. Division by zero

By implementing a HMM, numerical stability should be considered. Specially the Forward Backward scaling recursions. This involves repeated multiplications of probabilities to determine the likelihood of an observation to a trained HMM. Smaller numbers less than one are extremely harmful as they effect on the probabilities unusual. These repeated multiplications of very small numbers generally lead to underflow [17]. However, this can be solved by normalizing after each recursion step, using the scaling algorithm defined in 3.2.6. Here the log Likelihood is computed directly by the negated sum of the logarithm of scaling factor. The relation is shown in Equation (3-47). Since the probabilities are still too large for the machine, logarithm base 10 is used instead of using natural logarithmic. In scaling model the α is calculated (shown in Equation (3-42)). By incoming a new observation w, if the observation is so different from the states PDF, then the probability underflows for the machine $b_i(w_t) \approx 0$. Thus, the Forward algorithm get a division by zero error. This is shown based on the formulas in the section 3.2.6 by the following equations.

$$w_t(\mathrm{i}) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(i) a_{ji}\, \mathrm{b}_i(w_t) = 0 \;\mathrm{i} \in \{1,2,3,\dots,\mathrm{N}\} \qquad (7\text{-}2)$$

$$c_t = \frac{1}{\sum_{i=1}^{N} w_t(i)} = \frac{1}{0} \qquad (7\text{-}3)$$

A solution is whenever $\sum_{i=1}^{N} w_t(i)$ get near zero, the c_t is set to ∞. By replacing the scaling factor to infinity, the log likelihood gets $-\infty$. This means that the relevant HMM is unlikely to produce this sequence.

7.3. Covariance matrix problems

A very common problem in this project was the difficulties with conditions in covariance matrix during training with Baum Welch. The errors can also occur in K-means segmental procedure. The hidden states distribute a multi-dimensional observation around the multi-dimensional Gaussian observations given in Equation (7-4) with μ as mean and Σ as covariance matrix of probability density function b_i.

$$b_i(O_t) = \frac{1}{2\pi^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(O_t - \mu)^T \Sigma^{-1}(O_t - \mu)\} \qquad (7\text{-}4)$$

Computing this requires to calculate the Σ^{-1}. If this density function contains a near singular covariance matrix Σ, the equation gives not the appropriate answer. To avoid this, a solution would be to calculate PDF $b_i(O_t)$ without computing Σ^{-1}. So, even a badly conditioned covariance matrix could produce a valid likelihood $b_i(O_t)$. An alternative would be to perform Rank, Pseudo-determinant, and Generalized inverse, see Equation (7-5).

$$f(x) = \frac{1}{\sqrt{(2\pi)^{rank(\Sigma)}\det^*(\Sigma)}} \mathrm{Exp}\left[-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right] \qquad (7\text{-}5)$$

The linear independent rows in Σ is the rank of the matrix. So, if a matrix $n \times n$ has the rank of n is full ranked and has determinant $|\Sigma| > 0$. A full ranked matrix with positive determinant has always an inverse Σ^{-1}. However, a full rank Σ might have $|\Sigma|$ to zero. So, it would be difficult to inverse it [44].

Pseudo-determinant ensures that also matrices with zero determinants in machine precisions are still supported by Opdf. In Equation (7-5) the pseudo-determinant replaces the determinant in Equation (7-4). The formula for a $n \times n$ full rank Σ is shown in following equation:

$$det^*(\Sigma) = \lim_{\alpha \to \infty} \frac{|\Sigma + \alpha I|}{\alpha^{n - rank(A)}} \qquad (7\text{-}6)$$

Generalized inverse is a completed form of inverse matrix. The product of the matrix and its invers should be the identity matrix ($\Sigma\Sigma^{-1} = I$). HMM performs well on predicting the labels (hidden states) of a fully observed sequence, but not for completing a sequence.

7.4. Long Sequences

The probability of generating a sequence by a trained HMM decreases by increasing the sequence length around 100. This probability is mostly near zero and is undefined for machine [15]. This results in undefined classification. Logically it is obvious that the longer the sequence get so the smaller the probabilities of producing the specific observation. A solution is using the Scaling method, but since the problem in this project is a collection of defined issues, Scaling helps not usually as expected. The sequences longer than 450 become sometimes not defined probabilities. Another issue is that the sequences for training and testing have different lengths. By choosing the tours randomly, the chance of training the model with various lengths is high. A solution for the long sequences would be to concatenate it to smaller sequences, but the observation vectors has a strong intern relationship. Since the failure code get active and stay active till the it reaches his end timestamp, this makes it unsuitable for this study. Another opportunity is to give a weight to the sequences, to trade between frequency of observe and the sequence length.

An alternative would be as in the joint distribution: There is a sum of probabilities given states and observation regarding to learned HMM. Dynamic programming is proposed by moving the summations inside. So, a lot of work can be saved and results get better.

In this chapter, some difficulties faced in this project were introduced. Each solution used to solve the problems were defined. In the next chapter, the archived results are demonstrated.

8. Results and Evaluation

This chapter presents and analyzes the results of the procedures described in Chapter 6, implementing procedure and the emerged issues and their solutions in Chapter 7. Two sections organize the chapter. The result of the model pool and classifier is introduced at first, followed by evaluating the classifiers result on the test data.

8.1. Results

The procedure of Model pool is described in the section 6.3. There are 285 models in failure model pool and 1770 models in the normal model pool. As shown in Table 8-1, the evaluated parameters are listed as: Hidden states, number of training data, iterating amount and stability range. The models are built by setting the mentioned numbers for HMM parameters. The maximum accuracy chooses the optimized model. By equal accuracy the one with lowest BIC is selected. The failure model matches with 2 hidden states and Stability Range of [0-60] as best. There is a limitation of training data for failure model. Since there are just 40 tours, 30 tours randomly are chosen for training, the rest are set for test. The iteration number do not have considerable impact on the learning ability in this model. In contrary for normal model 3 hidden states are set. The amount of 50 training tours are set to learn from. By increasing the tour number the model, adhere to a local optimum and get over trained. In contrary decreasing this amount results to an uncomplete learning model. The model cannot learn this variety of features sets. The optimal parameters for each model is shown in Table 8-1.

