Analysis of sequences of machine states

Martin Stimpfl

May 5, 2010

Analysis of sequences of machine states

Master's Thesis

at

Graz University of Technology

submitted by

Martin Stimpfl

Knowledge Management Institute (KMI),
Graz University of Technology
A-8010 Graz, Austria

May 5, 2010

©Copyright 2010 by Martin Stimpfl

Advisor: Dipl.-Ing. Dr.techn. Michael Granitzer

Analyse von Maschinenzustandssequenzen

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Martin Stimpfl

Institut für Wissensmanagement (IWM),
Technische Universität Graz
A-8010 Graz

May 5, 2010

©Copyright 2010, Martin Stimpfl

Diese Arbeit ist in englischer Sprache verfasst.

Betreuer: Dipl.-Ing. Dr.techn. Michael Granitzer

Abstract

Data Mining is a keyword for methods and algorithms which are more and more evolving and raising expectations in many fields of information technology nowadays. Computer-aided ”mining” for complex correlations in huge bulks of data is a topic of particular interest for a software company from Graz being engaged in data analysis for manufacturing plants.

This thesis is about a first scenario from this manufacturing area: the recognition of patterns in the history of states of an arbitrary manufacturing machine taken over time. There is nearly no information available but the states themselves and their duration. The question therefore is if there exist significant correlations between the states of the machine.

For this analysis the data's sequential nature allows two different approaches: one using classical techniques from the field of supervised classification, the other one applying methods from the field of sequence- or episode-mining. First this thesis introduces different applicable methods from both fields. Afterwards it presents results from conducted experiments showing that a recognition of meaningful patterns is possible.

Kurzfassung

Data Mining ist ein Schlagwort, in das heutzutage viele Erwartungen im Bereich der Informatik gesteckt werden. Das maschinen-unterstützte "Graben" und "Fördern" von komplexen Zusammenhängen in großen Datenbeständen ist auch von Interesse für eine Grazer Softwarefirma, die sich mit Datenanalyse im Bereich der Produktion beschäftigt.

Diese Arbeit bearbeitet ein erstes Szenario dem sich die Firma widmen möchte: das Erkennen von Mustern in der Zustandshistorie einer produzierenden Maschine. Außer der Sequenz der Zustände und deren Dauer steht nur wenig Information zur Verfügung, die verwendet werden kann. Die Frage ist daher, ob zwischen den einzelnen Maschinenzuständen signifikante Korrelationen bestehen.

Die sequentielle Natur der Daten bedingt zwei unterschiedliche Zugänge der Bearbeitung: zum einen bieten sich klassische Methoden der Klassifikation, zum anderen Methoden des so genannten Sequence- beziehungsweise des Episode-Mining an. Diese Arbeit präsentiert zunächst verschiedene mögliche Ansätze aus beiden Gebieten, um danach eine Methode aufzugreifen und Ergebnisse erster Versuche zu liefern. Diese Versuchen sollen zeigen, dass ein Auffinden von Mustern möglich ist.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ………………………………………………..
date (signature)

Danksagung

Ich möchte mich auf diesem Weg aus tiefstem Herzen bei meinen Eltern bedanken, die mir durch ihre Unterstützung meinen bisherig sorgenfreien und zielgerichteten Lebensweg aufopfernd und liebevoll ermöglicht haben. Es lässt sich hier nicht zu Papier bringen, wie dankbar ich für die Chancen und Möglichkeiten bin, die sich mir durch ihren Einsatz und ihre Mühen aufgetan haben und die mich voller Spannung zukünftigen Herausforderungen entgegen blicken lassen.

Großer Dank gilt auch dem Betreuer dieser Arbeit Dr.techn. Michael Granitzer, welcher sich äußerst hilfsbereit und mit großem Engagement dem Thema dieser Arbeit gewidmet hat. Ihm ist es zu verdanken, dass ein Aufgreifen des Themas im Zuge der Dilpomarbeit und eine komplikationslose Abarbeitung möglich waren. Seine fachliche Unterstützung war für diese Arbeit von essentieller Bedeutung.

Besonders möchte ich meiner Freundin Katharina danken, deren Unterstützung und Begleitung die letzten Jahre meines Studiums bereicherten und zu einer ganz besonderen und wertvollen Zeit für mich machten.

Meinen Dank spreche ich auch allen Menschen aus, die mich in verschiedenen Bereichen meines Lebens begleiten und begleitet haben.

Contents

List of Figures

List of Tables

1 Introduction

Data Mining is a popular term nowadays. Computational power combined with more and more approved methods and algorithms make it possible that computers help humans analysing and understanding bulks of data by searching and identifying interesting patterns. The purpose of Data Mining is gathering information from sets of data reaching sizes that make it hard for humans to interpret it on one's own. Facing how much data is generated day by day in different scopes of our every day life (see [GRC$^+$07] for details) it seems these techniques get indispensable for understanding the information documented by data.

Working for a software solutions company the question arose if the application of state-of-the-art Data-Mining-methods in data sets generated by the firm's software tools would help to unearth information as well as knowledge not being apparent at the moment. Information about manufacturing machines is recorded and filed in these data sets. Analysing these sets is clearly one of the main issues the company wants to enable its customers. Therefore the company's software tools already provide some kind of visual Data-Mining possibilities. What is being assumed is that there exist patterns in this data whose discovery is not covered by the existing methods yet.

Amongst others the states a machine took over time are recorded. It is of interest if there exist correlations between these states and if a particular state entails other ones. The answering of this question is the topic of this thesis.

This first chapter presents the thesis' purpose, its motivation and backgrounds in terms of analysing a sequence of machine data. At the beginning the company is introduced in more detail. Thereby the working field which made this work evolve and the expected results are inducted. In the end notations are listed which are apparent throughout the hole work.

1.1 Application scenario

The company which made the thesis possible is named GAMED® and is located in Graz, Austria. At this moment GAMED holds two offices namely, the headquarters in

Graz and another-one in Peking, China whereas 17 employees are engaged in Graz and around 8 in Peking. Since 1985 the company's fields of work include several software solutions for manufacturing services, amongst others an efficiency- and productivity-analysis tool called *OEE-Analyser*^{TM}.

The OEE-Analyser$^{\mathrm{TM}}$investigates if manufacturing machines work in time with the effectiveness and quality they are supposed to do and provides the user with useful visual analysis tools by applying the well-known OEE(overall equipment effectiveness)-method forming part of the maintenance process TPM (total productive maintenance) [Nak89].

According to the OEE-method a machine's performance is evaluated by four factors:

1. the loading,
2. the availability,
3. the performance and
4. the quality

The loading is the time the machine is to work in proportion to the total calendar time. The availability says in what time the machine actually worked and so expresses unforeseen downtimes. The ratio between the target-speed of the machine and the speed it actually worked at is given by the performance. At last the quality states the scrap's proportion to the total number of units produced.

1.2 Goal of this thesis

Important for the OEE-evaluation is keeping track of states the machine takes over time. These states document if a machine is producing, if it is broken, maintained or something different. The analysis of the states' history is therefore a crucial part of interpreting how well a machine performed, if it met the targets or underachieved.

According to this it is of high interest if there exist repeating patterns in the time-line of the machine states, such as if certain machine states lead to another ones. Such knowledge is of high value when operating a machine and analysing its performance. Different machines could be compared as well according to pattern occurrences. Furthermore a system equipped with this information of correlations between states may act as an early-warning-system for major machine breaks if according patterns exist.

How many data there is generated varies from plant to plant but to give a feeling that computer-aided methods are inevitable: there are around 150 lines of data generated per day for one machine that is investigated in the experiments' section 6. In the machine's plant there are around 30 other machines installed. A system searching for patterns and summarizing it is the main idea behind this work. In section 2 the problem is formally defined and the detailed expectations of this work are listed. Chapter 3 and 4 theoretically provide and introduce possible methods and algorithms whereas in chapter 6 first concepts are applied and evaluated.

1.3 Notation

This section gives a brief overview of notations used throughout this thesis.

$\overrightarrow{x}$ denotes a column vector.

$\overrightarrow{x}^T$ denotes a row vector.

$(x_1, ..., x_K)$ parenthesis indicate the variables $x_1, ..., x_K$ to be dimensions of a vector $\overrightarrow{x}$ or $\overrightarrow{x}^T$ where K is the size of the vector.

$p(x)$ denotes a probability distribution over some variable x.

$p(x|y)$ denotes a probability distribution over some variable x conditioned on the fact that variable y has already taken a fixed value.

$\overline{I}$ an over-line indicates an arbitrary variable I being an item set.

$\{i_1, ..., i_K\}$ braces indicate the variables $i_1, ..., i_K$ to form a set $\overline{I}$ of variables where K is the size of $\overline{I}$. $i_m \neq i_n$ holds for every $1 \leq m \leq K$, $1 \leq n \leq K$ and $m \neq n$.

$\widehat{S}$ a hat indicates an arbitrary variable S being a sequence which is defined later on.

$\langle s_1, ..., s_K \rangle$ indicate the variable $s_1, ..., s_K$ being sequence items of a sequence $\widehat{S}$ where K is the length of a sequence.

$\mathfrak{S}$ stands for the history of the states of a machine and indicates the sequence we want to analyse and understand.

2 Problem Statement

The purpose of this thesis is to unearth correlations in between states of manufacturing machines. Before possible techniques addressing this problem can be presented one has to face the problem in more detail. Therefore this section introduces the formal problem statement associated with the outlined application scenario. The history of states a machine takes over time can be understood as a sequence of events taking place one after another in the time-line. Therefore this section defines a sequence. Furthermore it has to made clear what information one expects to gather from that sequence.

Let the history of a machine's states be denoted by $\mathfrak{S}$. A machine can take one particular state out of several possible at one moment. This set of possible states is written as $\overline{I} = \{i_1, i_2, ..., i_H\}$ where $i_h \in \overline{I}$ and H is the number of different states. A sequence $\widehat{S} = \langle s_1, s_2, ...s_L\rangle$ is an ordered list where L denotes its length and $s_l \in \widehat{S}$ represents the state of a machine at time l. $s_1, s_2, ...s_L$ are referred to as sequence items. $\mathfrak{S}$ can therefore be understood as such a sequence. Additionally to the sequence there exist duration information as well as date information and shift information of every sequence item s_l in $\mathfrak{S}$ whereby this correlation is not defined at the moment. A sequence $\widehat{S_t^K} = \langle s_t, s_{t+1}, ..., s_{t+K-1}\rangle$ is a subsequence of $\widehat{S}$ denoted by $\widehat{S_t^K} \sqsubseteq \widehat{S}$ where t is the start point of $\widehat{S_t^K}$ in $\widehat{S}$ and K the subsequence's length. One can say that subsequence $\widehat{S_t^K}$ is contained by $\widehat{S}$.

Given no further information but the sequence $\widehat{S} = \langle s_1, s_2, ...s_L\rangle$ and the additional information in $\mathfrak{S}$ we want to know if there exist repeating patterns in $\widehat{S}$. To be more precise it is of interest whether machine state i_h can be predicted by the information provided by it's predecessors in $\widehat{S}$ and so if there exists a correlation between several i_h's. One could generalize this problem to getting a deeper understanding of the sequence at hand and finding regularities if there exist some. This is to be supported and facilitated by state-of-the-art computer algorithms and heuristics.

Several machine-aided methods exist which tackle the problem from particular angles. Some yield solutions which are more or less simple to understand and to read for humans. Others may be difficult to be interpreted but may lead to more sophisticated results and performances for prediction. It has to be stated that in the end the focus has to lie on human interpretable solutions as the understanding of the sequence of machine states is essential. Nevertheless methods yielding more complicated but sophisticated solutions can be used for evaluation if prediction is possible at all. One can assume that if these methods don't manage to construct proper solutions the simpler ones won't either. So the entire work may be grouped into three parts:

1. presenting possible methods being able to tackle the problem of finding correlation between several machine states i_h,

2. evaluating if a prediction of a particular machine state i_h is possible by using the information of such correlations and

3. if yes, understanding the correlations, what means which machine states are responsible for predicting what others and the order between them.

Problem (1) is addressed by chapters 3 and 4 where probabilistic and non-probabilistic methods are introduced and explained. (2) is evaluated in the experiments' section in chapter 6. Although the methods for solving problem (3) are theoretically presented its realisation is not part of this thesis and may be an objective for further works.

3 Classification

This chapter introduces the concepts of probabilistic and non-probabilistic classifier. It is based on the work of [Bis07], [KT07] and [SM06] and thereby gives an summary over the methods.

3.1 Decision Theory

Let us consider the problem again. We want to know if there exist patterns in the sequence of machine states $\widehat{S}$ in $\mathfrak{S}$ such that an item i_h can be predicted by using the information provided by i_h's predecessors in $\widehat{S}$. This problem can be generalized to a *decision* or *classification* problem as it is called in the science field of *Machine Learning*, *Pattern Recognition* and *Data Mining* [Bis07] [KT07].

Classification is the mapping of an input vector $\overrightarrow{x} = (x_1, x_2, ...x_N)$ on a target *label* y which is a discrete variable. Each $x_n \in \overrightarrow{x}$ is called a *feature* which can take discrete as well as continuous and binary values. Methods performing such a mapping are called *classifiers* and try to find representative feature value ranges for each label value. The mapping of $\overrightarrow{x}$ to a continuous variable y is called *regression* problem and is not further discussed here (see [Bis07] for further information). Figure 3.1 illustrates a simple classification task in a two-dimensional feature-space.

The x's and y's are input data points whereas x and y indicates their class-membership. The border which separates the x's from the y's is called *decision boundary.*

When $\overrightarrow{x}$ represents the input vector and y the target label in general there exist three ways how to tackle the problem of finding this decision boundary:

- The *generative* approach where one tries to model the joint probability distribution of the variables $p(y, \overrightarrow{x})$.
- The *discriminative* approach where the conditioned probability distribution $p(y|\overrightarrow{x})$ is directly modelled.
- A non probabilistic, direct mapping by constructing a function where it's output directly determines the class label.

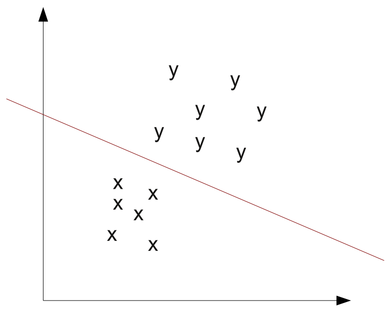

Figure 3.1: A simple classification task where the x's are data points of class X and the y's those of class Y. The red line forms a possible decision boundary.

The advantage of probabilistic methods compared to the direct mapping is that one gets an insight how confident a decision for a value of y can be made compared to other values by considering the probabilities. It can be considered as a drawback that in direct mapping solutions one only gets the information if a label value was chosen or not.

Different methods exist which all model these probability distributions or mappings in slightly different ways as we shall see shortly. Some construct more or less simple models for classification what may lead to poor results in classifying complex data. Others generate highly sophisticated models which may yield good results on complex data but are hard to interpret and to understand for humans or computational infeasible. As the understanding of the sequence at hand is important in this work those sophisticated algorithms may be used rather for evaluation if patterns for classification in the data exist. In the end understandable and interpretable models about the data have to be constructed.

Classification is usually done in two phases called *Training-* and *Test-*phase. In the former the algorithms introduced in the next chapters construct *models* of the information provided by the data. The latter exists for testing how well the constructed model *generalizes* (or how well it performs) on data the algorithm has not seen for training. Therefore the data set is split into a training- and a test-data set. Fitting the algorithm's model to the data is then performed on the training set whereas the evaluation of the model's performance is done on the test set. There exist several ways to accomplish this. One is to simply cut the data set in two at a certain point. Another one is called *n-fold-cross-validation* where the whole data set is cut into n pieces and in n runs every piece is considered once as test set while the merged remaining pieces form the training set. The outcome of different runs are then combined to one. [WF02] gives an introduction to this topic.

In the following an overview over such classification algorithms and methods is given as well as a slightly different approach of handling this task called *Sequence* or *Episode* Mining in chapter 4. In chapter 6 the outcome of first experiments using one of the presented methods is summarized. It should make clear that the applied algorithm did find correlations between several machine states what makes the prediction of some of them possible.

3.2 Naive Bayes Classifier

The first classification method presented is called *Naive Bayes Classifier*. It belongs to the family of generative approaches by trying to construct an assumed probability distribution $p(\overrightarrow{x}, y)$ over the data.