Table 8-1: Optimal model pool

Model	States	Training Data	Iteration	Stability Range	Experiments
Failure	2	30	100	60	285
Normal	3	50	100	60	1770

The optimal parameters have set to the model and the results of the model pool is shown in Figure 8-1. This shows the trade between maximum accuracy of each stability range. The diagram shows that the failure model learns better, in contrast to the normal model. The logarithmic stability range is set from 1 to 100. This can be seen on the horizontal axes of the diagram. The vertical axes shows the accuracies. The optimal models are selected for evaluation.

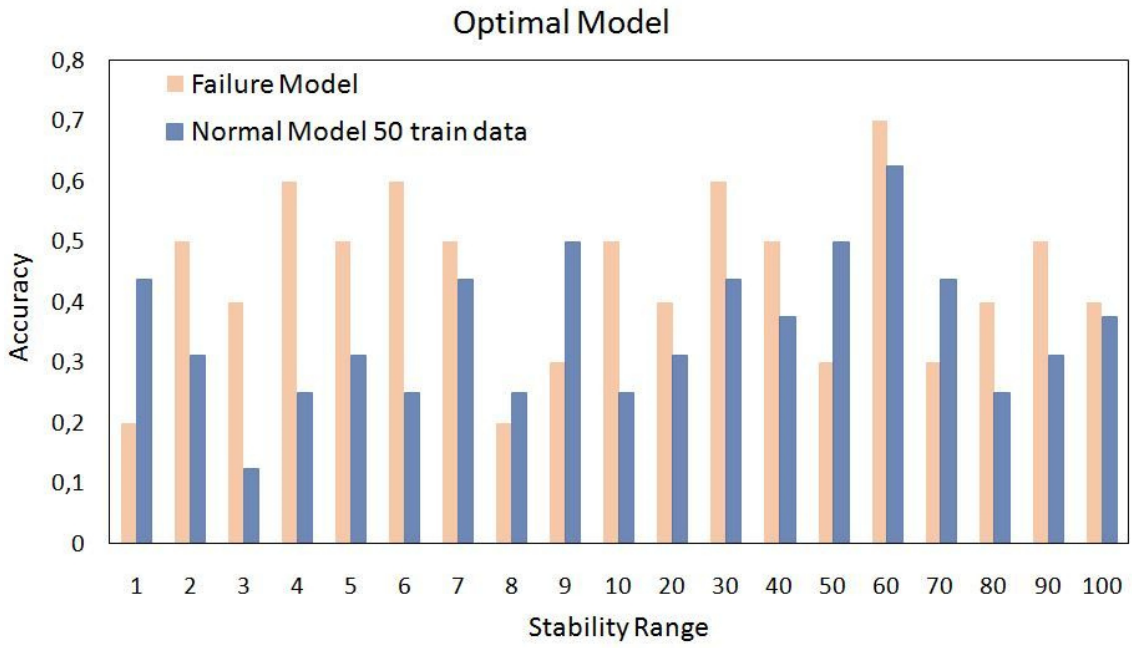

Figure 8-1: Optimal model pool chosen by accuracy

The models are strongly dependent on the training data. The accuracies change extremely by selecting other data sets for training. This leads to improvement or worsening accuracies. Finding out the optimal training set for failure model is easier than the normal model. The normal model can be trained with a huge variety of training data with different sequence lengths. These cause the complexity of training the model.

Different sequence length creates a complicated learning conditions for normal models to learn appropriately. As a solution here, the normal model is trained with length-limited sequences. That means a maximum number of 400 diagnostic codes can happen in each tour. Maximum bound is only set for training data and the test data could have longer sequences. This limitation has a positive impact on the model accuracy. The accuracy increases from 33% to 59% and the longer sequence probabilities are easier recognized.

The impact of training data on normal model is shown in Figure 8-2. This diagram is the representation of two different amount of training data and their accuracy rates. First the model learns from 100 tours. In contrary by the second experiment the model is trained with 50 tours. Every other parameter remains the same. The line graph shows by decreasing the amount of data, the model gets higher accuracies. In almost all the scaling ranges this issue holds true. On the other hand, by less data than 50, the model cannot learn suitably. Figure 8-3 shows the impression from different training data on accuracies from

failure models. The lines have some similarities. Since the sequence length of observation in training data in this model has less volatility, the accuracies make visible changes. More over, this shows the training data has to be chosen consciously, instead of selecting them randomly.

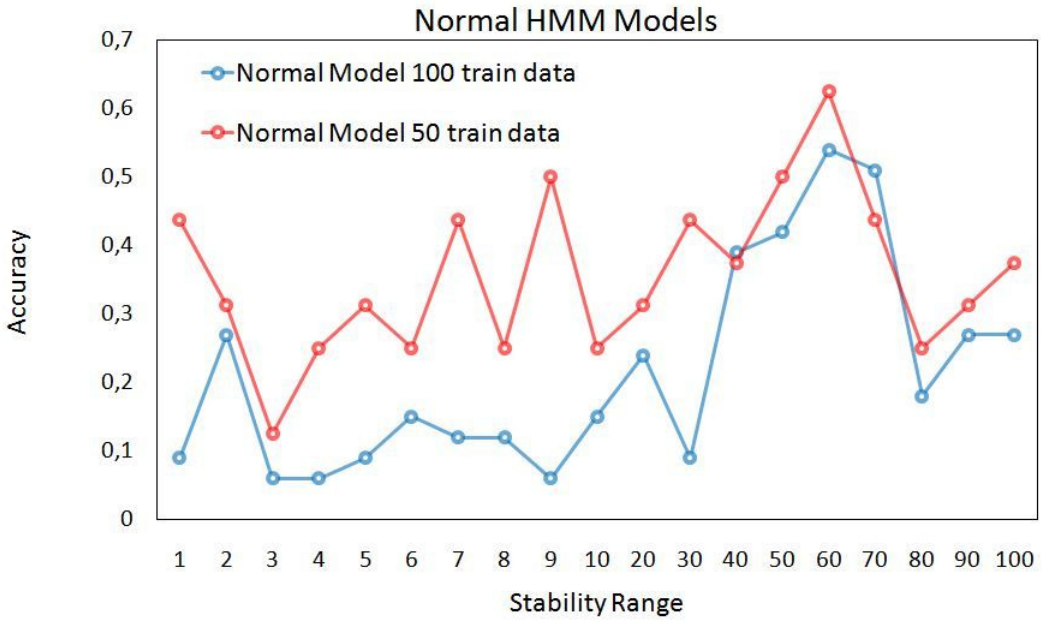

Figure 8-2: Normal model with different number of training data

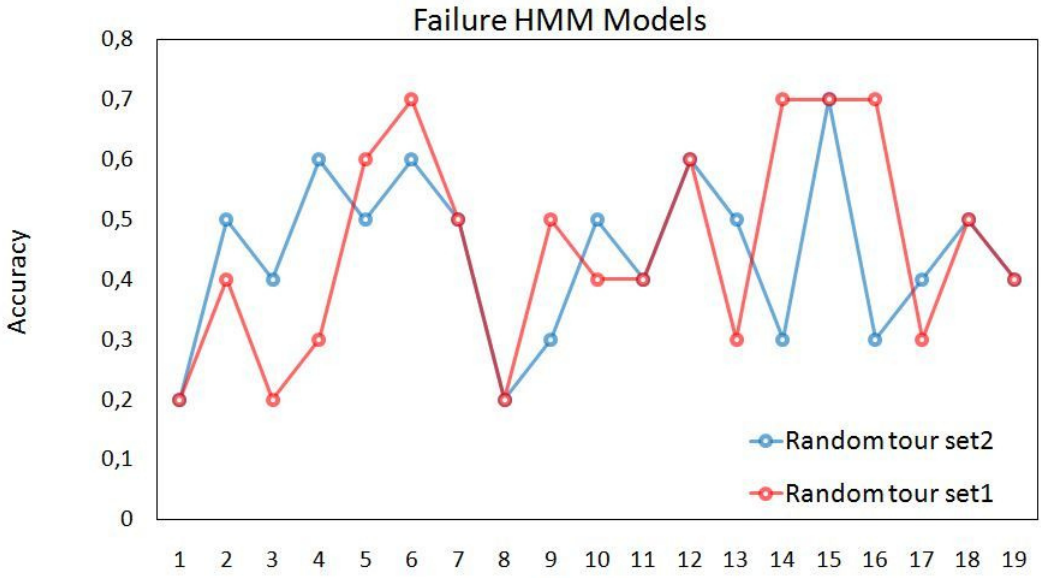

Figure 8-3: Failure model with different training data

By feeding the model with complete different training data, the output would be different. In this work, the training data is selected randomly. The model learns with the training data. In the learning iterations, the contingent's parameters are determined. These parameters are described in the section 6.2. The learned parameters from optimal models are shown in Figure 8-4 and Figure 8-5, as initial probabilities, transition probabilities and emission probabilities. As mentioned, the failure model is learned with 2 hidden states and the normal model with 3 states. The observed probability density means (Opdf) are listed for each state. Between failure model and normal model, the means have extreme differences. Opdf illustrates the mean value of the multi variate Gaussian. It can be interpreted as, how many times the codes happen in that state. As it is shown, the breakdown index is for normal model zero, in all three states. But in failure model, this value is set. The values are dependent on the training data. By changing the training data, means are changed. But the breakdown index is always set for failure models. A conclusion can be that if the breakdown index changes from zero to a number, the probability that the tour fail increases.

By learning the optimal training tours, the model produces the maximum accuracy. Classifier is the join of these two models. Prediction is done by comparing the models output. The outputs likelihoods can be listed as below:

- Finite number: This is the optimal solution for the model and shows the model can produce this sequence.
- Infinite Number: Shows that the sequence cannot be produce by this model.
- Not a Number (NaN): The model can produce this sequence but the probability is close to zero. Therefore, the number in not define for the machine.