The probability distribution $p(y|\overrightarrow{x})$ which is to be modelled in the end can be expressed according to Bayes' theorem by

$$p(y|\overrightarrow{x}) = \frac{p(\overrightarrow{x}|y) * p(y)}{p(\overrightarrow{x})}. \tag{3.1}$$

As according to [KT07] and [WF02] the denominator $p(\overrightarrow{x})$ is just a normalization constant and can therefore be omitted for classification it is that

$$p(y|\overrightarrow{x}) \propto p(\overrightarrow{x}|y) * p(y) = p(\overrightarrow{x}, y) \tag{3.2}$$

according to the sum rule of probability theory [1].

When $\overrightarrow{x} = (x_1, x_2, ..., x_N)$ we can rewrite the probability distribution:

$$p(y|\overrightarrow{x}) \propto p(y, x_N, x_{N-1}, x_{N-2}, ..., x_1). \tag{3.3}$$

According to the sum rule this can be expressed by

$$p(y|\overrightarrow{x}) \propto p(y)*p(x_1|y)*p(x_2|x_1, y)*..*p(x_N|x_{N-1}, ..., x_1, y) = p(y)\prod_{n=1}^{m} p(x_n|x_{n-1}, ..., x_1, y). \tag{3.4}$$

As it is practically impossible to model all dependencies between the input dimensions $x_n \in \overrightarrow{x}$ it is assumed that the dimensions are independent of each other conditioned on y such that $p(x_n|x_{n-1}, ..., x_1, y) = p(x_n|y)$. Equation 3.4 therefore simplifies

[1] $p(X, Y) = p(Y|X) * p(X)$ [Bis07]

to

$$p(y|\overrightarrow{x}) \propto p(y) \prod_{i=1}^{m} p(x_n|y) \tag{3.5}$$

which forms the Naive Bayes classifier.

Because of the conditional independence assumption this classifier builds up a simplified model about the data being inspected. Even though Naive Bayes performs surprisingly well on many real world classification tasks (see for instance [KM02] for email classification) it performs worse than logistic regression on average and gives only poor probability estimates according to [SM06].

To apply the Naive Bayes classifier one has to model the probability distribution of the label $p(y)$ and the probability distribution of the input vector conditioned on the label $p(\overrightarrow{x}|y)$ (this means the probability distribution of $\overrightarrow{x}$ is modelled for each class independently) to achieve $p(y|\overrightarrow{x})$ according to equation 3.1. Alternatively one could model $p(\overrightarrow{x}, y)$ directly. All the probability distributions have to be constructed under the Naive Bayes assumption of conditional independence between the features x_n. The decision boundaries between two classes will occur in this method implicitly. They happen in the input space of $\overrightarrow{x}$ where probabilities of different classes are equal.

As for our mining task of$\mathfrak{S}$ we assume that a machine state i_h somehow emerge from it's predecessors in $\mathfrak{S}$ and as the predecessors are i_h's themselves we imply that there exists a strong correlation between the features x_n. As there exist algorithms which take this correlation explicitly into account (algorithms such as the Hidden Markov Models in section 3.6 shall be introduced shortly) the simple Naive Bayes classifier will not be the first choice in the experiments in chapter 6. Nevertheless since it forms the basis of section 3.6 it is of essential interest.

3.3 Logistic Regression and Maximum Entropy Model

Section 3.2 explained the Naive Bayes Classifier which falls into the family of *generative models*. That is because one has to model the probability distribution $p(\overrightarrow{x}, y)$ for its application and could sample synthetic data points from it. However there is no real reason to model the entire joint probability distribution of $p(\overrightarrow{x}, y)$ when one is only interested in the conditional distribution $p(y|\overrightarrow{x})$ for a sole classification task.

For *logistic regression* one tries to directly construct the conditional distribution $p(y|\overrightarrow{x})$ by assuming the distribution's form takes the logistic sigmoid function (or the softmax function when y takes more than two values). This is illustrated by Figure 3.2.

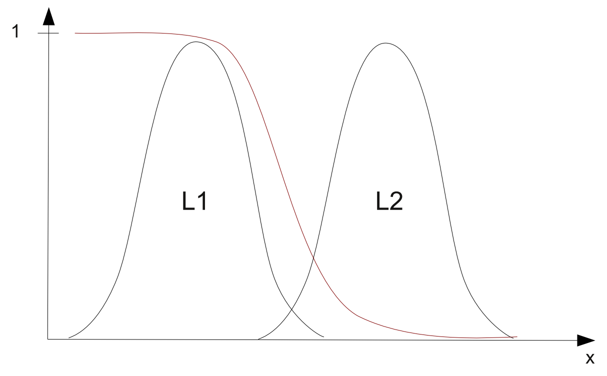

Figure 3.2: Two Gaussian distributions of two class label values L1 and L2. The red line is the conditional probability of $p(L1|x)$ which takes the form of the logistic sigmoid function.

However not for every conditional probability distribution $p(y|\overrightarrow{x})$ a logistic sigmoid function can be assumed so [Bis07] introduces the *probit regression* which acts as a threshold model for more general Gaussian distributions and finally introduces the *Laplace Approximation* for approximating unknown probability distributions to enable a Bayesian treating of the problem.

[KT07] explains the *Maximum Entropy Model* which is based on the *Principle of Maximum Entropy* [Jay57]. This says that if there just exists incomplete information about a probability distribution the only unbiased assumption which can be made about the distribution is a probability distribution "which is as uniform as possible given the available information" [Jay57]. As one cannot be sure if the data excerpt to be analysed contains the whole information for the variables' entire probability distributions this means the distribution to be chosen is the one which maximizes the entropy fulfilling the constraints provided by the excerpt. For $p(y|\overrightarrow{x})$ the conditional entropy model

$$H(y|\overrightarrow{x}) = -\sum_{x\in\chi, y\in\Gamma} p(y, \overrightarrow{x}) \log p(y|\overrightarrow{x}) \tag{3.6}$$

has to be applied.

χ contains all the possible input values whereas Γ contains all possible output values which means all possible combinations of input and output values are used for $H(y|\overrightarrow{x})$ and not just those which actually appear in the data. The problem to be solved is to find the maximal entropy model $H(y|\overrightarrow{x})$ which is consistent with the data's constraints.

Finding the maximum for $H(y|\overrightarrow{x})$ is achieved by setting the derivative of $H(y|\overrightarrow{x})$ fulfilling the constraints to zero. The equation to be deviated is formulated with the help of Lagrange-multipliers and is expressed as follows:

$$\Lambda(p, \overrightarrow{\lambda}) = H(y|\overrightarrow{x}) + \sum_{i}^{m} \lambda_i \big(E(f_i) - E'(f_i)\big) + \lambda_{m+1} \Bigg(\sum_{y \in \gamma} p(y|\overrightarrow{x} - 1) \Bigg). \tag{3.7}$$

The first term in equation 3.7 is the maximum entropy model which is to be maximized by setting it's derivate to zero. The first addend represents the constraints provided by the data. $E'(f_i)$ is the the probability distribution $p'(y, \overrightarrow{x})$ as it appears in the data while $E(f_i)$ is the model probability distribution. The f_i's are the features of the data. They are binary functions $f_i(y, \overrightarrow{x}) \in 0, 1$ where $1 \leq i \leq I$ where I is the number of possible combinations of input values and labels. When N is the number of instances in the trainings-dataset τ the probability distribution $E'(f_i)$ is calculated by

$$E'(f_i) = \frac{1}{N} \sum_{(x,y) \in \tau} f_i(x, y). \tag{3.8}$$

Each f_i stands for a possible combination of input value and label value. Equation 3.8 does nothing else than counting how often a specific combination of those values appears in the trainings data and setting it in relation to the number of training instances.

According to [KT07] the model probability distribution which is searched during the maximization is defined as follows:

$$E(f_i) \approx \frac{1}{N} \sum_{x \in \tau} \sum_{y \in \Gamma} p(y|x) f_i(x, y). \tag{3.9}$$

As the model distribution has to capture all constraints given by the distribution gathered from the data the first addend in equation 3.7 emerges from the constraint $E(f_i) = E'(f_i)$. The second addend in equation 3.7 ensures the result is a proper probability distribution. This is achieved by setting $\sum_{y \in \Gamma} p(y|x) = 1$ for all x and $p(y|x) \geq 0$ for all x,y.

The derivation the equation 3.7 and equalizing with zero is discussed in detail in [KT07]. In the end the Maximum Entropy model is expressed as:

$$p(y|\overrightarrow{x}) = \frac{1}{Z(x)} \exp \Bigg(\sum_{i=1}^{m} \lambda_i f_i(\overrightarrow{x}, y) \Bigg), \tag{3.10}$$

where

$$Z(x) = \sum_{y \in \Gamma} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(\overrightarrow{x}, y) \right). \tag{3.11}$$

3.4 Support Vector Machines

The Support Vector Machines (SVM) [CV95] fall into the family of the direct-mapping-approaches. Originally they do not produce a probabilistic output although there exist approaches of extending SVMs in such a way [CL01]. Training SVMs to get them solving a classification task on a given dataset means to solve an optimization problem of where to place the decision boundary in the feature space and to find a proper shape for the boundary so that it fits the data well.

3.4.1 SVM for the separable case

The approach of SVM's in a separable case is to place the boundary in between the data-points of different labels. In the basic version of SVMs these boundaries are considered to be linear. As there are indefinite possibilities of how to place that linear boundary the SVM places it exactly in the middle between the data-points. This is equivalent to placing the boundary where it's nearest data points of each label have maximum distance. Figure 3.3 illustrates this for a two-class classification problem. The problem can be formulated as *maximizing the margin* which is the region between the boundary and it's nearest data points of each label.

The decision boundary indicated by the dashed line in Figure 3.3 can be formulated as

$$y(\overrightarrow{x}) = \overrightarrow{w}^T \phi(\overrightarrow{x}) + b \tag{3.12}$$

where $\phi(\overrightarrow{x})$ denotes a feature-space transformation of input vector $\overrightarrow{x}$. A feature space transformation $\phi(\overrightarrow{x})$ is a non-linear transformation of input vector $\overrightarrow{x}$ which might make linear separation of data possible in feature space when it is not possible in the original space of $\overrightarrow{x}$ what is discussed in detail later. For the linear separable case there is simply $\phi(\overrightarrow{x}) = \overrightarrow{x}$. $\overrightarrow{w}$ is the vector normal to the decision boundary which determines it's orientation while b is the so called *threshold* value and is responsible for the offset of the boundary to the origin. For the two-class case the sign of the outcome indicates if the data point $\overrightarrow{x}$ belongs to one or the other class. The outcome's absolute value is the distance of the data point to the decision boundary in relation to the norm of vector $\overrightarrow{w}$. The outcome divided by $\overrightarrow{w}$'s norm gives the actual distance. The goal of fitting the SVM to a given dataset is to determine the proper values of $\overrightarrow{w}$ and b.

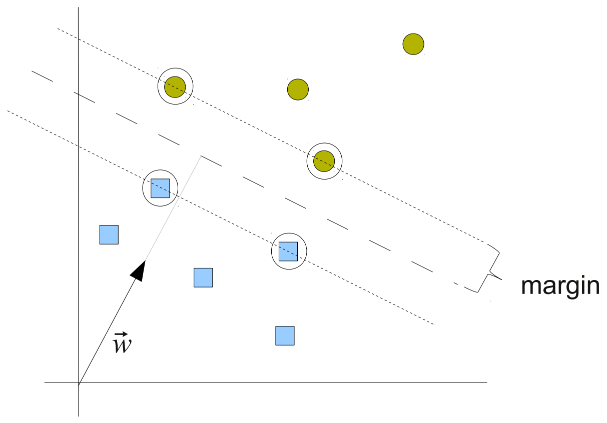

Figure 3.3: Decision boundary determined by a SVM for a linear separable case. The area between the nearest data points of each label and the boundary is called margin which is maximized when choosing the right boundary. The nearest data points are called *Support Vectors* which are surrounded by circles in the figure.

The nearest data points in figure 3.12 are called *support vectors*. When fitting the SVM to a dataset one chooses $\overrightarrow{w}$ and b such that $y(\overrightarrow{x}) = 1$ for support vectors belonging to one class and $y(\overrightarrow{x}) = -1$ for support vectors belonging to the other class (all other vectors $\overrightarrow{x}$ therefore have a value $y(\overrightarrow{x}) > 1$). When t_n signifies the actual class of point $\overrightarrow{x}$ one can write this constraint as

$$y(\overrightarrow{x}) * t_n \geq 1 \tag{3.13}$$

what holds for every point in a data set if it is correctly classified by equation 3.12. As the margin in figure 3.3 is then given by $\frac{2}{||\overrightarrow{w}||_2}$ to maximize this means to minimize $\frac{1}{2}||\overrightarrow{w}||_2^2$.

This can be done by using Lagrange multipliers $\overrightarrow{\lambda} = (\lambda_1..\lambda_N)$ where N and $\lambda_n \in \overrightarrow{\lambda}$ is the size of the data set what leads to the following equation

$$L(\overrightarrow{w}, b, \overrightarrow{\lambda}) = \frac{1}{2}||\overrightarrow{x}||_2^2 - \sum_{n=1}^{N} \lambda_n \big(t_n(\overrightarrow{w}^T \phi(\overrightarrow{x}) + b) - 1\big). \tag{3.14}$$

Setting the derivate of equation 3.14 with respect to $\overrightarrow{w}$ and b to zero leads to

$$\overrightarrow{w} = \sum_{n=1}^{N} \lambda_n \phi_n(\overrightarrow{x}) \tag{3.15}$$

and

$$0 = \sum_{n=1}^{N} \lambda_n t_n. \tag{3.16}$$

Substituting these two results back into 3.14 gives the dual representation of the problem of which a kernel formulation in the form $k(x, x') = \phi(\overrightarrow{x})^T \phi(\overrightarrow{x'})$ arises. As mentioned before the feature space transformation $\phi(\overrightarrow{x})$ allows the space of $\overrightarrow{x}$ to be transformed non-linearly into another feature space. With the help of the kernel function the whole problem can be transformed into a feature space of higher dimension in which the problem might get linear separable when this is not possible in the original space. When transforming back the learned decision hyperplane into the original space this gives a non-linear decision boundary or plane. Such kernels can be designed differently. [Bis07] gives an introduction and examples what common designs exist and how to construct new ones.

When substituting equation 3.15 back into equation 3.12 this gives

$$y(\overrightarrow{x}) = \sum_{n=1}^{N} \lambda_n t_n k(x, x_n) + b. \tag{3.17}$$

This means the value for an input vector $\overrightarrow{x}$ is determined by the values of λ_n and b which have been defined by solving the dual representation of equation 3.14 and the kernel evaluation of $\overrightarrow{x}$ with each x_n multiplied by it's original label. Because the problem has been solved by the help of Lagrange multipliers and constraint 3.13 forms an inequality constraint the solution of this optimization problem satisfy the *Karush-Kuhn-Tucker* (KTT) [Kar39] [KT50] conditions which say

$\lambda_n \geq 0$

$t_n(\overrightarrow{w}^T \phi(\overrightarrow{x}) + b) \geq 0$

$\lambda_n \big(t_n(\overrightarrow{w}^T \phi(\overrightarrow{x}) + b)\big) = 0.$

This means that either $\lambda_n = 0$ or $t_n(\overrightarrow{w}^T \phi(\overrightarrow{x}) + b) = 1$. So data points for which $\lambda_n = 0$ play no role for the evaluation equation 3.17. The remaining have $y(\overrightarrow{x}) = 1$ and are hence the support vectors. According to this, the evaluation of what class new data points $\overrightarrow{x}$ belong to depends on a kernel transformation done with $\overrightarrow{x}$ and the support vectors. The remaining vectors not being support vectors can be omitted for this evaluation.

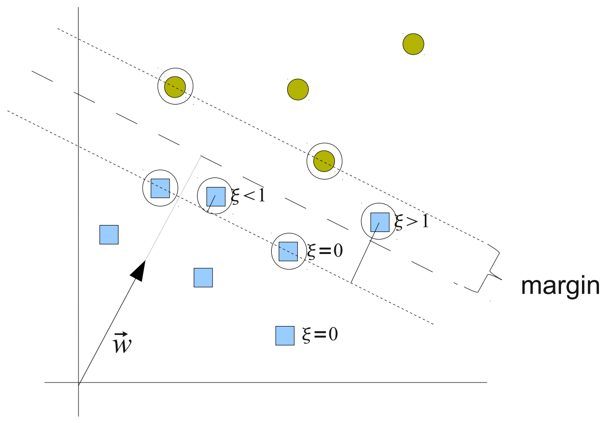

Figure 3.4: Decision boundary determined by a SVM for a not linear separable case.

3.4.2 SVM for the non-separable case

In practical applications data points from different labels are not entirely separable and are therefore somehow overlapping each other. The concepts introduced yet only allow the data to be separable properly so they have to be modified slightly. To accomplish this so-called *slack variables* ξ are introduced. To goal is to allow misclassifications but to softly penalize it according to the distance from the decision boundary. The slack variables ξ are assigned to every data point and are defined to be $\xi = 0$ for correctly classified data points and $\xi = |t_n - y(x_n)|$ for misclassified ones. Figure 3.4 illustrates this.

The constraint for placing the decision boundary 3.13 is then reformulated to

$$y(\overrightarrow{x_n}) * t_n \geq 1 - \xi_n \tag{3.18}$$

for every data point x_n from the dataset and ξ_n being it's slack variable. Instead of minimizing solely $\frac{1}{2}||\overrightarrow{x}||_2^2$ one now minimizes $C\sum_{n=1}^{N}\xi_n + \frac{1}{2}||\overrightarrow{x}||_2^2$ during optimization where $C > 0$ is a trade-off parameter between the maximum margin size and the penalty gathered by ξ. The equation to be optimized formulated again with the help of Lagrange multiplies $\overrightarrow{\lambda} = (\lambda_1..\lambda_N)$ and $\overrightarrow{\mu} = (\mu_1..\mu_N)$ where N denotes the size of the data set and $\mu_n \in \overrightarrow{\mu}$ is therefore

$$L(\overrightarrow{w}, b, \overrightarrow{\lambda}, \overrightarrow{\mu}) = \frac{1}{2}||\overrightarrow{x}||_2^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\lambda_n\big(t_n(\overrightarrow{w}^T\phi(\overrightarrow{x}) + b) - 1\big) - \sum_{n=1}^{N}\mu_n\xi_n \tag{3.19}$$

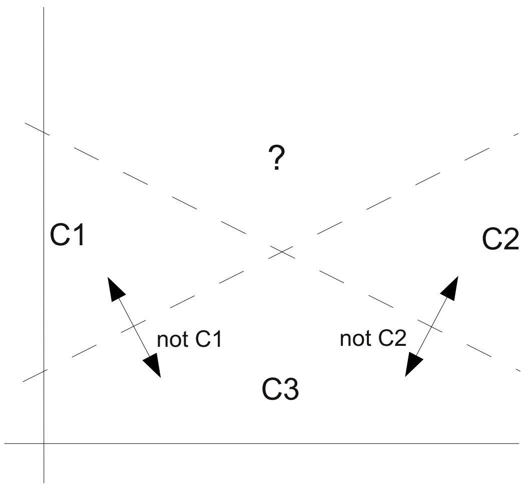

Figure 3.5: Decision boundaries illustrating the *one-versus-the-rest* approach for three classes what leads to ambiguous regions. The dashed lines symbolize decision boundaries.

where $\sum_{n=1}^{N} \mu_n \xi_n$ makes sure that $\xi_n \geq 0$ for all $1 \leq n \leq N$.

3.4.3 SVM for the multi-class case

Only the two-class classification has been discussed so far. The question is how to formulate the decision boundary defined in equation 3.12 for the $K > 2$ classes problem. The first approach is called *one-versus-the-rest* where for K classes $K - 1$ decision boundaries are introduced each solving a two-class problem.

Figure 3.5 illustrates the problem in the case of three classes what leads to a ambiguous region on the top of the figure. Another approach is named *one-versus-one* where $K(K-1)/2$ decision boundaries are used. Every decision boundary votes for one class and the one class most voted is chosen. As figure 3.6 shows this may lead to ambiguous regions as well.

There exists one possibility which overcomes the problem of these ambiguous regions. It stipulates one linear function of the form

$$y_k(\overrightarrow{x}) = \overrightarrow{w}_k^T \phi(\overrightarrow{x}) + b_k \tag{3.20}$$

for every class $1 \leq k \leq K$ where K denotes the number of classes and assigns class m

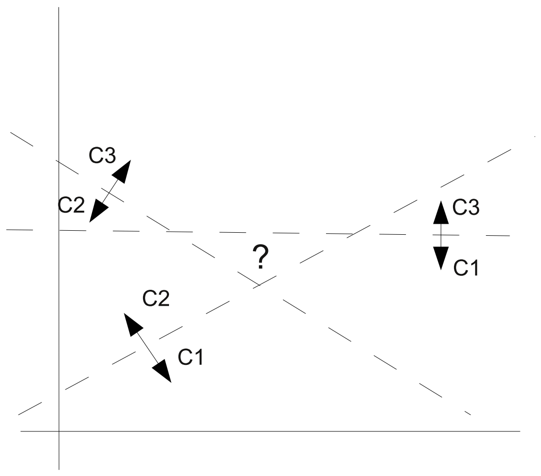

Figure 3.6: Decision boundaries illustrating the *one-versus-one* approach for three classes what leads to ambiguous regions too. The dashed lines symbolize decision boundaries.

if $y_m(\overrightarrow{x}) > y_n(\overrightarrow{x})$ for every $m \neq n$ and $1 \leq m \leq K, 1 \leq n \leq K$.

The decision boundary between two classes m and n then arises from $y_m(\overrightarrow{x}) = y_n(\overrightarrow{x})$ and is defined by

$$0 = (\overrightarrow{w}_m - \overrightarrow{w}_n)^T \phi(\overrightarrow{x}) + (b_m - b_n) \tag{3.21}$$

which takes the same form as the decision boundary for the two-class case (see equation 3.12) and has therefore the same geometric properties.

All these approaches suffer two major limitations for the training of a SVM. First, every decision boundary has to be trained on different tasks what leads to different scales for the outcome of different $y_k(\overrightarrow{x})$. Second, the SVM is trained on imbalanced data sets when only one of K classes is said to have outcome $+1$ while all the others have -1. [Bis07] gives an introduction of how these problems are faced in practical applications.

3.4.4 SVM compared to probabilistic classifiers

Compared to the probabilistic classifiers discussed so far there is a major difference. Probabilistic classifiers build a model based on a given dataset and do not need nothing else for classifying new data but the constructed model. The SVM uses information from the data set directly for it's model building namely the support vectors. There exist several other methods which do that such as the K-Nearest-Neighbour algorithm [Bis07]. This algorithm classifies new data points $\overrightarrow{x}$ according to it's distance to other data points of different labels. For classification the one label is chosen of which K data points are nearest to $\overrightarrow{x}$. This means when implementing this algorithm on a computer the hole trainings-data-set has to be kept in memory for classifying new data points whereas for the SVM only the support vectors have to be saved.

3.5 Graphical Models

Probability distributions can be represented by probabilistic graphical models. There are two kind of representations: the *Conditional Independence Graph* and the *Factorization Graph*. The probability distributions constructed by the Naive Bayes classifier of section 3.2 can be expressed as such a graphical model as well which is shown in figure 3.7. Graphical models lay the basis for the concepts explained in sections 3.6 and 3.7 following this one.

3.5.1 Directed Conditional Independence Graph

The Conditional Independence Graph can be either directed or undirected. Probability variables are represented by the nodes in both the directed and undirected graph. The edges indicate conditional dependence properties between the variables. Figure 3.7 gives an example for both versions of the joint probability distribution of the variables a, b, c, d. The left figure represents a directed graph and can be read as the probability distribution $p(a, b, c, d) = p(a) * p(b) * p(c|a, b) * p(d|c)$. Such a model is also called *Bayesian Network* and can generally be expressed as

$$p(\overrightarrow{x}) = \prod_{x=1}^{N} p(x_n|p_n) \tag{3.22}$$

where $\overrightarrow{x} = (x_1, .., x_N)$ denotes the vector containing all variables $x_n \in \overrightarrow{x}$ and p_n represents x_n's parent nodes (meaning nodes on which x_n is conditioned).

3.5.2 Undirected Conditional Independence Graph

The right figure in figure 3.7 is an undirected conditional independence graph. It is a more general representation than the directed one. Undirected graphs are also called *Markov Random Fields* what foretells some kind of relationship to the probabilistic

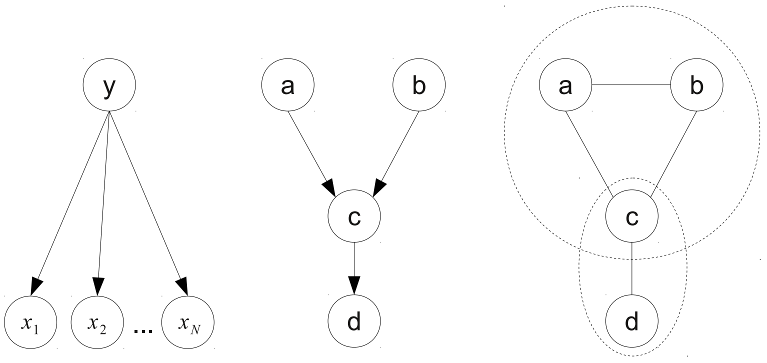

Figure 3.7: Three conditional independence graphs - the left one representing the Naive Bayes assumption of section 3.2 (conditional independence of input variables), the one in the middle representing an arbitrary directed graph and the right one representing its undirected version. The dotted ellipses in the right model indicate maximal cliques.

model described in section 3.7. The joint probability distribution is derived from the *maximal cliques* of the graph which are indicated by shaded ellipses in figure 3.7. A maximal clique is defined as a sub-graph of the original graph where all nodes are interconnected and adding any other node from the original graph would violate this constraint. When a maximal clique is denoted by C and it's variables as x_C the joint probability is expressed by potential functions $\psi_C(x_C)$ over the maximal cliques C written as

$$p(\overrightarrow{x}) = \frac{1}{Z}\prod_C \psi_C(\overrightarrow{x_C}) \tag{3.23}$$

where $\overrightarrow{x} = (x_1, .., x_N)$ is the vector containing all K variables and

$$Z = \sum_x \prod_C \psi_C(\overrightarrow{x_C}). \tag{3.24}$$

Note that the only constraint of the potential functions $\psi_C(x_C)$ is to be larger than zero. They do not have to be of a probabilistic nature. This gives more generality and freedom of choice how the variables in x_C are related but makes the denominator Z necessary so that the outcome of equation 3.23 is actually probabilistic.

The Maximum Entropy Classifier described in section 3.3 can be expressed by such potential functions:

$$p(y|\overrightarrow{x}) = \frac{1}{Z(x)}\prod_{i=1}^{m} \exp\big(\lambda_i f_i(\overrightarrow{x}, y)\big). \tag{3.25}$$

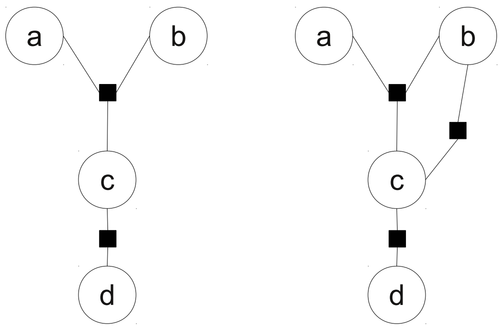

Figure 3.8: Two factor graphs

Directed graphs can be transformed into undirected ones. Actually the right graph in figure 3.7 represents the undirected version of the graph in the middle. The top edge has to be added in the undirected graph because the definition requires a maximal clique to be fully connected. As the joint distribution of the left graph contains the factor $p(c|a,b)$ these three variables a, b, c have to be in one clique of the undirected version.

3.5.3 Factor Graph

Factor graphs make this "partitioning" of the original graph explicit and introduce the concept of *factors*. The joint probability of a graph's variables is given by

$$p(\overrightarrow{x}) = \prod_s f_s(\overrightarrow{x_s}) \tag{3.26}$$

where f_s denotes a factor and $\overrightarrow{x_s}$ holds the variables contained in factor f_s.

Both the directed and undirected conditional independence graph are special cases of the factor graph. For the former equation 3.22 is the factors; for the latter equation 3.23 is the factors.

Figure 3.8 illustrates two factor graphs. The factors are represented by black squares and are interconnected with the variables which are contained by the factor. Note that both two graphs represent possible versions of the conditional dependency graphs in figure 3.7.

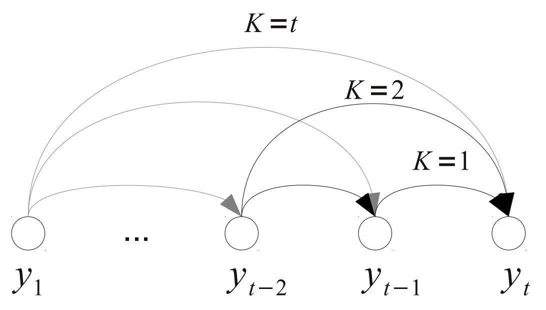

Figure 3.9: K^{th} order Markov chain

3.6 Hidden Markov Models

Markov Models take the nature of sequences explicitly into account and therefore yield consideration of dependencies between sequence-items. When talking about Markov Models one speaks about states and transitions in between them. What is being inspected are the probabilities of a variable for taking different values when it has seen a sequence of other values in the past. The classification mapping this time is not from an input vector $\overrightarrow{x}$ onto some arbitrary other discrete variable y but is a mapping onto x_t based on the observations of x_t in the past what is the vector $\overrightarrow{x} = (x_{t-1}, x_{t-2}, ...)$ when $1 \leq t \leq T$ describes the steps in time. In the case of $\mathfrak{S}$ variable x_t would be the state a manufacturing machine takes at a moment. With a Markov Model one could calculate the transition probabilities of the machine's state into another one based amongst other upon the information provided by former states.

The probability distribution of a probability variable x at time t is given by

$$p(x_t|x_{t-1}, ..., x_1) \propto p(x_t|x_{t-1}, x_{t-2}, .., x_{t-K}). \tag{3.27}$$

The joint distribution of an entire sequence is therefore given by

$$p(x_t, x_{t-1}, ...x_2, x_1) = p(x_1) * p(x_2|x_1) * p(x_3|x_2, x_1) * ... * p(x_t|x_{t-1}, x_{t-2}, .., x_{t-K}). \tag{3.28}$$

If $K = 1$ this model is called *first-order Markov chain* and according to K K^{th} order Markov chain. The joint probability can be expressed as a graph as it is illustrated in Figure 3.9.

The circles symbolize the states whereas the arrows indicate conditional dependencies between them. Intuitively it is not necessary in a very long sequence that one

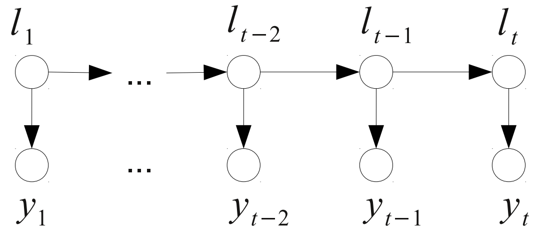

Figure 3.10: Hidden Markov chain

state depends on all of it's predecessors. Furthermore according to [Bis07] the number of parameters in a K^{th} order Markov chain grows exponentially with K so this model is impracticable for large values of K.

The Hidden Markov Model (HMM) overcomes this. A latent variable is introduced which kind of summarizes everything important that has happened in the past [Ben96]. Hereby the conditional dependencies between the states themselves are eliminated. What value the new latent variable takes at a certain time-step is conditioned solely on the value it took one time-step before. The different states are conditioned on the values of the latent variable. The states at different time-steps are therefore dependant on each other through the latent variable. If the latent variable is indicated by l the probability distribution variable x taking the different states at time-step t is given by

$$p(x_t|x_{t-1},...,x_1) = p(l_t|l_{t-1}) * p(x_t|l_t). \tag{3.29}$$

The joint probability distribution of an entire sequence is given by

$$p(x_t, x_{t-1},...,x_1) = \prod_{t=1}^{T} p(l_t|l_{t-1}) * p(x_t|l_t). \tag{3.30}$$

Figure 3.10 illustrates this graphically.

[Ben96] states in practical applications l and the values it can take have a particular meaning. Obviously it should be designed complex enough so it can actually capture all important information from the past.

3.7 Conditional Random Fields

As the Hidden Markov Model described in section 3.6 is the sequential version of the generative Naive Bayes classifier (section 3.2) there exists a sequential version

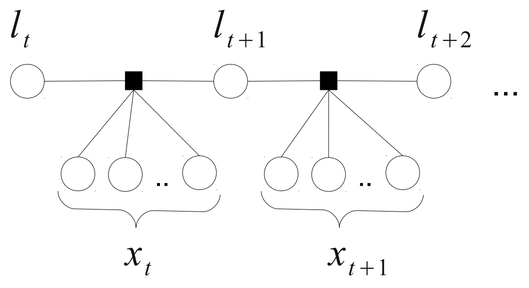

Figure 3.11: Linear chain CRF according to [SM06]

for the discriminative Maximum Entropy Classifier from section 3.3 as well. This classifier named *Conditional Random Field* (CRF) is described in this section and was first introduced by [LMP01]. The Maximum Entropy Classifier's advantage is the reduction of parameters to be estimated as described earlier. This is because the probability distribution $p(\overrightarrow{x})$ of input vector $\overrightarrow{x}$ does not have to be modelled. Because of this according to [KT07] there exist applications where the CRF is of higher order than $K = 1$. This means the probability distribution of a sequence's observation at time step t is not just conditioned on a variable from time step $t-1$ but on several others as well. [KT07] and [SM06] give examples of different designs.