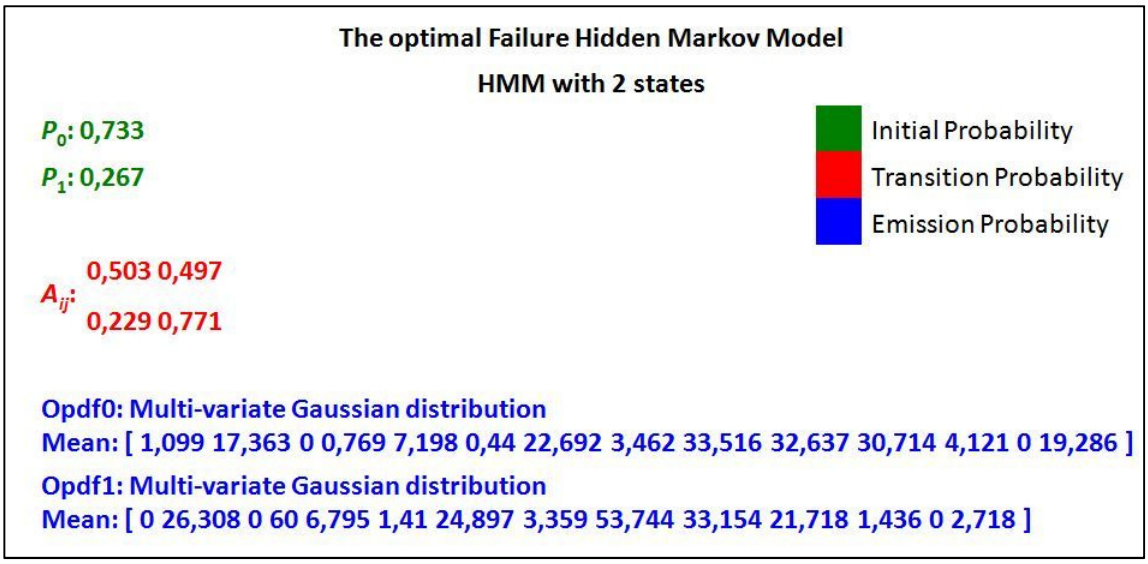

Figure 8-4: Optimal failure model

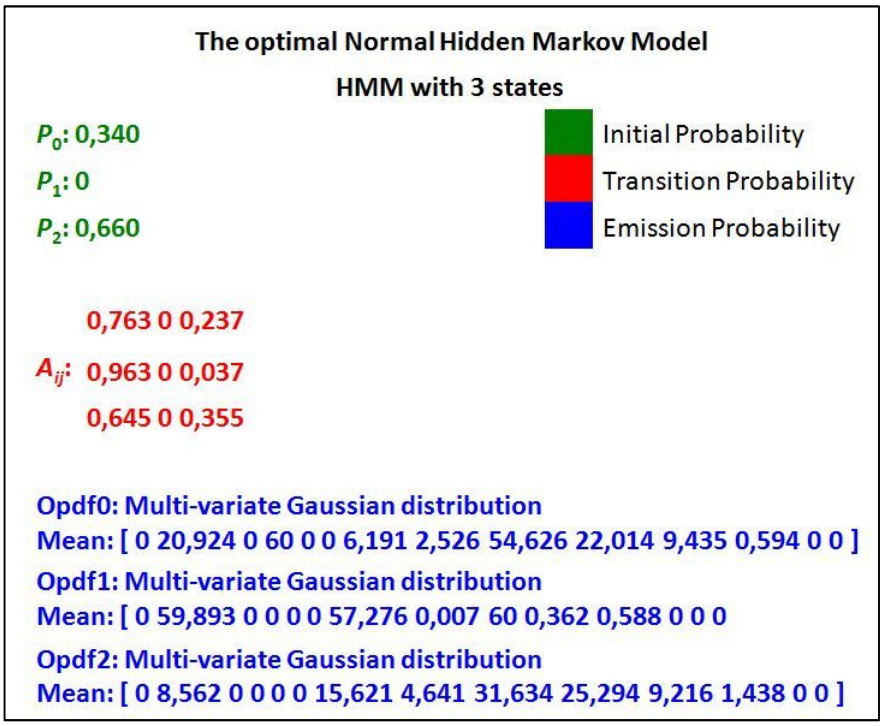

Figure 8-5: Optimal normal model

8.2. Evaluation

The classifier is tested on unseen tours by the model. The models compute the probability of the sequence separately. At the end the classifier compares between these two probabilities and make a prediction. The classifier can predict between normal tour, failure tour and not-classified. In situations which both models could not produce the sequence or both predict it with close to zero probability (which the machine define it as Not a Number (NaN)), the sequence is labeled as not-classified. Infinity probability means that a division by zero is happened and demonstrates that the model cannot generate this observation. Therefore, it is labeled as not-classified by classifier. Therefore, the tour will stay undefined. Based on a priori probabilities, these could also be set as normal. Here it is not set like that. Because in situations which both models get NaN as probability, there cannot be a comparison between them. The criterium which classifier is unable to predict is listed:

- Choosing between two infinite probabilities
- Choosing between two Not a Number probabilities
- Choosing between an infinite and a to Not a Number probabilities

The model quality is not constant. The model performance is highly dependent on training tours. These dependency is shown in Table 8-2.Two tests have been done, each with 50 randomly chosen training tours and tested on 400 tours. The results show the performance of the accuracy hangs out from training tour set. Here it can be seen that by changing the training data the accuracy of the classifier changed dramatically.

Table 8-2: Training data trade

Series	Model	Training Data	TP	TN	FN	FP	Not-classified	Accuracy	Precision	Recall	f1
test1	Failure	10	5	385	0	5	10	0,96	0,50	1,00	0,67
	Normal	400									
test2	Failure	10	5	84	0	229	87	0,22	0,02	1,00	0,04
	Normal	400									

This does not mean more data, but rather better qualified data are needed. The optimal training data have been recognized and the models are trained by them. Thus, specific training data must be selected instead of choosing them randomly. The quality of the model is dependent on the quality of the data. These should have specific criteria. First the training data must have different lengths from small tours till tours with length of 400. More than this makes a confusing impact on models. Tours should have low covariance on features for normal data and high covariance for failure data. This results in better probability density functions. Based on this the classifier is trained. As mentioned before the relation between normal tours and failure tours are (~73000:40) for all the model series. For a good evaluation, this relation must be reproduced in the experiments to get closely truth outcome. Unfortunately, this was not possible due to lack of time. Therefore, the evaluation is done on 2810 test data. The relation of the test data is 2800:10. The outcome is promising. The result is shown in Table 8-3.

Table 8-3: Results of the classifier with two models

	Test Data Set	Training Data	Label	Failure	Normal	Not-classified
Clasifier	Failure	10	Failure	5	0	5
	Normal	2800	Normal	57	2684	59

This table shows the impact of the classifier on the test data. The amount of test data is given. 10 tours from Failure model and 2800 tours from Normal model. Here the number of correct classified data are 5 from failure test and 2684 from normal test data. The classifier has labeled 57 normal tours as failure tour. Beside the total number of 64 are labeled as not-classified. The number of correct classified tours with failure model is acceptable. In contrary, it predicts an amount of number from

normal tours as failures. The classifier has a good separation between these two classes. None of the failure tours are predicted as normal tours. An indicator used in this project is the Classifier Accuracy (Equation (6-1) described in the section 6.3.1). This is used to evaluate between the failure and normal models. The Classifier Accuracy of the models is given in Table 8-4.