The one design similar to the Hidden Markov Model called *Linear Chain CRF* is discussed in detail here as it suites well for explaining the fundamental theory behind Conditional Random Fields.

The CRF is generally formulated as

$$p(\overrightarrow{l}|\overrightarrow{x}) = \frac{1}{Z(\overrightarrow{x})} \prod_C \psi_C(\overrightarrow{x_C}, \overrightarrow{l_C}) \tag{3.31}$$

where the denominator is defined by

$$Z(\overrightarrow{x}) = \sum_{l'} \prod_C \psi_C(\overrightarrow{x_C}, \overrightarrow{l'}). \tag{3.32}$$

When comparing equation 3.31 to 3.23 one sees the only difference is the added $\overrightarrow{l}$ which comes from the conditioned probability distribution which has to be estimated by the model.

[SM06] defines the linear chain CRF as follows

$$p(\overrightarrow{l}|\overrightarrow{x}) = \frac{1}{Z(\overrightarrow{x})} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(l_t, l_{t-1}, \overrightarrow{x_t})\right) \tag{3.33}$$

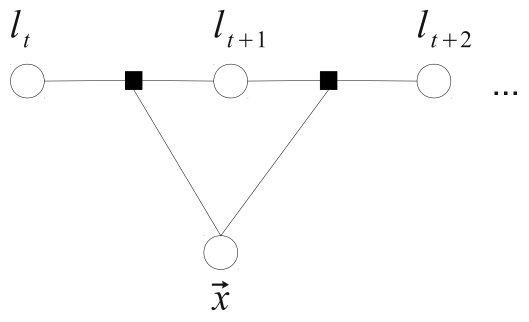

Figure 3.12: Linear chain CRF according to [KT07]

where t represents time steps and $\overrightarrow{x_t}$ symbolizes the observations in $\overrightarrow{x}$ which are needed for the computation at time step t. Denominator $Z(\overrightarrow{x})$ is given by

$$Z(\overrightarrow{x}) = \sum_{l} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(l_t, l_{t-1}, \overrightarrow{x_t})\right). \tag{3.34}$$

Figure 3.11 visualizes this concept. [KT07] amend this by introducing the concept of parameter tying. This means feature function f_i does not use it's own parameter $\overrightarrow{x_t}$ any-more but references the hole input vector $\overrightarrow{x}$. f_i receives a new input parameter j which denotes the information for which time step the calculation is done. The linear chain CRF takes therefore the following form:

$$p(\overrightarrow{l}|\overrightarrow{x}) = \frac{1}{Z(\overrightarrow{x})} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(l_j, l_{j-1}, \overrightarrow{x_t}, j)\right) \tag{3.35}$$

where $Z(\overrightarrow{x})$ is given analogously to equation 3.34 $\overrightarrow{x_t}$ exchanged by $\overrightarrow{x}$ and extended by the parameter j. Figure 3.12 illustrates this.

4 Sequence Mining

Given a set of sequences $\widehat{S_1}, \widehat{S_2}, ..., \widehat{S_n}$ *Sequence Mining* is the term for algorithms which search for common sub-sequences in $\widehat{S_1}, \widehat{S_2}, ..., \widehat{S_n}$. Sequence Mining is no classification method per se but it can be useful for finding common sub-sequences occurring before some machine state in $\mathfrak{S}$. As the sequence mining algorithms compare multiple sequences a possible application would be chopping $\mathfrak{S}$ into pieces so that one piece maps on a particular state. Multiple pieces mapping on the same machine state may then be analysed by such sequence mining algorithms to find out common sub-sequences.

Those algorithms decide whether a sub-sequence in several sequences is common or not based on a threshold or *support* that one stipulates in advance. The first sequence algorithm *GSP* was introduced 1995 by [AS95]. As being often referenced in other scientific papers the algorithms described in this chapter seem to be the primal approaches yet existing. This chapter is based on the work of existing scientific work which are all referenced throughout this chapter. None of the described methods is derived by myself.

4.1 Further definitions

Srikant and Agrawal's [AS95] definitions of a sub-sequence are slightly different to the definitions in chapter 2. They define a sequence $\widehat{B} = \langle b_1, b_2, .., b_N \rangle$ as subsequence of $\widehat{A} = \langle a_1, a_2, .., a_M \rangle$ if $N \leq M$ and there exist integers $1 \leq i_1 < i_2 < ... < i_N \leq M$ such that $b_1 = a_{i1}$, $b_2 = a_{i2}$, ... $, b_N = a_{iN}$. This means that between two sequence items a_x, a_y in $\widehat{A}$ where $1 \leq x < y \leq M$ corresponding to two adjacent items b_z, b_{z+1} in subsequence $\widehat{B}$ where $1 \leq z \leq N$ there might exist several $a_{q_1}, a_{q_2}, ..., a_{q_O}$ where $x < a_{q_1} < a_{q_2} < ... < a_{q_O} < y$ saying that there might exist a gap of arbitrary size between a_x and a_y in $\widehat{A}$. Throughout the hole chapter this definition is used when speaking of a subsequence. In the chapters afterwards the definition of chapter 2 is used again.

In addition they add the term of the *sequence database* $\bigoplus$ as a set of tuples $\langle sid, \widehat{S} \rangle$ where *sid* is the sequence id of sequence $\widehat{S}$. A tuple $\langle sid, \widehat{S} \rangle$ contains another sequence

$\widehat{A}$ when $\widehat{S} \sqsupseteq \widehat{A}$ and therefore S is a super-sequence of $\widehat{A}$ and $\widehat{A}$ is a sub-sequence of $\widehat{S}$. The *support* of a sequence $\widehat{A}$ is defined by the number of tuples $\langle sid, \widehat{S} \rangle$ in the sequence database which contain $\widehat{A}$. They call a sequence $\widehat{A}$ a *sequential pattern* if it's support in sequence database $\bigoplus$ exceeds a pre-defined positive integer denoted as *support threshold*.

Furthermore the term *element* of a sequence is introduced which may not be of high interest for our actual task but may enable some sort of multi-dimensional sequence mining as discussed in [JHY05]. In a sequence $\widehat{S} = \langle s_1, s_2, ...s_L \rangle$ s_n is denotes an element of $\widehat{S}$. It can be either an item $i_h \in \overline{I} = \{i_1, ..., i_H\}$ directly or another set of items $\{i_1, ..., i_H\}$ called *transactions*. Those transactions are a set of items which happened at the same time. This means the order of the items in a transaction does not matter. The origin of those transactions lies in the purpose Srikant and Agrawal introduced GSP for namely, customer behaviour. A transaction represents actions of a customer he has done at the same time or for which this can be assumed, for example ordering different books in one step. An example for a sequence of item set $\overline{I} = \{a, b, c\}$ containing a transaction could be $\langle ac(ab)b \rangle$. An item i_k can appear multiple times in different elements of a sequence $\widehat{S}$ but can never appear more than once in one element and therefore in one transaction.

4.2 Apriori algorithms

The first family of algorithms presented is the family based on the *apriori-principle* which states that every super-pattern of an infrequent pattern cannot be frequent. Two algorithms are introduced; first the *GSP* algorithm and second the *SPADE* algorithm. In the end the concept of a third algorithm called *SPAM* is shortly noted.

4.2.1 GSP

As mentioned in the introduction of this chapter the GSP algorithm was the first sequence mining algorithm and was introduced by [AS95]. It is an algorithm which builds up candidate sequential patterns based on a horizontal data representation. The algorithm grows candidate patterns from length one until it can't find no more candidates. It therefore evaluates every possible combination of items from the item set until it reaches the stopping criterion for every search branch. The stopping criterion is the *apriori* principle which means that it stops growing a candidate branch when the support of the candidate is below the threshold. When the algorithm finds a

Sequence id	Sequence
1	$\langle a(abc)(ac)d(cf)\rangle$
2	$\langle (ad)c(bc)(ae)\rangle$
3	$\langle (ef)(ab)(df)cb\rangle$
4	$\langle eg(af)cbc\rangle$

Table 4.1: Sequence database according to the example of [JHY05]

non-frequent pattern at a certain branch it stops growing it. Until then it performs a candidate-generation-and-test search in which it iteratively combines frequent patterns found at a certain iteration to generate new candidates for the next iteration and finally tests if the candidates are actually frequent in the given sequences. This is best illustrated in a short example which is adopted from [JHY05].

Table 4.1 contains an example sequence database which should serve for explanation purposes. Analysing the four sequences of table 4.1 GSP works as follows. First it scans the database for frequent sequential patterns of length 1. It therefore finds $\langle a\rangle : 4, \langle b\rangle : 4, \langle c\rangle : 4, \langle d\rangle : 3, \langle e\rangle : 3, \langle f\rangle : 3, \langle g\rangle : 1$. Given a support threshold of 2 the sequential pattern $\langle g\rangle$ is considered as frequent and it's search branch for further growing is closed. All other patterns are investigated more deeply.

GSP works iteratively so the results from one iteration are passed to the next and used as *seeds* for generating new pattern-candidates. In each step it combines the seeds by piecing the seeds together. This combination of two seeds is only possible if the subsequence gathered from the first seed by deleting it's first item equals the subsequence from the second seed by deleting it's last item. The result of combining two seeds $\widehat{S_1}$ and $\widehat{S_2}$ is $\widehat{S_1}$ extended by the last item of $\widehat{S_2}$. It has to be checked whether the last item of $\widehat{S_2}$ can be added both as part of the last transaction in $\widehat{S_1}$ and as item on its own respectively. This means in iteration two of our example GSP finds 51 new pattern-candidates by combining the seeds what leads to the set $\{\langle aa\rangle, \langle ab\rangle, ..., \langle af\rangle, \langle ba\rangle, ..., \langle ff\rangle, \langle (ab)\rangle, \langle (ac)\rangle, ..., \langle (ef)\rangle\}$.

Srikant and Agrawal explain the candidate-testing algorithm of GSP in [AS95] considering gap constraints. For evaluating if a sequence contains several candidate patterns they use a hash tree structure of the candidate patterns for efficiently reducing the number of candidates to be checked. The support of a candidate is given by the number of sequences it contains. If the support of a candidate is below the support threshold this candidate is deleted from the set. This means this algorithm produces candidates what does not mean these candidates are actually frequent. In fact, it produces candidates which need not to occur in the sequences to be analysed at all.

In this way of combining the different seeds in one iteration GSP reduces the search space of candidate patterns. Nevertheless [JHY05] states three "inherent costs".

First the GSP algorithm may have to generate many candidate patterns. When N_c depicts the number of seeds in one iteration the total number of candidates to be generated is calculated by $N_c^2 + \frac{N_c(N_c-1)}{2}$. The first addend stands for the seeds that have to be combined as own element whereas the second stands for the combination as part of the last transaction. [JHY05] gives an example that for 1000 seeds at an iteration this algorithm will produce $1000^2 + \dfrac{1000 * 999}{2} = 1,499,500$ candidates.

Second [JHY05] criticizes the many database scans the algorithm has to perform what is supposed to be very costly.

Related to the first point is third which says that this algorithm has severe problems mining long sequences. [JHY05] shows that for a sequence of length 100 where every item appears only once in the sequence GSP has to generate approximately 10^{30} candidates.

4.2.2 SPADE

SPADE is similar to GSP an algorithm that is based on the apriori-principle and was introduced by [Zak01]. Unlike GSP it does not use a horizontal data representation but transforms the sequence database into a vertical one and so combines different subsequences or patterns slightly different what leads to less database accesses and "reduced computational costs by using efficient search schemes" according to [Zak01]. A horizontal representation means a tuple in the database to consist of the following entries: *SID*, *EID*, *Elements*; whereas *SID* depicts the sequence id, *EID* denotes the time a sequence element "happened" and this element is denoted by the entry *Elements*.

Table 4.2 illustrates the vertically transformed version of the first sequence in table 4.1. Similar to GSP SPADE grows sequential patterns from length one. For each pattern the information in which sequence SID at which time EID it is contained is stored in the vertical representation. In this way the support of the pattern is easily evaluated by counting the distinct SID's. Patterns of the length $l + 1$ are generated by

SID	EID	Elements
1	1	a
1	2	(abc)
1	3	(ac)
1	4	d
1	5	(cf)
2	1	(ad)
2	2	c
2	3	(bc)
...	...	...

Table 4.2: Vertically transformed version of the sequence database in table 4.1

combining patterns of length l similar to GSP and making use of the apriori-principle. Therefore the tuples containing the SID's and EID's of all the elements in the pattern are considered.

If an item is denoted by i for a pattern of length l these tuples are stored as follows: $SID, EID(i_1), EID(i_2), ..., EID(i_l)$. Each pattern has therefore a set of such tuples. The tables in figure 4.3 illustrate this concept for the length-1 pattern a, b and their combination ab. When combining two patterns SPADE uses a slightly different method from that of GSP. Whereas GSP deletes the first item in the first sequence and the last from the second sequence to check if the resulting subsequences are equal SPADE combines patterns with the same prefix of length $k-1$ when k depicts the length of the pattern. [Zak01] uses the fact that a sequence $\widehat{S}$ can be constructed from two of its subsequences of the sequence's length - 1. Namely, the first being the one with the last item deleted from $\widehat{S}$ and the second having dropped the second to the last item of $\widehat{S}$. This way not all EID's of all items in the pattern have to be stored as mentioned before but only the last. When merging two patterns with the same prefix of length $k-1$ when k depicts the length of the pattern three candidates are constructed namely, one with the last item from the first pattern as last item, one with the last item of the second pattern as last item and one having the two last items in one transaction. For all these candidates its constructing patterns' last items have to be investigated of having the right EID's. This means for instance if two patterns $\widehat{S_1} = abc$ and $\widehat{S_2} = abd$ construct a candidate $abcd$ the EID of c in $\widehat{S_1}$ has to be lower than the one of d in $\widehat{S_2}$ to increment the support of $abcd$ by one.

a

SID	EID
1	1
1	2
1	3
2	1
..	..

b

SID	EID
1	2
2	3
..	..

ab

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

Table 4.3: Pattern candidate *ab* being constructed from frequent patterns *a* and *b* according to table 4.2. The support of *ab* is a intersection of the support of *a* and *b* whereas for a particular SID the EID's have to appear in the right order.

abc

SID	EID
1	15
1	20
4	20

abd

SID	EID
1	15
1	22
4	25

abcd

SID	EID
1	22
4	25

Table 4.4: Combination of two 3-length frequent patterns *abc* and *abd* to construct *abcd*. Only one EID namely the one of the last item per occurrence of the patterns is stored. When combining two frequent patterns to a new candidate only the one EID has to be investigated of being in the right order.

Figure 4.4 illustrates this example. If the number of distinct SID's in the pattern's set of SID's and EID's (e.g. its support) is below the support threshold this pattern is considered infrequent and not used for further combinations.

Although SPADE brings plenty of improvements compared to GSP [JHY05] states that it still generates too many pattern candidates in it's breadth-first search to grow long sequential patterns. That is why the authors introduce *PrefixSpan* which belongs to the family of pattern-growth algorithms [HPY99] and is explained in the next chapter.

4.2.3 SPAM

The SPAM-algorithm [AGYF02] is another apriori-approach which chooses a bitmap-representation of the sequences what leads to the advantage of generating sequential candidate patterns by bit-operations which can be done very efficiently. In the experiments done by [AGYF02] the algorithm outperforms GSP, SPADE and even the pattern-growth algorithms described in the next section on large datasets.

4.3 Pattern-growth algorithms

These algorithms search for common sequential patterns in sequences by partitioning the sequence database where the patterns are searched. Two algorithms are discussed; the first being the *FreeSpan* [HPMA+00] algorithm and the second being the *PrefixSpan* [JHY05] algorithm which is actually based on the former.