Table 8-4: Classifiers accuracy

Model	Training Data	Label	Classifier Accuracy
Failure	10	5	0,50
Normal	2800	2684	0,96

The Classifier Accuracy confirms the previous results. Here the accuracy of failure model is acceptable. In contrary normal model predicts 96 %. Thus, the normal model has predicted more correct classified tours as failure model. However, the failure model has an acceptable accuracy, but several tours are not classified. To get better results, these tours should be more analyzed. because of time reasons, it is not happened in this work.

If the classifier predicts perfectly, 2800 tours has to be classified as normal and 10 tours has to be classified as failure. But rather in this result, the true positives shows 5 correct classified failure tours. In contrary true negatives are 2684 correct classified normal tours. Then, false negatives is failure tours classified incorrectly as normal, here zero. The false positives are normal tours classified incorrectly as failure. The classifier detected 57 normal tour as failure. In Table 8-5 the result of the classifier is illustrated.

Table 8-5: Model evaluation

Model	Training Data	TP	TN	FN	FP	Not-classified	Accuracy	Precision	Recall	f1
Failure	10	5	2684	0	57	64	0,96	0,08	1,00	0,15
Normal	2800							1,00	0,98	0,99

The accuracy of the classifier shows a high accuracy. This shows that it has classified more instances correctly. By getting details on the two models inside the classifier, precision and recall evaluates their performance. If the classifier focuses on finding the normal tours, the normal precision and recall should be evaluated. But if the classifies insist of classifying the failure tours, the failure model's precision and recall should be looked closer. The predictors work with comparing the models

output. Therefore, for evaluation states, the classifier should be a compact model, not as individuum models. Having good performance in one model cause not a negative impact on the other models. So, each model should have good performances individually.

Precision shows here the amount of failure predictions divided by total number of failure predictions. The precision of failure model is 0.08 and the precision of normal model is 1. Precision shows the exactness of the classifier. So, low precision (in failure model) shows large number of incorrectly classified failure tours. As conclusion, the precision 1 in normal model shows the classifier is a good classifier as it does not predict any false negatives. Hence, none of the normal tours are predicted as failures. In contrary, the failure model is not acceptable, since it has predicted many normal tours as failures.

Recall demonstrates the number of correctly classified failures divided by the whole failure class in test data. Recall demonstrates how sensitive the classifier is and demonstrates a measurement of classifier completeness. Therefore, high recall shows that the classifier has classified less normal tours incorrectly as failure tours. Classifier has a high recall since the false positive rate is 0. Thus, in this project the recall must be large.

A balance between precision and recall is the F1 measure. By scenarios, where the false negative and false positive are equally worse, this must be used.

Predicting a normal tour as failure is better than to predict a failure tour as normal. Thus, false negative is worse than false positive. The precision is for normal model 1 and for failure model 0.08. This demonstrated that the model can predict the normal models better. Beside it has not classified failure tour as normal. For failure, the statement could be although the model classifies relevant results but it also classified many irrelevant tours as failures. In general, beside 5 correct classified tours, 57 from normal tours are also lobed as failure. On the other hand, recall =1 shows that failure model has returned most of his relevant result. Generally, the classifier can predict normal tours better than failure tours. A solution is to train the models not by random tours rather by optimal training tours. However, the failure model could not reach great accuracy, but the results shows that the model could differentiate between the diagnostic codes, which happen in each specific tour.

9. Conclusion

The aim of this work was to design a failure prediction model. This has to predict the power convertor failure on the historical data driven from DB Cargo systems. For designing the predictor two HMMs were applied. Each correspond to one tour category: failure and normal. Data are encoded as multi-dimensional binary vectors. The feature set defines the dimensionality. To set them appropriately, a subset of diagnostic codes is chosen. A model pool is designed to resolve the optimal parameter configuration. The pools vary from different parameters settings. The model is an incremental model. Therefore, less training data is needed to determine a better model. The model's quality depends on the quality of data. However, the data quality lives a lot to be desired. Thus, specific training data must be the selected instead of choosing them randomly. Finally, the optimal parameters are set to every model. So, the models are trained and finalized to predict. The classifier predicts on new incoming tours and gets the Classifier Accuracy of 96%. This indicates that the classifier can predict the tours in good rates.

The aim was to design a power converter failures. The results show that the failure model learns good enough to predict half of the relevant tours as failures. The normal model is also trained so good that can predict almost near to all his tours as normal. The classifier has not predicted any normal tour as failure. This is a very good result. This shows the models has so differences that a failure observation is for normal model undefined.

The predictors are performed by comparing the models output. Having good performance in one model does not cause a negative impact on the other model. So, each model should have good performances individually. The performance of the classifier can be evaluated by analyzing them. The classifier is evaluated by precision and recall to support the prediction. In this scenario predicting a failure tour as normal is worse than predicting normal tour as failure. Classifier Accuracy for the models determines how good they have classified the same label tours. The Classifier Accuracy rate for normal model is 96% and for failure model is 50%. As a conclusion, the model is reliable to have an almost good prediction of failure tours, which could undoubtedly still be improved.

An important aspect is that the model's performance varies in different tours situations. Longer tours are more difficult to classify. Tours with less codes or less events have a bad influence on the classifier. In general, the model has an acceptable measure of performance. However, analysis is done on a subset of feature. These encourages to generalize the model forward all features. More research and development is called for.

Further ideas could be applied to get better performance. For learning the models instead of Expectation Maximization, deviation techniques could be applied. This avoids learning problems like learning long sequences and high covariances facing up in this project. Another thing which should be considered is the list of training set. The criteria of finding good training data could be more analyzed. An alternative would be to use balancing or weighting methods to predict. The problem with very long sequences should be treated. Two opportunities are recommended. First to split the sequences and handle them as individual tours. Second, the length of the tours get ranged. Then the tours get classified by their range. In this situation for each range a separate model is learned. In other words, dividing the tours in subcategories based on length, and learning separate HMM on them.

For analyzing the whole features an encoding process can be used. The features are categorized into groups, some like the breakdown bit used in this project. So, instead of features, the groups are set as states. The classifier can also be modeled with other models like neural networks and Support Vector Machines. These algorithms are more suitable for large features. Lastly, the model can be changed to a discriminative model like random field. The discriminative models are more flexible with complex features.

10. REFERENCES

[1] Marjanovic, A.; Kvascev, G.; Tadic, P.; Djurovic, Z. Applications of predictive maintenance techniques in industrial systems. Serb. J. Electr. Eng., 2011, 8, 263–279.

[2] Zahlen, Daten & Fakten: https://www.dbcargo.com/rail-deutschland-de/unternehmen/zahlen-fakten/dbcargo-in-zahlen.html, 2017.

[3] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P. From Data Mining to Knowledge Discovery in Databases. IEEE Expert: Intelligent Systems and Their Applications, 1996.

[4] Bishop C.M. Pattern Recognition and Machine Learning. Springer, 2006.

[5] Ayfan, M. Machine learning. http://www.ineffable.in/technology/ database/machine-learning/, 2016.

[6] Witten, I.H.; Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. The Morgan Kaufmann Series in Data Management Systems, 2011.

[7] Fisher, R.A. On the Mathematical Foundations of Theoretical Statistics. Philosophical Transactions of the Royal Society, 1922.

[8] Stuart, A.; Ord, J.K. Kendall's advanced theory of statistics, 6th ed.; Wiley, 2004.

[9] Gernot A. F. Markov Models for Pattern Recognition: From Theory to Applications. Springer Science & Business Media, 2008.

[10] Barber, D. Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012.

[11] Saxena, A.; Celaya, J.; Saha, B.; Saha, S.; Goebel, K. Evaluating algorithm performance metrics tailored for prognostics. In: IEEE Aerospace conference 2009; pp. 1–13.

[12] Maimon, O.; Rokach.L. Data Mining and Knowledge Discovery Handbook; Springer, 2010.

[13] Esmael, B.; Arnaout, A.; Fruhwirth, R.K.; Thonhauser, G. Improving time series classification using Hidden Markov Models, 12th International Conference 2012.

[14] NIST/ SEMATECH. e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/.

[15] Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition-Proceedings of the IEEE, 1986.

[16] Gultyaeva, T.A.; Kokoreva, V.V. Classification of multidimensional observation sequences described by Hidden Markov Models 12th International Conference 2014; pp. 556–561.

[17] Ross, M.P. Multi-Observation Continuous Density Hidden Markov Models for Anomaly Detection in Full Motion Video. Master Thesis 2012.

[18] Burnham, K.P.; Anderson, D.R. Model selection and multimodel inference: A practical information theoretic approach, 2nd ed.; Springer, 2002.

[19] Schwarz, G. Estimating the Dimension of a Model. Institute of Mathematical Statistics is collaborating with JSTOR to digitize, preserve, and extend access to, 1978, 461-464.

[20] Burnham, K.P.; Anderson, D.R. Multimodel Inference: Understanding AIC and BIC in Model Selection. Colorado Cooperative Fish and Wildlife Research Unit (USGS-BRD), 2004.

[21] Kass.R. E; Wasserman. L. A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. Journal of the American Statistical Association, 1995, 928–934.

[22] Robertson, A.; Kirshner, S.; Smyth, P. Downscaling of daily rainfall occurence over Northeast Brazil using a hidden markov model. Journal of Climate, 2004, 4407–4424.

[23] Ripley, B.D. Pattern Recognition and Neural Networks; Cambridge University Press, 2000.

[24] Celeux, G.; Durand, J.B. Selecting Hidden Markov Model State Number with Cross-Validated Likelihood. Springer, 2007.

[25] Sandesh Nair. Predictive Maintenance in a Railway Scenario using One-Class Support Vector Machines,: Master Thesis, 2016.

[26] Kauschke, S. Machine Learning zur Vorhersage von Schäden an Lokomotiven: Diplomarbeit, 2014.

[27] Kauschke, S. Learning to Predict Component Failures in Trains. In: Introduction to Nonviolence. Jahanbegloo, R., Ed.; Macmillan Education UK: London, 2014; pp. 1–10.

[28] Kauschke, S.; Janssen, F.; Schweizer, I. On the Challenges of RealWorld Data in Predictive Maintenance Scenarios: A Railway Application. Processing of the LWA, 2015.

[29] Kauschke, S.; Janssen, F.; Schweizer, I. AdvancesInPredictiveMaintenance. Technical Report TUD–KE, 2015.

[30] Kauschke, S.; Fürnkranz, J.; Janssen, F., Eds. Predicting Cargo Train Failures: A Machine Learning Approach for a Lightweight Prototype; Springer: Cham, 2016.

[31] Remagnino, P.; Jones, G.A. Classifying Surveillance Events from Attributes and Behaviour. British Machine Vision Conference 2001.

[32] Mirceva, G.; Davcev, D. HMM based approach for classifying protein structures; Springer, Berlin, Heidelberg, 2009.

[33] Samaria, Ferdinando and Steve Young. HMM-based architecture for face identification. Image and Vision Computing, 1997.

[34] Vstovsky, G.V. and A.V. Vstovskaya. A class of hidden Markov models for image processing. Pattern Recognition Letters, 1993.

[35] Wilson, A.D. and A.F. Bobick. Parametric hidden markov models for gesture recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on,, 1999.

[36] Huang, C.L., et al. Gesture recognition using the multi-PDM method and hidden Markov model. Image and Vision Computing, 18(11), 2000.

[37] Tokuda, K., et al. Multi-space probability distribution HMM. IEICE Transactions on Information and Systems E series D, 2002.

[38] Vogler, C. and D. Metaxas. A framework for recognizing the simultaneous aspects of american sign language. Computer Vision and Image Understanding, 81(3), 2001.