4.3.1 FreeSpan

The idea of FreeSpan is to partition the sequence database according to *projections* to reduce the search space for new candidate patterns. A $\widehat{S}$-projected-database is the set of sequences which contain the sequence $\widehat{S}$. Furthermore the set of sequential patterns is divided into disjoint subsets. When $\overline{I} = \{i_1, i_2, ..., i_N\}$ is the item set containing all frequent items (this means all patterns of length one - these patterns are sorted in descending order according to their support) the patterns can be divided as follows: the patterns containing i_1 but no item after i_1 in $\overline{I}$, the patterns containing i_1 and i_2 but no item after i_2 in $\overline{I}$ and so on. This means the n^{th} subset contains all items i_1 to i_n but no item after i_n in $\overline{I}$.

Based on these principles the algorithm first searches for all length-one frequent patterns and sorts them in descending order according to their support. After that it builds the projected databases of every length-one frequent pattern and searches for further frequent patterns of length two. Again the algorithm constructs the projected databases of the length-two frequent patterns to search for length-three patterns and so on.

Let's take again the example database of table 4.1 for illustration purposes where the support threshold is 2. As GSP FreeSpan first finds the length-1 one frequent patterns in the following order: $\langle a\rangle : 4, \langle b\rangle : 4, \langle c\rangle : 4, \langle d\rangle : 3, \langle e\rangle : 3, \langle f\rangle : 3$. Next the algorithm constructs the projected databases. First the one for pattern $\langle a\rangle$ is processed leading to the $\langle a\rangle$-projected-database $\{\langle aaa\rangle, \langle aa\rangle, \langle a\rangle, \langle a\rangle\}$. Mining this projected database only

$\langle aa \rangle$ is found as frequent length-two pattern. Processing $\langle b \rangle$ leads to the $\langle b \rangle$-projected-database which takes the following form: $\{\langle a(ab)a \rangle, \langle aba \rangle, \langle (ab)b \rangle, \langle ab \rangle\}$. Three length-two sequential patterns are extracted from this database: $\{\langle ab \rangle, \langle ba \rangle, \langle (ab) \rangle\rangle$. Building the $\langle ab \rangle$-projected-database takes the form $\{\langle a(ab)a \rangle, \langle aba \rangle\}$ resulting in the frequent length-two pattern $\langle aba \rangle$. The mining of all the other projected databases of length-two patterns does not result in any longer frequent patterns as well as mining the projected database of pattern $\langle aba \rangle$ so the projected database construction is stopped here. The algorithm proceeds with length-one pattern c to continue this proceedings for all frequent patterns.

According to [HPMA$^+$00] this algorithm outperforms GSP but has the disadvantage of constructing many non-trivial databases as stated by [JHY05]. The algorithm PrefixSpan introduced by [JHY05] and discussed in the next section fixes the order of item projection and allows the application of a technique called *pseudo-projection* which faces this problem.

4.3.2 PrefixSpan

The PrefixSpan [JHY05] algorithm exploits a projection technique similar to the FreeSpan algorithm although it projects the databases slightly different. First of all PrefixSpan brings the element in a transaction in a alphabetical order. A sequence $a(ba)$ would therefore be the same as $a(ab)$. As the items in one transaction happen all at the same time this transformation is valid. The database projections do not consist of whole sequences any-more but contain the suffixes of the sequences which follow on a particular sequence prefix $\widehat{P}$ when the database is projected with respect to $\widehat{P}$.

The definitions of a prefix and a suffix of a sequence are stated as follows. Given a sequence $\widehat{S} = \langle s1, s2, s3, ..., s_N \rangle$ a sequence $\widehat{P} = \langle p_1, p_2, p_3, ..., p_M \rangle$ is called a prefix of $\widehat{S}$ if

- $s_i = p_i$ for every $i < M$
- $p_M \subseteq s_M$ and
- all items in transaction $(s_M - p_M)$ are alphabetically after the ones in p_M.

Sequence $\widehat{Q} = \langle q_1, q_2, ..., q_O \rangle$ where $q_1 = (s_M - p_M)$, $O = N - M + 1$ and $q_j = s_{M+j-1}$ for $1 < j \leqslant O$ is called suffix of sequence $\widehat{S}$ with respect to $\widehat{P}$. The suffix $\widehat{Q}$ of $\widehat{S}$ with respect to $\widehat{P}$ is empty if prefix $\widehat{P}$ is not a suffix of sequence $\widehat{S}$.

Again the set of sequential patterns is split into several subsets as we have seen it similarly for FreeSpan. This works slightly different this time. First the complete set of sequential patterns can be divided into subsets, each corresponding to the set of sequences having the length-one pattern x_n as prefix when $\{x_1, x_2, ..., x_n\}$ denotes the frequent patterns of length one in the given sequences. These subsets can be further divided. When prefix $\widehat{P}$ is a sequential pattern of length l let the set of length $l+1$-sequential-patterns having $\widehat{P}$ as prefix be the set $\{\widehat{S_1}, \widehat{S_2}, ..., \widehat{S_M}\}$. $\widehat{S_1}, \widehat{S_2}, ..., \widehat{S_M}$ can be prefixes again on their own and so divide the entire set of frequent patterns into M further disjoint subsets. The m^th subset where $1 \leq m \leq M$ are the frequent patterns with $\widehat{S_m}$ as their prefix.

The algorithm then grows frequent patterns from length one to recursively project the sequence databases with respect to the frequent patterns of length l when searching for patterns of length $l+1$. In the projected databases frequent items (or patterns of length one) are searched and concatenated to pattern responsible for the database projection. This gives a new frequent pattern and a new possibility for a projection.

The algorithm is best illustrated by the example from table 4.1 with support threshold 2. Once again the algorithm finds the frequent patterns of length one: $\langle a \rangle : 4, \langle b \rangle : 4, \langle c \rangle : 4, \langle d \rangle : 3, \langle e \rangle : 3, \langle f \rangle : 3$. After this subpatterns with prefix $\langle a \rangle$ are searched. Therefore the $\langle a \rangle$-projected-database is constructed which takes to following form: $\langle a(abc)(ac)d(cf) \rangle, \langle (\_d)c(bc)(ae) \rangle, \langle (\_b)(df)cb \rangle, \langle (\_f)cbc \rangle$. A transaction which contains an underline like $(\_d)$ means that the following items (in this case the d) is contained in the same transaction as the pattern responsible for the projection. In the $\langle a \rangle$-projected-database the algorithm searches for frequent items again. This time it finds: $a : 2, b : 4, \_b : 2, c : 4, d : 2, f : 2$ resulting in the length-two frequent patterns in this search branch: $\langle aa \rangle : 2, \langle ab \rangle : 4, \langle (ab) \rangle : 2, \langle ac \rangle : 4, \langle ad \rangle : 2, \langle af \rangle : 2$. The result opens new branches for mining for frequent patterns. The projection of the sequence database with respect to $\langle aa \rangle$ leads to $\langle (\_bc)(ac)d(cf) \rangle, \langle (\_e) \rangle$ where no new frequent items are found. The $\langle ab \rangle$-projected database takes the form $\langle (\_c)(ac)d(cf) \rangle, \langle (\_c)(ae) \rangle, \langle c \rangle$ leading to frequent items $(\_c) : 2, a : 2, c : 2$. The $\langle a(bc) \rangle$-projected-database has one frequent item, namely $a : 2$ while all the other projections of length-three patterns in this branch have no frequent items. The resulting frequent patterns with prefix $\langle ab \rangle$ are therefore $\langle a(bc) \rangle : 2, \langle \langle a(bc)a \rangle : 2, \langle aba \rangle : 2, \langle abc \rangle : 2$. The same is done with all the other branches.

This kind of projection allows to apply a technique which [JHY05] calls *Pseudo-Projection*. The projected-databases are therefore not constructed actually physically but with the help of pointers and offset-information the projecting can be accomplished directly in the original sequences. Details how this can be done are given in [JHY05].

4.4 Constraint based sequence mining

All the algorithms are extended in that way that it is possible to search for sequential patterns by defining constraints. This has two advantages:

1. the user is not overwhelmed with sequential patterns he is not interested in
2. when the constraints are pushed deeply into the mining algorithm the search space is reduced leading to a better performance

This is why all authors of the algorithms described so far extended their algorithms in that way. In [GRS99] the authors describe how to extent the GSP algorithm. As they consider regular expressions to be a very powerful way for defining constraints a framework is presented which describes how to push regular expressions into the algorithm. Because regular expressions are not *anti-monotone*, what means that a sequence fulfilling a pattern constraint does not mean that all of it subsequences fulfil this constraint either, they present several algorithms which push the constraint at different levels of degree into the GSP algorithm. Because a regular expression can be thought of as a automata one can distinguish if the sequence is

- *legal with respect to a state of the automata*, what means that every two elements in the sequence are described by an edge between two states of the automata,
- *valid with respect to a state of the automata*, what means that every two elements in the sequence are described by an edge between two states of the automata and the last element of the sequence corresponds to an end-state or
- *valid to the automata*, what means that every two elements in the sequence are described by an edge between two states of the automata and the last element of the sequence corresponds to an end-state and the first element to a start-state.

According to this [GRS99] introduces the *Spirit(N)*-algorithm which simply checks if all elements in a sequence actually occur in the regular expression, the *Spirit(L)*-algorithm which requires the sequence to be legal with respect to some state of the regular expression, the *Spirit(V)*-algorithm which requires the sequence to be valid with respect to some state of the regular expression and the *Spirit(R)*-algorithm which requires the sequence to be valid to the regular expression. As according to [GRS99] there is no efficient method for the Spirit(R)-algorithm to generate pattern candidates but a brute-force method the Spirit(V)-algorithm is named the overall winner in performance on mining general sequences.

SPADE is extended by [Zak00] to incorporate constraints in the algorithm. M.J. Zaki shows how to push constraints into the algorithm concerning the length (number of overall elements) and width (number of elements in one transaction) of a pattern, the minimum and maximum gaps between two elements in a pattern, the maximum duration of a pattern, constraints about particular elements and a class handling.

[PHW07] explains how constraints are pushed into the PrefixSpan-algorithm. Item constraints, length constraints, gap constraints, regular expressions and aggregate functions are handled and grouped according to monotonicity (every super-sequence also fulfil the constraints fulfilled by a sequence), anti-monotonicity and succinctness (constraints provide information so that sequential patterns fulfilling the constraints can be constructed directly) by the authors.

4.5 Combining sequence mining and classification

Sequence mining and classification could fit together well as sequence mining may reduce the feature space for the classification task. [LZO99] proposes exactly this combination and states that in an experiment on three datasets the classification accuracy could be improved by 10-50%. The authors believe that three heuristics are essential for selecting the right features:

1. Features should be frequent
2. Features should be distinctive
3. Features should not be redundant

According to this [LZO99] proposes an algorithm which is an extension of the SPADE-algorithm and considers the heuristics listed above.

Furthermore [TL09] introduces a new apriori-based algorithm for mining distinctive features. Instead of using common classification algorithms the authors apply their own classifier constructed by a scoring metric.

4.6 Episode Mining

Sequence Mining is an approach where common patterns in different sequences are searched and presented to the user. In our case of mining the history of machine states $\mathfrak{S}$ the only way of applying a sequence mining algorithm would be to slice the sequence of states in $\mathfrak{S}$. When one is just interested in finding common patterns

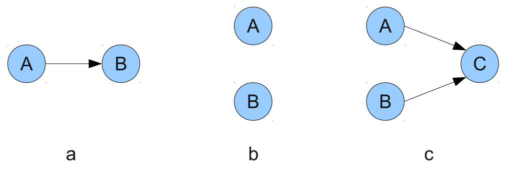

Figure 4.1: Different kind of episodes. The one above the a symbolises a serial episode, the one above the b symbolises a parallel episode and the one above the c symbolises a non-serial and non-parallel episode.

containing some machine state then the slices would form the neighbourhood nearby that one interesting state.

Things get more complicated if one searches for patterns without knowing which machine states should be involved. Every possible slice of a certain length would have to be cut out from the one sequence and grouped for each machine state they are mapping on. For each state common patterns would have to be searched by applying sequence mining algorithms and amongst those states the common patterns would have to be compared to find exclusive ones. Besides the fact that this procedure seems to be very complicated there is a major issue with choosing the length of such a slice. As one does not know how long the patterns might be the length of the slices has to be chosen arbitrarily. If they are chosen too short it might be that longer patterns are cut off and will not be found. Choosing the length of the slices too long may lead to finding patterns where the machine states are widely spread [CG03]. This could be a problem if the occurrences of different machine states is strongly varying leading to a higher probability a certain subsequence is actually found. Another problem is described by [MR04] where the authors state that in sequence mining the frequency of a pattern is determined by the number of sequences containing it. So if there appears a pattern more than one time in a sequence its frequency count is nevertheless increased just by 1.

Episode Mining tries to overcome these drawbacks. It is explicitly designed for finding all possible re-occurring patterns called episodes in one long sequence.

One of the first scientific works concerning episode mining is [MTV97]. They formally introduce what an episode is. They consider a sequence equally as in sequence mining as an ordered set of events happening one after another. As an episode they consider a triple $(\overline{V}, \leq, g)$ where $\overline{V}$ is a set of nodes, $\leq$ is a partial order and $g : \overline{V} \rightarrow \overline{I}$ where $\overline{I}$ is the set of events is a mapping associating each node with an event type.

Figure 4.1 illustrates different kind of episodes. The nodes A, B, C are nodes from set $\overline{V}$. In episode mining it is searched for multiple occurrences of events which can be mapped to these patterns. The left pattern above the (a) is a serial pattern which states that an event has to happen after another. Patterns searched and found in sequence mining only take the form of a serial pattern. Furthermore in episode mining there exist the parallel pattern where the relative order between two states is not fixed and the non-parallel and non-serial pattern where two events precede in a non fixed order before another event.

[MTV97] proposes two kinds of search algorithms. The first one called *WINEPI* is a window based algorithm which shifts a window of a certain size across the hole sequence and handles each subsequence finding place in the window as the sequence algorithms handle their sequences. The frequency of an episode is thereby determined by the number of windows in which the episode occurs. The size of the window has to be chosen by the user as well as a threshold for the ratio of the overall number of possible windows compared to the number of windows containing a pattern that indicates if that pattern is considered frequent or not. Episode candidates are thereby generated equally as candidate patterns are generated by the sequence mining algorithm GSP namely, by combining existing ones. Indeed it is the same generate-and-test apriori approach as applied by GSP. Candidates are tested by scanning over the sequence using an automata to check whether a candidate is entirely contained in a window.

The second algorithm is called *MINEPI* and handles the problem slightly different. This algorithm introduces the concept of *minimal occurrences*. The authors state that an episode has a minimal occurrence in a time window where it does not occur in any sub-window of it. Here a support threshold (number of minimal occurrences) has to be defined by the user to distinguish a frequent episode from a non-frequent one. Candidate episodes of length l are again constructed by combining frequent episodes of length $l-1$. To verify if an candidate episode is actually frequent only the minimal occurrences of the sub-episodes have to be considered. The algorithm therefore needs some data-structure to store the information about the minimal occurrences. When scanning over the minimal occurrences of the sub-episodes of a candidate episode again an automation is used to detect occurrences.

In the same context[MTV97] introduces *episode rules*. Episode rules state that if there exists a frequent episode e_1 which is contained by another frequent episode e_2 how probable is e_2 within a certain time window when e_1 occurs within another time window? Therefore frequent episodes are sliced in two so that only the minimal occurrences of the sub-episodes have to be considered for calculating the probability.

As it is not quite clear of how to choose these window sizes [MR04] gives an heuristic of how to do so.

[LSU07b] gives an introduction for efficiently working with and restricting the automata used for candidate testing. The authors state that only *distinct* and *non-overlapped* occurrences of episodes in a sequence should count for their frequency to provide efficiency in space and time complexity of the algorithm. Distinct means occurrences of episodes share no events with other ones. Non-overlapping means an occurrence of an episode does not begin in between the events of another occurrence.

The same authors introduce in [LSU07a] the concept of episodes when the events of episodes are combined with a duration information which might be of particular interest in our case. In fact in [LSU07a] the authors inspect a log of machine states similar to the case of this thesis. They give an introduction of how to handle the time information and give some principal design hints. In episode generation and testing their algorithm proposed works similar to that of [MTV97].

5 Approach

This chapter describes the approach how the experiments in chapter 6 were accomplished. First of all it shortly repeats what is the goal of the experiments to subsequently introduce the approach of how the experiments were conducted and therefore how the original data was preprocessed and provided to the algorithm used for the experiments.

The experiments and evaluations are done to face problem (2) as explained in the introduction of this thesis what is, showing that patterns exploitable for a classification task (meaning correlations between machine states in sequence $\mathfrak{S}$) exist in real-world data-sets and that finding them is possible.