[39] Mann, T.P. Numerically Stable Hidden Markov Model Implementation, 2006, 1–8.

[40] Perera, S. Introduction to Anomaly Detection: Concepts and Techniques, https://iwringer.wordpress.com/2015/11/17/anomaly-detection-concepts-and-techniques/, 2015.

[41] Francois, J.M. Jahmm - An implementation of HMM in Java.

[42] Kudo,Mineichi. Sklansky,Jack. Comparison of algorithms that select features for patternclassifiers. Pattern recognition, 2000.

[43] Kekre, H.B.; Sarode, T.K. An Efficient Fast Algorithm to Generate Codebook for Vector Quantization,First International Conference 2008; pp. 62–67.

[44] Easton, M.L. Multivariate Statistics: a Vector Space Approach. John Wiley and Sons, 1983.

11. Appendix

Table 11-1: Abbreviations

AIC	Akaike information criterion
BIC	Bayesian Information Criterion
MSE	Mean Square Error
MAPE	Mean Absolute Percentage Error
ML	Maximum Likelihood
HMM	Hidden Markov Model
SFFS	Sequential Floating Forward Search
SFBS	Sequential Floating Backward Search
PCF	Power Converter Failure
PDF	Probability Density Function
VQ	Vector Quantization
MOCDHMM	Multi-Observation Continuous Density Hidden Markov Model

Table 11-2: Pre-selection diagnostic codes

Diagnostic codes first selection												
94	509	606	1021	1024	1026	1060	1095	1096	1099	1105	1106	1107
1137	1141	1142	1147	1148	1149	1150	1154	1188	1191	1198	1200	1203
1204	1205	1248	1249	1251	1252	1253	1280	1282	1288	1289	1290	1292
1294	1298	1300	1304	1314	1316	1317	1319	1354	1408	1411	1417	1418
1420	1421	1424	1425	1426	1427	1429	1430	1434	1436	1446	1447	1449
1451	1457	1458	1460	1437	1443	1444	1466	1468	1471	1474	1475	1480
1482	1463	1464	1465	1487	1489	1491	1492	1495	1496	1501	1483	1485
1486	1511	1512	1513	1514	1522	1523	1524	1502	1507	1510	1544	1545
1546	1548	1551	1554	1559	1525	1531	1536	1579	1582	1585	1587	1591
1593	1594	1574	1576	1578	1595	1599	1600	1601	1605	1606	1607	1612
1613	1614	1615	1616	1618	1619	1620	1621	1634	1635	1651	1652	1664
1667	1668	1670	1672	1680	1682	1683	1697	1698	1699	1726	1738	1749
1750	1753	1755	1807	1808	1809	1810	1811	1812	2186	2188	2189	2191
2192	2195	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2216
2218	2223	2243	2245	2255	2256	2275	2287	2304	2305	2308	2310	2311
3332	3337	3339	3340	3341	3984	3985	3992	3993	3994	3995	4000	4001
4002	4003	4004	4005	4006	4011	4012	4014	4015	4016	4025	4029	4031
4032	4033	4034	4035	4017	4019	4024	4064	4065	4066	4067	4068	4069
4070	4038	4039	4059	4075	4080	4081	4082	4083	4084	5120	4071	4072
5150	5156	5160	5161	5162	5121	5122	4074	5125	5148	5210	5211	5212
36990	36991	36982	5213	5214	5163	5166	5123	5194	5200	5220	5221	5222
36983	34734	36988	5223	5224	5215	5216	5192	5218	5219	5230	5231	5232
37003	37004	37066	5233	5234	5225	5226	5217	5228	5229	5240	5241	5242
8916	8917	8466	8467	8454	8522	8544	8958	8959	9984	25615	25650	24759
9986	9987	8921	8923	8511	8956	8957	9997	9998	9999	25744	25745	25655
10000	10001	9988	9990	8955	9994	9995	10118	10119	10238	26122	26125	25747
10251	10252	10002	10003	9993	10112	10114	10270	10271	10294	27655	27766	26126
16375	16376	14214	14218	14179	14224	14225	16908	16921	16939	24760	24713	26149
10295	10297	10257	10265	10004	10267	10268	10763	10769	10777	27872	27949	27773
16955	16957	16377	16378	14221	16380	16382	17450	17451	17454	25730	24761	27784
10778	10779	10303	10305	10266	10358	10689	10806	10809	10815	28154	28177	27953
17461	17466	16963	16971	16379	17414	17426	17486	18180	18187	25748	25731	27960
10816	10817	10780	10781	10318	10786	10801	11272	11273	11276	28295	28302	28191
18234	18245	17467	17468	16997	17483	17485	18275	18277	18279	26144	25756	28251