The experiments are accomplished with a tool called *LibSVM* [CL01] for constructing Support Vector Machines so the problem of achieving the thesis' goal is handled as a classification task. libSVM is chosen because it is a tool well approved and Support Vector Machines are said of being able to handle better the *curse of dimensionality* problem than other methods [BDLR06]. The curse of dimensionality states that classification algorithms' performances decrease with the number of features used for the classification task. This means adding more features to the instances of the trainings- and test-sets should not make the SVM to suffer that hard as other methods in classification. Choosing the Support Vectors to be instances on the borders of each class in the feature space the interpretation of the constructed model should be possible as well. To keep the model built by the Support Vector Machine more interpretable only linear basis functions for feature transformation are used.

Algorithms and methods falling in the sequence- and episode mining approach are not used in the experiments because no appropriate freely-available software-tools were found at the time this thesis is being written. The application (and therefore maybe an implementation) of a method described in chapter 4 is left open for further works.

In this chapter first preliminary approaches not meeting the expected results are introduced. Then the procedure of transforming the original data set into a representation which can be interpreted by a Support Vector Machine is described.

5.1 Preliminary approaches

This section shortly summarizes what other methods had been tried out before making use of Support Vector Machines by applying LibSVM and why the results accomplished were not satisfying.

The first method investigated were Conditional Random Fields as described in section 3.7. A tool called *MALLET* [McC02] was applied but as its task is to classify and tag text and documents it did not fit the task at hand. For its sequence tagging (what means to classify a sequence's items) MALLET seems to include the information provided by a sequence item's successor. This lead to the fact that MALLET classifies randomly generated sequences containing no patterns with an accuracy of $> 98\%$ when the task is processed as in our case.

A freely available PrefixSpan 4.3.2 implementation [Tab10] was tried out. That for, subsequences mapping on a particular machine state in $\mathfrak{S}$ were cut out of the sequence of machine states $\widehat{S}$ to search for common patterns. Unfortunately this implementation does not provide gap constraints so the number of resulting, mostly irrelevant patterns is overwhelming so that the examination of actually useful patterns in short time becomes not possible. Other freely available sequence- and episode mining implementations were not found at this time.

Before using the Support Vector Machine algorithm of LibSVM another sequence tagging algorithm was applied called SVM$^{\mathrm{HMM}}$ [Joa10]. It is a tool that trains Hidden Markov Models (section 3.6) using the Support Vector Machine formulation. This means in all equations of section 3.4 there are transition and emission probabilities according to the Hidden Markov Models introduced. For details [Joa10] refers to [ATH03]. One major drawback of this method is that there is no possibility to give weightings on target labels in the data. Reasons why one should do that in very imbalanced data sets like the one used in the experiments for this thesis are given in [AKJ04]. Another problem is that the algorithm is not able to output the confidence by what the algorithm made its classifications (see section 5.5.2 why the confidence is important for the experiments).

After those methods LibSVM was applied and yielded first promising results.

5.2 The original data set

The original data is stored in a relational database system. One table in this system stores the sequence of machine states in the following form:

machine	status	date from	date to	shift nr
BU10	SCHM	27.07.2007 09:46:46	07.08.2007 09:47:50	1
OL13	PROD	27.07.2007 09:46:32	07.08.2007 09:47:55	1
BU10	PROD	27.07.2007 09:45:10	27.07.2007 09:46:46	1
..	..	..	..	..

As the sequence of states is logged for more than one machine the first column from the left indicates what machine the data row belongs to - in this case the top row is an entry of machine "BU10" the row in the middle of machine "OL13" and the lower row of machine "BU10" again. The second column from the left indicates the name of the state which lasts from "date from" until "date to" denoted by the third and fourth column from the left - in this example there is the state "SCHM" logged which lasted from 09:46:46 o'clock until 09:47:50 o'clock on the 27th of July 2007 for machine "BU10". Additionally in the rightmost column there is the shift recorded in which this state occurred (in this case the status "SCHM" took place in the first shift).

In this way the states of each machine are logged over the time without gaps. The machine states reach from production (indicated by "PROD" in the example) to planned and unforeseen downtimes. To extract the sequence of states for one machine only the rows concerning this machine have to be considered (which means the leftmost column must contain the name of the machine). According to the time information ("date from" and "date to") the states recorded in column status form the sequence of states for one machine. In other words; reading column "status" top down in rows concerning one machine gives the sequence of states in $\mathfrak{S}$for this machine. "date from" and "date to" provide additional duration information.

To make it possible that a Support Vector Machine builds models about this data it has to be transformed and represented in a different way what is the topic for the rest of this chapter.

5.3 Vectorization

In chapter 3 classification was explained as the mapping from an input vector $\overrightarrow{x}$ on a discrete value of a variable y. In other words representative value combinations in the dimensions of $\overrightarrow{x}$ are searched for each value y can take.

The sequence of states for one machine in $\mathfrak{S}$ has the form $\widehat{S} = \langle s_1, s_2, ...s_L \rangle$ where L is the length of the sequence ($s_1, s_2, ...s_L$ denote the column "state" in the original data table read top down for one machine). In the task of predicting an sequence item

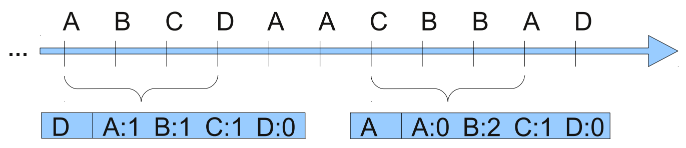

Figure 5.1: Example how $\mathfrak{S}$ is transformed. The blue arrow symbolizes the history of machine states. In this example four possible states are possible, namely A, B, C and D. The metric in this example would simply count the occurrences of each state until three states before the label. In the two blue boxes two example instances are given. On the left of each box the label is separated by a vertical line from the features. On the right hand side each state A, B, C, D is assigned its value. Note that this is just an example as the duration of every state is ignored and states have take Integer values in LibSVM.

s_l the item itself is clearly the label that is to be predicted and can therefore be seen as the variable y. As we want to construct a mapping onto s_l from its predecessors in $\widehat{S}$ it is them providing the information for input vector $\overrightarrow{x}$. Therefore it is crucial to decide how many predecessors are considered and how the information about their distance to the label in $\widehat{S}$, their duration and the number of their occurrences nearby the label is provided to the Support Vector Machine.

LibSVM receives its training- and test-data via an ASCII-file. In this file each line represents an instance of either the training-set or the test-set. One such line has to take the following form:

```
Label Feature1:<value1> Feature2:<value2> ... FeatureQ:<valueQ>
```

Label depicts the label (and therefore variable y) which is represented by an Integer value. The features *Featurex* forming the input vector $\overrightarrow{x}$ are represented by Integer values as well. Each dimension of $\overrightarrow{x}$ is thereby one *Featurex*. The values of the features $< valueX >$ take Float values. These values should be expressive enough to separate different classes (machine states in the broader sense) if the separation is possible. The question is therefore how to design these values as we have to transform the discrete formulation of $\mathfrak{S}$ into a numerical one.

Figure 5.1 illustrates an example of how training- and test-instances are generated from the history of machine states $\mathfrak{S}$ for being processed by the Support Vector Machine. For each machine state on the time-line one instance in the form of the two blue boxes is generated. In this example the value of each feature is given by the number of states occurring within the window of states being considered. Other informations such as the duration of each state or its distance to the label my be considered as well as we shall see shortly. This technique is called *sliding window*-technique as a window of a defined value is shifted one item after another over a sequence. Each subsequence finding place in the window is extracted to form one instance of whether training- or test-dataset.

First the right size K for the window is chosen. If $\widehat{S} = \langle s_1, s_2, ...s_L \rangle$ describes the sequence of machine states in $\mathfrak{S}$ then a subsequence finding place in the window can be denoted by $\widehat{S_t^K}$ where $1 \leq t \leq L - K$ is an arbitrary start point in $\widehat{S}$. This window is then shifted across $\mathfrak{S}$ by incrementing the subsequence's start index t from 1 to $L-K$. After each incrementation $\widehat{S_t^K} = \langle s_t, s_{t+1}, ..., s_{t+K-1} \rangle$ is transformed into whether a training- or test-instance by extracting the label y and constructing the input vector $\overrightarrow{x}$.

The label y is the last item s_{t+K-1} of $\widehat{S_t^K}$ and can therefore take as value each possible machine state $i_h \in \overline{I}$ if $\overline{I} = \{i_1, ..., i_H\}$ denotes the set of possible machine states. $\overrightarrow{x}$ is constructed from the information provided by $s_t, s_{t+1}, ..., s_{t+K-2}$. Before it can be explained how these sequence items contribute to $\overrightarrow{x}$ it has to be described what the dimensions of $\overrightarrow{x}$ (which are the features) are. Each dimension $x_n \in \overrightarrow{x}$ stands for a possible predecessor of label s_{t+K-1}. Such a predecessor might be a single machine state i_h as well a transition from $i_{h1} \in \overline{I}$ to one or several other machine states $i_{h2}, i_{h3}, ..., i_{hT} \in \overline{I}$. T hereby denotes the length and therefore the number how many machine states are being considered in one transition(if $T = 1$ then this transition is simply i_h). Such a transition can be denoted by $d_n = i_{h1} \bullet i_{h2} \bullet ... \bullet i_{hT}$. For a transition $i_{h1} \bullet i_{h2} \bullet ... \bullet i_{hT}$ to appear in sequence $\widehat{S}$ a subsequence of the form $\widehat{S_t^T} = \langle i_{h1}, i_{h2}, ..., i_{hT} \rangle$ has to be contained at some point t in $\widehat{S}$ (therefore a transition corresponds to the one particular subsequence containing all the transition's items and having the transition items' order). This concept is illustrated in figure 5.2. One dimension x_n therefore corresponds to the n$^{\text{th}}$ possible unique transition of machine states in the history of a machine, denoted by d_n. Because a transition corresponds to a subsequence the number of dimensions in $\overrightarrow{x}$ is given by the number of unique subsequences from length 1 to length N (where N has to be predefined) found in the machine history of states $\widehat{S}$. When referring to x_n we refer to the sequence of machine states contained in d_n.

Every subsequence $\widehat{S_{t'}^n}$ where $t \leq t' \leq K - n$ and $n = 1..N$ appearing in $\widehat{S_t^{K-1}}$

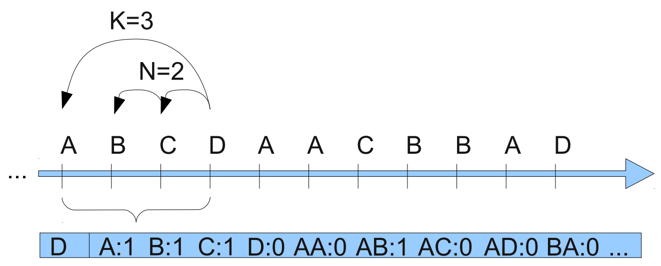

Figure 5.2: Example how $\mathfrak{S}$ is transformed. K depicts the number of machine states considered before a label. N stands for the number of state transitions to be regarded in the feature space of a label. In this example there is $k = 3$ and $n = 2$ leading to the blue box instance example where the state transitions $AA, AB, AC, AD, BA, ..., DD$ considered.

influences the value of only the corresponding dimension $x_n \in \overrightarrow{x}$. The presence of a transition d_n in $\widehat{S_t^{K-1}}$ shall be denoted by $d_n \in \widehat{S_t^{K-1}}$. The absence of a possible transition d_n in $\widehat{S_t^{K-2}}$ is denoted by $d_n \notin \widehat{S_t^{K-2}}$ and leads to the corresponding dimension x_n having value zero. How the presence of a transition d_n in $\widehat{S_t^{K-1}}$ influences the value of its corresponding dimension x_n is described in the next section.

5.4 Dimension weighting

In the previous section it was explained how the data from the history of machine states $\mathfrak{S}$ has to be transformed for being processed by the Support Vector Machine learning algorithm LibSVM. It was explained that for the generation of training- and test-instances for the classification task a sliding window technique is applied. This sliding window is shifted over the sequence of machine states $\widehat{S}$ in $\mathfrak{S}$ whereas of each subsequence $\widehat{S_t^{K-1}}$ finding place in the window an instance for whether the training- or test-dataset is generated. Thereby the label y is extracted and the input vector $\overrightarrow{x}$ is generated. While the previous section described what the dimensions $x_n \in \overrightarrow{x}$ stand for this section gives insight how the values for each dimension are generated.

As the previous section mentioned if a transaction of machine states $d_n = i_{h1} \bullet i_{h2} \bullet ... \bullet i_{hT}$ where $i_{h1}, i_{h2}, ..., i_{hT} \in \overline{I}$ denoting the set of possible machine states does not occur in $\widehat{S_t^{K-1}}$ then its dimension x_n in $\overrightarrow{x}$ has got value zero in the training- or test-instance constructed from $\widehat{S_t^{K-1}}$. If d_n does occur in $\widehat{S_t^{K-1}}$ there are several values x_n could take which are all listed as different metrics in the following.

The first metric is a simple check whether the transaction d_n occurs in $\widehat{S_t^{K-1}}$ and therefore would be

$$x_n = \begin{cases} 1 & d_n \in \widehat{S_t^{K-1}} \\ 0 & otherwise \end{cases} . \tag{5.1}$$

The second metric is a counting how often a transaction d_n occurs in $\widehat{S_t^{K-1}}$. So if $\#d_n \in \widehat{S_t^{K-1}}$ denotes the number d_n occurs in $\widehat{S_t^{K-1}}$ the metric is defined as

$$x_n = \#d_n \in \widehat{S_t^{K-1}}. \tag{5.2}$$

The third metric is an exponential function whose base is the summarized distance of the occurrences of transaction d_n in $\widehat{S_t^{K-1}}$ to label state s_{t+K-1} in $\widehat{S}$ (meaning by distance the number of other states between s_{t+K-1} and the last state of an occurrence of d_n). This can be summarized as

$$x_n = exp(E) = \sum_{d_n \in \widehat{S_t^{K-1}}} \big(t + K - 1 - pos(d_n)\big)^E \tag{5.3}$$

where $pos(d_n)$ denotes the index of the last machine state of a particular occurrence of transaction d_n in $\widehat{S}$.

By combining these metrics it is hoped to achieve a better performance in the experiments. So it makes sense to combine the second with the third metric what doubles the number of dimensions x_n whereas the first half taking values calculated by the second metric and the second half taking values calculated by the third metric. As the value for x_n computed by the third metric is a summary of multiple occurrences of d_n it may be that x_n would take the same value in an training- or test-instance when d_n appeared once nearby the label y as when it appeared multiple times in greater distance to y. Combined with the counting metric it is assumed that the Support Vector Machine is able to distinguish between these two cases. Nevertheless an overlapping of values for dimensions $x_n \in \overrightarrow{x}$ for different circumstances in subsequences S_i^k in the third metric exists even when being combined with the other metrics. This drawback is taken into account and left open for designing more sophisticated metrics or using more sophisticated data mining methods than Support Vector Machines.

It has to be mentioned that in the end the values have to be normalised when different metrics are combined so that one does not measure with different scales. Throughout the experiments in section 6 all values are normalised to have zero as mean value and a standard deviation of 1. [SS06] proposes this way of normalisation when the absolute minimum and maximum values of variables are not know such as in this case.

The original data sets are summarized in Appendix B by which it should get clear that there is a heavy imbalance how often machine states appear in the data sets. That is why a feature weighting method is introduced which is called "term frequency-inverse document frequency"(idf) in information retrieval [SB87]. For machine states that occur very often the probability is high that they occur in $\widehat{S_t^{K-1}}$ for the instances of almost all label states. The assumption in idf is that features appearing less often might separate classes better in a classification task. Therefore dimensions corresponding to machine states transaction that appear less often are given a higher weighting. In addition infrequent machine states leading to other states might be of higher interest than the most frequent ones as this would result in clearer patterns. The concept of idf can be expressed as

$$x_n = x_n * log(\frac{\#d_n \in \widehat{S}}{L - K}) \tag{5.4}$$

where L denotes the length of the sequence $\widehat{S}$ in $\mathfrak{S}$ and K the length of the transaction d_n.

As well as the sequence of machine states itself and its positions of states the duration of the machine states transitions are taken into account. The duration of an occurrence of a transition is the summarized duration of all its machine states. Therefore the input vector $\overrightarrow{x}$ is expanded by adding new dimensions x_n which take as value the summarized duration of a machine state transition in subsequence $\widehat{S_t^{K-1}}$. It has to be noted that in between the dimensions $x_n \in \overrightarrow{x}$ there exists no explicit connection which can be taken into account for training the Support Vector Machine.

5.5 Evaluation methods

What is being evaluated in a classification task is how well the model built by a classifier like the Support Vector Machine upon the training-dataset performs on predicting the right labels for a test-set the algorithm has not seen in training-phase. Two ways of evaluating the performance are done. This chapter explains the two ways.

A classical approach would be to compare common characteristics like the *Accuracy* [WF02] as ratio how often the Support Vector Machine labelled the right machine state to the total number of test-instances. Indeed it turned out during the experiments that such a measure is not very appropriate for evaluating the performance of the Support Vector Machine on sequence data. The reasons are given in the next two sections.

5.5.1 Distance measure

The first reason why a measure like the accuracy described above is not appropriate is, because of the sequential nature very often there occurs a blurring in the labelling. One could think of the labels predicted by the Support Vector Machine as a second sequence. In the end there are two sequences of machine states: the first being the original sequence of $\mathfrak{S}$ and the second being the sequence of labels indicating the machine states the Support Vector Machine suggested for each test-instance. It might be that if a machine state appears in the sequence of $\mathfrak{S}$ at time step a it does not appear exactly at time a in the sequence of labels but somewhere in its neighbourhood at time step b. The accuracy as it was defined before would only count labels which are placed at exact the right position as correct but as this blurring cannot be considered to be entirely wrong one can use the distance from a test-instance to the right label as an error-measure.

To give an example: in the synthetic data sets generated to show that the methods applied do work and described in detail in section 6.1 there are patterns introduced such that one machine state A appears almost exclusively before another one B at a position which is varying from one to two positions from time to time. In one test set this label appears 40 times. 22 times out of that 40 the Support Vector Machine placed the label for state B at exactly the time it appeared in $\mathfrak{S}$ what leads to an accuracy as defined above of 55%. When taking a more detailed look it turned out that 13 times the Support Vector Machine placed that label for state B in the close neighbourhood of the time step it actually appeared in $\mathfrak{S}$. This would lead to the fact that the Support Vector Machine would have labelled all occurrences of the pattern (because five times state B appeared without the corresponding state A) if the close neighbourhood is considered correct as well. As the variation of positions in patterns of real-world data can be assumed to be much more varying it makes sense to take the neighbourhood in the sequence of labels of the time step a machine state occurs in $\mathfrak{S}$ into account.

Therefore for each label the average distance from a to b may be considered. The lesser these distances are the better the constructed model is or the more significant patterns exist. This can be done the other way around as well namely, determining the average distance from b to a. This distances indicate how "good" the guesses of the Support Vector Machine are. The distance is a crucial measure because it directly influences in what time span a machine state may be predicted. Figure 5.3 illustrates this concept.

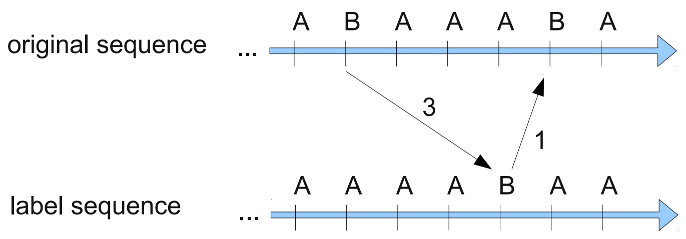

Figure 5.3: Distances of machine states in the original and the label sequence. The upper sequence symbolizes an extract of $\mathfrak{S}$ whereas the lower could be a labelling sequence generated by a Support Vector Machine. So for instance the first state A in $\mathfrak{S}$ was correctly labelled as an A by the SVM. For the second state B in $\mathfrak{S}$ the Support Vector Machine erroneously labelled an A as well. One could now measure the distance from that state B in $\mathfrak{S}$ to the first occurrence of a labelled B which is in this case of length 3. Vice-versa the distance from label B to the next equivalent in $\mathfrak{S}$ is 1 as indicated by the second arrow. One sees one drawback of this method in this example as one label could be used for measuring the distance to two or more items in $\mathfrak{S}$ and vice-versa.

5.5.2 Increase in probability

The second reason why classical evaluations like the accuracy measure above is rather inappropriate for our case gets illustrated in section 6.2 of the experiments' chapter. In this section it should get clear that what the Support Vector Machine actually labels and therefore assumes as machine state for a particular test-instance is not of highest interest because of the imbalanced number of machine states. This imbalance might lead to the fact that a very rare machine state does not get labelled by the Support Vector Machine because the overall probability that another very frequent state follows is still higher - even though there exists a pattern which makes the probability for that infrequent state slightly higher. To put this into an example: if there exists a state A which appears in 20% of all instances and another state B which appears in only 5% of all cases then there might exist a pattern for state B which make its probability of an appearance to double to 10% for a certain test-instance. But even though this probability increase the Support Vector Machine will label state A as its probability of (a little bit less than) 20% is still higher then the 10% of state B.

What can be done with LibSVM is not just to output the labelled machine state

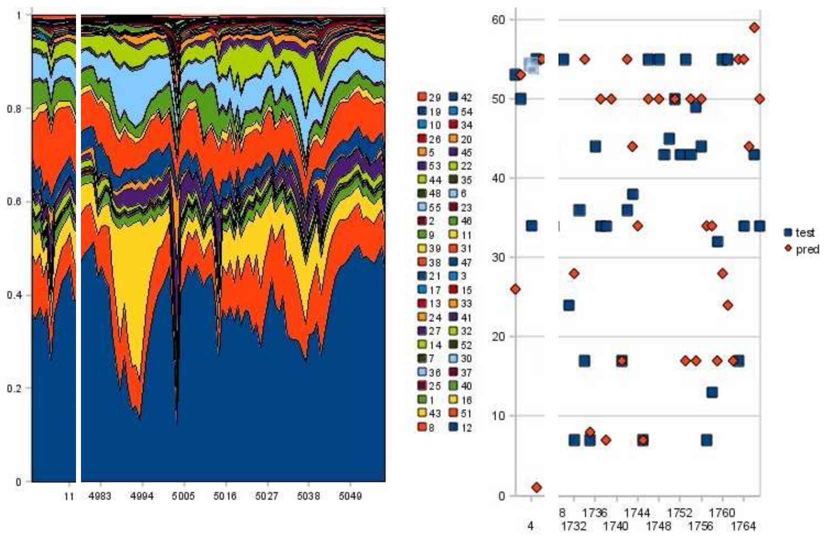

Figure 5.4: The two kind of evaluation charts. The left chart is the area chart symbolizing the probability distribution of the different labels estimated for each test-instance by the Support Vector Machine. Note that LibSVM works with numerical labels only so the machine states have to be mapped onto numbers which are listed on the chart's legend. The right chart illustrates the distribution of labels given by the test data (blue boxes) and the labels predicted by the Support Vector Machine (orange diamonds).

but to additionally output the probability each machine state takes for being labelled for one particular test-instance (naturally the Support Vector Machine picks the one machine state with highest probability as the label and these probabilities express the confidence by which a machine state is chosen as a label compared to the others). It is variations of probabilities for each machine state from one instance to another what might give insight if there exist patterns for that state or not.

This can be done in two ways. The first one is an instance-by-instance search for an increase of that probability for each machine state. To facilitate this search two kind of charts are used what gives the possibility of a visual analysis. Figure 5.4 introduces these two kind of charts.

On the left-hand-side one sees an area chart which is used for showing the probability distribution of each machine state to be labelled estimated by the Support Vector Machine at a certain moment. According to this on the vertical axis there is plotted the cumulated probability which ranges from 0 to 1. The horizontal axis symbolizes each test-instance what leads to the fact that the probability distribution of each label is shown over the time where each time step may take a different duration which is the nature of our test-instances in these experiments. Therefore each number on the horizontal axis is a test-instance and on the vertical axis for each such test-instance the probability of each label is plotted by the area. When there are strong differences in a label's probability from one test-instance to another it can be assumed that the Support Vector Machine found some patterns leading to this variation.

The right-hand-side chart takes for the horizontal axis the same information as the left chart which is each test-instance. This time on the vertical axis each label is plotted. What is being compared is what label a certain test-instance took (blue boxes) and what labels the Support Vector Machine assumed for this test-instance (orange diamonds). In the end the Support Vector Machine picks the label with the highest probability for a test-item what is to say that this label takes most area for the test-item in the left chart. When these two charts are laid one above another so that the vertical axis fit they can be well compared and it can be investigated if the appearance of a machine state as a test-instance leads to a corresponding increase in probability.

The second way of evaluating the increase in probability is to average the increase for each machine state and to put it into a number. What is being compared is the average probability the Support Vector Machine assumes for such a machine state when it does appear as a test-instance and when it does not. If the former average value is significantly higher than the latter it can be assumed the Support Vector Machine found patterns in the data which lead to this correct increase in probability. Furthermore with the help of the probabilities a ranking for each test-instance can be constructed giving each machine state a rank according to their probabilities. The first ranked machine state would therefore have the highest probability, the second ranked machine state the second highest probability and so on where the last ranked machine state would have the least probability. The average ranking of a machine state when it should be predicted compared to it's overall average ranking could be compared as well.

The numbers of all evaluations are given in detail in appendix C. The main figures analysed for the experiments' evaluation are therefore

- the average increase of probability of a machine state when it appears as a test-instance (which is accomplished by comparing the values of *prob hit* denoting the average probability of a machine state when it appears as a test-instance and the values of *prob rest* denoting the average probability of a machine state when it does not appear)

- the average increase of ranks of a machine state when it appears as a test-instance (which is accomplished by comparing the values of *rnk hit* denoting the average rank of a machine state when it appears as a test-instance and the values of *rnk rest* denoting the average rank of a machine state when it does not appear)

hi rnk hit, *hi rnk* and *lo rnk* are additional information which give the highest rank of a machine state when it appears as a test-instance , the highest rank of a machine state when it does not appear as a test-instance and the lowest rank of a machine state.

In the next section the outcome of the experiments is summarized.

6 Experiments

In this chapter the results of the experiments conducted for this thesis are summarized and evaluated. These Experiments are to show that there exist correlations in between machine states in $\mathfrak{S}$ and that the application of the presented method is able to detect them. Experiments are done on four datasets: two synthetic sets to illustrate how the Support Vector Machine performs on known data and two original OEE-Analyser^{TM}data sets gathered by a customer of GAMED to find first patterns which might be of interest.

This is done on four datasets. The first two are synthetic datasets which are built in several ways for understanding how the applied algorithm performs and how solutions look like. These synthetic datasets have the advantage that information can be pushed into them so that an evaluation how well the algorithm unearths this information gets possible. The second dataset is a real-world dataset where it is to be shown that machine states can be predicted by other ones.

6.1 Synthetic data equally distributed

First of all the evaluation methods described in the previous section are shown on synthetic data sets to give a feeling how these behave on data containing only one pattern. Therefore two synthetic data sets have been constructed. They take the same form as the sequence of machine states in the original data from the OEE-Analyser^{TM}so they are handled and transformed as though they were original data. In the first set all the machine states are randomly appended one to another whereas all the states appear approximately equally often in the set. The second synthetic data set is the same but the states appear unequally often. In both data sets there appears one small pattern namely, one machine state that appears almost exclusively at one position before another one. Both synthetic data sets are described in detail in Appendix A. The equally distributed data set is analysed in this section - the non equally distributed in the next.

This synthetic test-set is analysed with the following parameters: 30 states before a

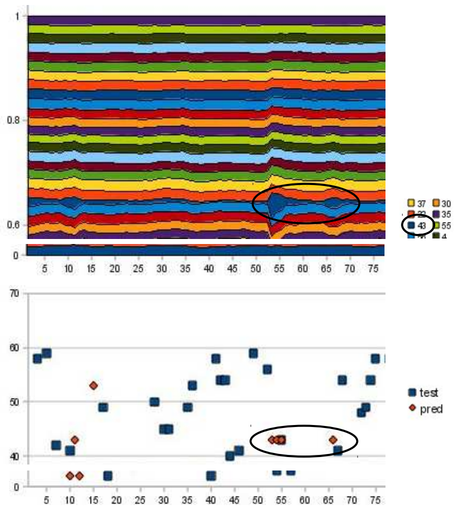

Figure 6.1: Visual evaluation of the equally distributed synthetic data set. The probabilities of all the labels are equally distributed as they are put together randomly. Only the one distribution in the black circle varies as this is the machine state having one other state appearing almost exclusively before it. In the lower chart one sees that this variation suffice to make the Support Vector Machine actually predicting correctly that label for this test-item.

label are being considered so K is set to 30. As the pattern appears only for a machine state and not for a transition no transitions are considered so N is set to 1. Metrics two 5.2 and three 5.3 of section 5.4 are combined. The parameter E of metric three is set to 1. No durations of machine states are taken into account as well as idf (5.4 in section 5.4) is switched off.

Figure 6.1 shows an extract of the charts for the equally distributed synthetic data set. As one can see all the probabilities but one are equally distributed in the upper chart as being expected because no patterns exist for these labels. The only information the Support Vector Machine is provided with for these labels is the overall number these labels occur in the data set. The one probability distribution that varies in the upper chart is the probability of that label 43 whose machine state has another state appearing almost exclusively before it. Here the Support Vector Machine unearths this pattern and raises the probability for that label at a moment the label actually

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
36	45	31.82	30.37	0.017417578	0.017474243
37	42	2.29	2.26	0.020117208	0.020146249
43	37	10.95	52.94	0.035496723	0.009532571
..	..	..	..	..	..

Table 6.1: Excerpt from Appendix C to illustrate the numbers of labels 37 and 43 for the equally distributed synthetic data set.

appears in the test-set.

Also the numerical evaluation methods indicate that there exists a pattern for label 43. It is the label having minimum distance from both the test-item to the next correctly predicted label and the label being predicted to the next correct test-item. Label 37 is another label having very low mean distance from the test-item to the next correct label (10.6 versus 38.12). But when looking at the average distance from the predicted labels to the next correct test-item one sees that this distance is much higher than that of label 43. Additionally the ratio between the number of label 37 actually appearing in the test-sequence and the label being predicted by the Support Vector Machine takes 1:38 whereas this ratio takes 1:4 for label 43.

When looking at the ranking according to the probabilities of each label this fact gets underlined. Despite the fact label 43 takes only the average rank 10.95 when an test-item appears with that label it has most variation in average probability when being to be predicted and when not (0.035 versus 0.009) as well as in ranking it takes rank 52 in average when it is not to be predicted which is an increase of about 42 ranks. Furthermore out of 37 cases when label 43 should have been predicted by the Support Vector Machine 29 times the label takes a rank below the first five ranks. The remaining 8 times might exists due to the fact that not always there appears this exclusive machine state before the one corresponding to label 43.

When looking at label 37 there is nearly no difference in all that numbers when comparing the averages in moments the label should have been predicted and when not. The fact why there is so little distance from an test-item taking that label to the next predicted label gets clear by the rankings as well. Label 37 has an overall average rank of 2 and is therefore predicted very often by the Support Vector Machine. Table 6.1 give the exact numbers of the two labels.

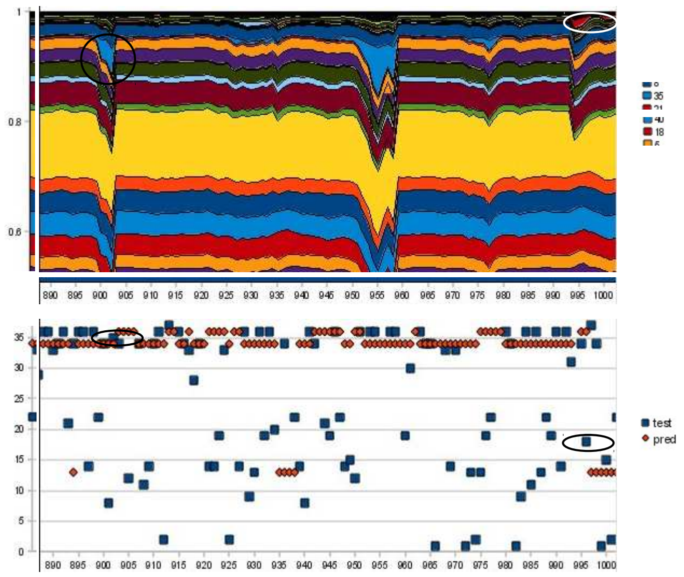

Figure 6.2: Visual evaluation of the non equally distributed synthetic data set.

6.2 Synthetic data non equally distributed

The second synthetic data set being inspected is the one with non equally distributed numbers of occurrences of each label. This time again there exist labels which have almost exclusive other labels appearing before them. This time these are the labels 35 and 18 whereas label 35 appears much more often in the data set than label 18.