11277	11282	10830	10870	10783	11200	11201	11528	11532	11533	30768	30976	28332
18285	18287	18246	18261	17472	18263	18274	18306	18310	18314	27774	26145	28569
30997	32897	32898	32899	33081	25759	26116	26188	26191	26499	28252	28670	30725
28593	31010	31024	31250	28290	27862	26183	27863	27864	27869	27964	27967	30756
11538	11539	11283	11284	10872	11286	11328	11794	11795	11796	31251	31252	30977
18315	18319	18293	18296	18262	18300	18305	18326	18327	20471	27956	27782	30996
11797	11798	11540	11541	11285	11584	11788	12051	12052	12053	33039	33040	31490
20472	20474	18320	18321	18299	18324	18325	21507	21512	21515	28192	27959	31763
12054	12096	11840	12040	11542	12045	12050	13354	13358	13359	33128	33130	33051
21527	21528	20475	20478	18322	21505	21506	23554	23555	23559	28333	28250	33077
33078	33170	33184	33218	34305	25732	25733	26119	26120	26121	32896	33079	33080
13370	13371	12101	12812	12044	12843	12867	14091	14138	14149	33220	33223	33132
6200	6201	6189	6190	6181	6193	6194	8218	8446	8447	34689	33227	33160
23560	23563	21534	21535	20489	21541	23553	23589	23713	23716	30984	28567	33164
14165	14166	13372	13376	12825	13390	14084	14189	14191	14197	34314	34355	33225
8450	8451	8192	8193	6192	8203	8206	8457	8458	8459	34710	34359	33228
24576	24578	23575	23580	21536	23583	23584	24590	24640	24641	31761	30995	33229
33265	34363	34481	33166	33274	24762	24769	25736	25738	25739	34624	34625	34306
14209	14210	14167	14178	13387	14181	14183	14226	14228	14231	34488	34493	34356
8464	8465	8452	8453	8202	8455	8456	8545	8703	8704	36906	34361	34362
24643	24651	24579	24580	23582	24587	24589	24744	24753	24754	33072	31762	34485
5243	5244	5235	5236	5227	5238	5239	5250	5251	5252	34627	34631	34496
5489	5496	5339	5340	5316	5364	5376	5507	5508	5509	34564	34360	34620
24756	24758	24652	24703	24581	24714	24742	24816	25605	25614	33159	33073	34626
5253	5254	5245	5246	5237	5248	5249	5279	5280	5281	34691	34693	34632
5511	5512	5497	5500	5352	5504	5505	5573	5574	5575	34656	34584	34680
5744	5749	5717	5722	5621	5726	5731	6014	6149	6150	34621	34688	34690
5302	5304	5255	5256	5247	5267	5268	5320	5327	5328	34712	34716	34695
5576	5578	5520	5523	5503	5525	5551	5704	5707	5708	34697	34664	34701
6156	6159	5843	5907	5725	6012	6013	6172	6173	6174	34682	34706	34711
5329	5338	5314	5315	5266	5317	5319	5388	5400	5488	36918	36978	34725
5713	5716	5582	5586	5524	5698	5699	5735	5740	5743	34732	34700	36728
6175	6177	6166	6168	6011	6170	6171	6184	6185	6186	34704	36905	36909
6187	6188	6179	6180	6169	6182	6183	6197	6198	6199	36864	36984	36989
27961	28271	28272	27962									

Table 11-3: The examined diagnostic codes

Row	Diagnostic code	Definition
1	1000	Unknown
2	1417	LZB Störschalter betätigt
3	3984	ASG1 freigegeben
4	3985	ASG2 freigegeben
5	3995	Lokgeschwindigkeit größer 3 kmh
6	4004	Führerraum 2 ein
7	4014	Fahr- oder Bremssperre FABR
8	4081	ZSG1 ist Master
9	5240	CAN1 Lebenszeichen SP3 F2
10	6172	IO-Modul 25A222 (DX7G2) defekt
11	8921	Bahnverwaltung SBB gewählt
12	9999	Nothalt empfangen
13	11272	Gerätetemperatur zu hoch Sensor 1

Table 11-4: The breakdown codes

Row	Diagnostic code	Definition
1	13345	WR Phase R Rückmeldeverzug
2	13346	WR Phase S Rückmeldeverzug
3	13347	WR Phase T Rückmeldeverzug
4	13377	Durchzündung durch Überstrom Phase R
5	13378	Durchzündung durch Überstrom Phase S
6	13379	Durchzündung durch Überstrom Phase T
7	13382	ZK-Spg Wandler1 WR1 zu groß,Durchzündung
8	13383	ZK-Spg Wandler1 WR2 zu groß,Durchzündung
9	13571	Durchzündung durch dIdt Phase R
10	13572	Durchzündung durch dIdt Phase S
11	13573	Durchzündung durch dIdt Phase T
12	14115	4QS Phase U Rückmeldeverzug
13	14116	4QS Phase V Rückmeldeverzug
14	14117	4QS Phase W Rückmeldeverzug
15	14118	4QS Phase X Rückmeldeverzug
16	14139	ZK-Spg Wandler1 4QS zu groß,Durchzündung
17	14140	ZK-Spg Wandler2 4QS zu groß,Durchzündung

18	14141	Durchzündung durch Phase U Überstrom
19	14142	Durchzündung durch Phase V Überstrom
20	14153	Durchzündung durch dIdt Phase U
21	14154	Durchzündung durch dIdt Phase V
22	14168	Durchzündung durch dIdt Phase W
23	14169	Durchzündung durch dIdt Phase X
24	14170	Durchzündung durch Phase W Überstrom
25	14171	Durchzündung durch Phase X Überstrom
26	17441	WR Phase R Rückmeldeverzug
27	17442	WR Phase S Rückmeldeverzug
28	17443	WR Phase T Rückmeldeverzug
29	17473	Durchzündung durch Überstrom Phase R
30	17474	Durchzündung durch Überstrom Phase S
31	17475	Durchzündung durch Überstrom Phase T
32	17478	ZK-Spg Wandler1 WR1 zu groß,Durchzündung
33	17479	ZK-Spg Wandler1 WR2 zu groß,Durchzündung
34	17667	Durchzündung durch dIdt Phase R
35	17668	Durchzündung durch dIdt Phase S
36	17669	Durchzündung durch dIdt Phase T
37	18211	4QS Phase U Rückmeldeverzug
38	18212	4QS Phase V Rückmeldeverzug

39	18213	4QS Phase W Rückmeldeverzug
40	18214	4QS Phase X Rückmeldeverzug
41	18235	ZK-Spg Wandler1 4QS zu groß,Durchzündung
42	18236	ZK-Spg Wandler2 4QS zu groß,Durchzündung
43	18237	Durchzündung durch Phase U Überstrom
44	18238	Durchzündung durch Phase V Überstrom
45	18249	Durchzündung durch dIdt Phase U
46	18250	Durchzündung durch dIdt Phase V
47	18264	Durchzündung durch dIdt Phase W
48	18265	Durchzündung durch dIdt Phase X
49	18266	Durchzündung durch Phase W Überstrom
50	18267	Durchzündung durch Phase X Überstrom