Figure 6.2 shows the visual evaluation of this synthetic data set. This time the probabilities of each label are not equally distributed any more. One sees that the Support Vector Machine gives a higher probability to those labels that appear more often than others. This yields to a problem. As one can see the two labels for which patterns exist do have variations in their probability distributions when the test-items appear but these variations do no suffice for the Support Vector Machine to actually place the label (no orange diamond) because due to their overall frequency other labels are still more probable. This leads to the fact that the variation in probabilities and numbers should be considered in more detail than the actual placement of labels by the Support Vector Machine. Another problem might be that the variation of the less frequent label 18 (red) is much weaker than that of the more frequent label 35 (blue).

The variation in numbers uncovers the patterns of these two labels once again. This

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
17	1	42.0	34.89	2.34735E-4	9.261885E-4
18	3	16.0	38.24	0.022868035	7.084805E-4
35	12	4.33	30.68	0.09712201	0.0028474857
..	..	..	..	..	..

Table 6.2: Excerpt from Appendix C to illustrate the numbers of labels 18 and 35 for the non equally distributed synthetic data set.

time looking at the distance from the test-items taking these labels to its predictions and vice-versa does not make much sense because label 35 is just labelled 6 times out of 12 occurrences and label 18 is labelled 0 times out of 3 occurrences. Although the average distance of a predicted label 35 to the next test-items is very good (0.33) the average distance of a test-item to the next label is rather poor (34.11) because this label was simply predicted so little.

When looking at the variations of probabilities and rankings things clear up. Label 35 and 18 are the only ones with major differences in both probabilities and rankings. So label 35 increases by an average of 25 ranks when a test-item appears in the test-data and label 18 increases by 22 ranks. When looking at the average probabilities of these two labels one sees that even when these two items have an increased probability at moments an corresponding test-item appears there exist other labels which have an higher overall probability. Table 6.4 gives the exact numbers

6.3 Analysis of Machine 62

The purpose of the experiments done on real-world data from the OEE-Analyser$^{\mathrm{TM}}$is to show if there exist patterns exploitable for a classification in the data and to find a configuration of parameters that makes the Support Vector Machine to perform as well as possible. What is being compared are different configurations of the parameter N (saying how many machine states in one transition are being considered), idf 5.4 turned on and off and different exponential values E for the third metric 5.3 of section 5.4. Throughout all the experiments on real-world data metric two 5.2 and three 5.3 of section 5.4 are being combined.

The first real-world data set to be inspected is the one of a manufacturing machine

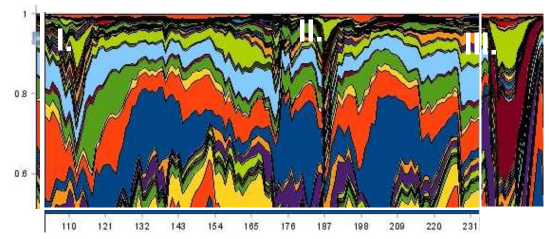

Figure 6.3: Machine 62, K = 30, N = 1, no idf, E = 1

with name "62". In the following different parameter settings are evaluated.

Figure 6.3 shows the first area chart of machine 62 with parameters K=30, N=1, no idf and E=1. What one can see here is that the chart differs greatly from the ones of the synthetic data. Labels like the ones corresponding to the upper light green area seem to have moments of high probability (at the right of the white Roman numbers) and others of very low probability. This means the Support Vector Machine must have seen patterns in the data so that it gives reason to increase the probability there.

When looking at the numbers one sees that the label indicted by the green area at the right of the Roman letter I. and taking number 14 has an increased probability of around 7 times (which are only 0.8 percent in this case) when it should be predicted compared to the moments it should not leading to an increase of 10 ranks (from rank 35 to rank 25). Label 44 corresponding to the light green area at the right of II. and III. has got an increase in probability from 0.2% to 2% yielding rank 15 when being to be predicted compared to rank 31 for the rest.

Figure 6.4 shows the same chart for the same machine and parameter settings but instead of N = 1 there is N = 2. The probabilities seem to be more varying than in figure 6.3 so it seems to be more different to the randomly distributed synthetic data sets. There appear more colours and therefore more labels than just those of the most frequent ones as in figure 6.3. The increased area of that label being label 14 in the figure 6.3 near test-item 110 has gone what is correct as there actually appears no such label in the test-data. That increase of label 44 in the previous example which is label 504 in this example is still there although there is no such label in the test-data as well. Beside that label there is a new increase in area from another label namely label 415. In this case indeed there is such a label in the test data around that area.

The increase of probability for label 14 being label 172 in this example has decreased to 5 times as well as the increase of ranks to 5 ranks at the moments it should be

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
14	2	25.5	35.11	0.009159606	0.0013156764
44	5	15.4	31.48	0.021003464	0.0023319414
..	..	..	..	..	..

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
172	2	31.5	36.31	0.0052873464	0.0013985289
415	357	18.55	18.82	0.008784693	0.007971122
504	5	17.8	31.41	0.02010085	0.002630011
..	..	..	..	..	..

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
1143	357	20.39	19.59	0.007745415	0.008021714
1923	5	20.6	31.97	0.015886622	0.0033250265
2226	2	32.0	37.47	0.0029192201	0.0014225753
..	..	..	..	..	..

Table 6.3: Excerpt from Appendix C to illustrate the numbers of interesting labels of machine 62 with different choices for N. The upper table contains the figures for N=1, the middle for N=2 and the lower for N=3

predicted by the Support Vector Machine. Probability and ranks have also increased slightly fewer for label 504 (corresponding to label 44 of the previous example). Label 415 shows nearly no increase in both probability and ranking. One problem might be the fact showing when analysing figure 6.4 in more detail. The grey vertical line shows that the increase of the label's probability happens short after the test-item takes the label. At the moment the test-item takes the label its probability is still at 2% and increases to 8% four items later. Other labels like 198 show an increase in ranks from 16 to around 6.5 which is an increase in probability by about 6 times.

When setting N = 3 it seems the variations of areas in the chart grow sharper and more. For label 2226 (14 or 172 previously) the probability increase has shrunk further but taking not very much effect on the increase of ranks as well as for label 1923 (44 or 504 formerly). Interestingly label 1143 (415 formerly) has in this example even a decreasing probability in moments when it should be predicted compared to the rest

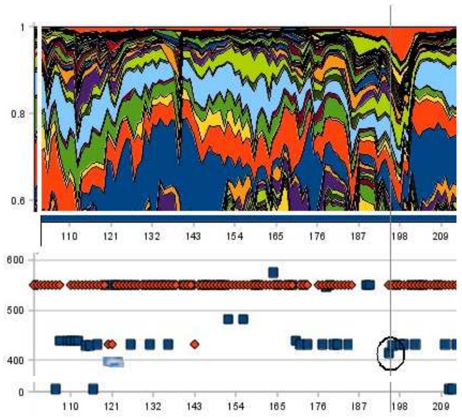

Figure 6.4: Machine 62, K = 30, N = 2, no idf, E = 1

although in figure 6.5 it seems the dislocation of increasing probability has lowered. It seems the prediction of this label is in this example very often off-place. Generally it has to be stated the numbers of this experiment do not indicate a better performance than the previous experiment. Label 1923 is the one with almost highest increase in ranks (11 ranks) - that is only outperformed by label 666 which only appears one time in the test-dataset. This holds for even the two labels indicated by I. and II.

Instead of increasing N there is the possibility of changing parameter E of metric three. Figure 6.6 compares area charts of different choices of E. There are no great differences but things seem to sharpen a little bit more with greater E. Also the numbers speak the same language. For almost all labels the increase of probability and ranks is a little bit higher for higher choices of exponential growth. This may lead to the assumption that machine states near the machine state to be predicted are more important for the prediction than machine states far away.

The inverse document frequency is the next thing to be compared. Figure 6.7 shows the chart for K=30, N = 1 and E = 1. Again there are not many differences between the two charts. Some peaks of areas seem to get clearer when enabling IDF and others seem to shrink. As well as the charts the numbers don't give explicit hints of whether IDF increases performance or not. Some labels have an higher increase in probability and ranks when IDF is turned on and others have a lower. It may depend on the task of whether it should be turned on or off. Furthermore it may be useful to consider both versions of evaluation.

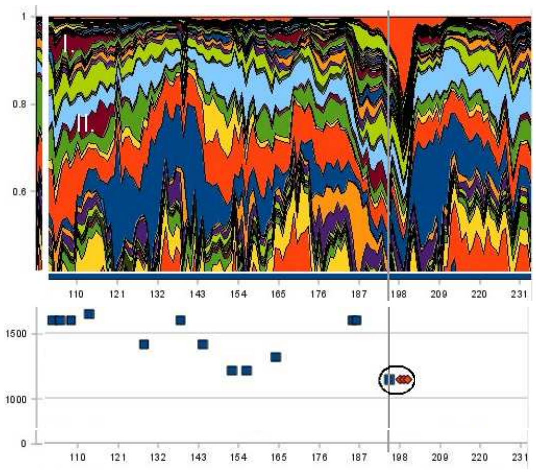

Figure 6.5: Machine 62, K = 30, N = 3, no idf, E = 1

In the following the three possibilities of parameter variation are tried to be combined. Therefore different N, different E and IDF being turned on or off are combined with each other. It is evaluated if these combinations lead to better results than the experiments before.

Figure 6.8 illustrates the comparison of area charts containing different combinations of IDF and E. There are only slight differences between each of them. For instance IDF seems to shift the orange area at the top a little bit more towards test-item 196 where the correct label actually takes place. $E > 1$ makes things a little bit more "peakier" - one may get the impression that there are slightly more colours apparent in the corresponding charts. The numbers impart the same. Greater E slightly outperforms the lower one for almost every label. IDF yields better performance only for particular labels - for others it may decrease performance.

6.4 Analysis of Machine 65

In this section another machine is evaluated. The most promising parameter configurations from experiments in the previous section are therefore applied for analysing the manufacturing machine with number 65.

In a first evaluation the parameters are chosen to be K = 30, N = 3, and E = 3.

A particular eye catcher in figure 6.9 is the dark red area around figures I. and II.

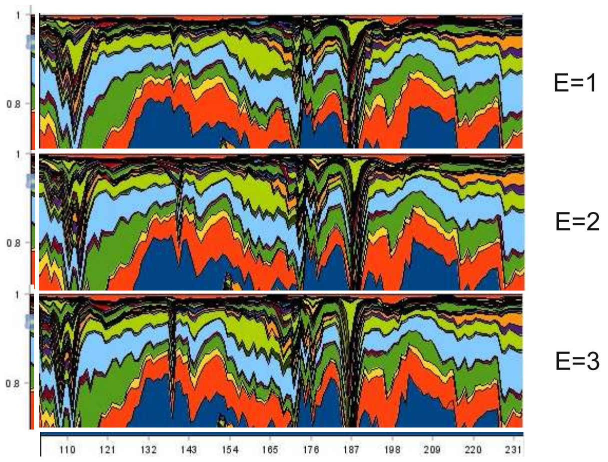

Figure 6.6: Area charts for machine 62 with different degrees of E for parameters K = 30, N = 1, no idf

It is the area of label 1421 which is also identified by the numbers to have a particular pattern. Label 1421 has an increase of about 20% when it should be labelled and therefore an increase of around 21 ranks. It might be of interest in this evaluation what makes the prediction of this labels so accurate.

Figure 6.10 shows an area chart where all the labels to be predicted are renamed: label 1421 is renamed to label 1 and the rest to label -1. Now the goal of the Support Vector Machine is to build a model being able of predicting this label 1. On the bottom one sees when label 1 appeared for a test-item (blue) and when the Support Vector Machine predicted this label (orange). In this chart it gets clear the Support Vector Machine found some pattern which lets it increase the probability for that label at the right moments. After considering this conspicuity with one expert from GAMED

label	count	rnk hit	rnk rest	prob hit	prob rest
..	..	..	..	..	..
1421	16	5.38	36.85	0.2838963	0.0042724586
..	..	..	..	..	..

Table 6.4: Excerpt from Appendix C to illustrate the numbers of label 1421 of machine 65.

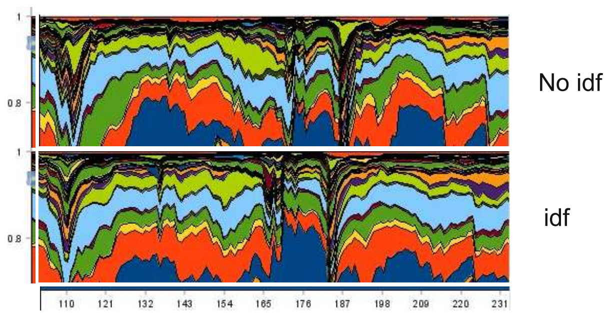

Figure 6.7: Area charts for machine 62 with and without IDF for parameters K = 30, N = 1, linear E = 1

it was figured out this label is corresponding to a machine state which forms the end state of another state. This means there exists a state which in most cases followed by another one forming the pattern unearthed by the Support Vector Machine.

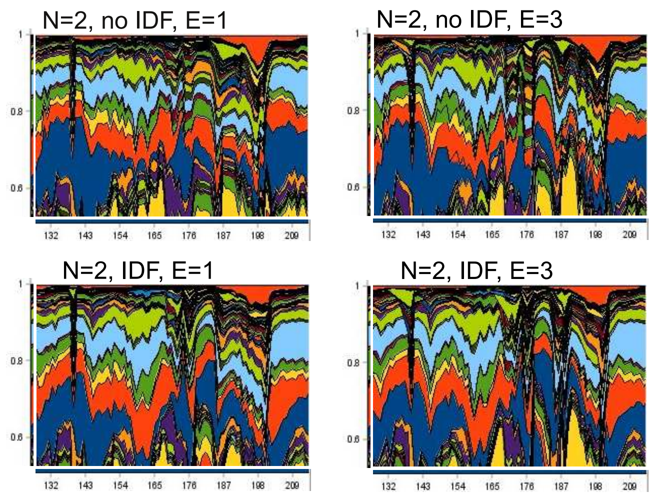

Figure 6.8: Area charts for machine 62 with N = 2 and different combinations of IDF and E

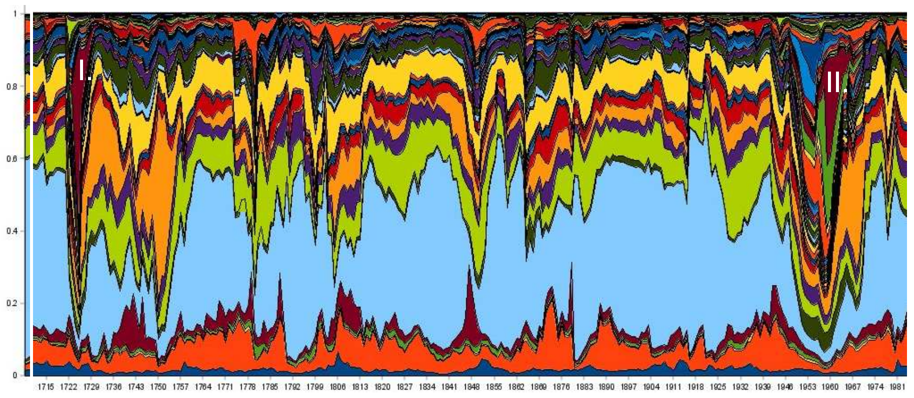

Figure 6.9: Area chart for machine 65 with N = 3, K = 30, IDF and E = 3

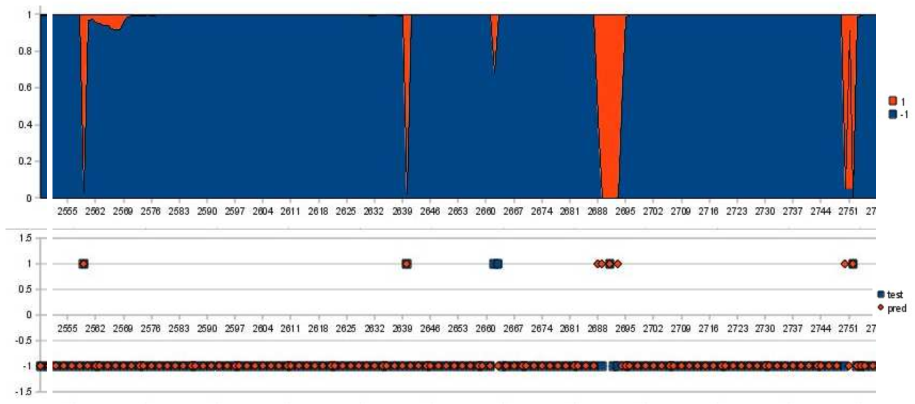

Figure 6.10: Area chart for machine 65 with N = 3, K = 30, IDF and E = 3 for only label 1421 versus the rest

7 Conclusion and Outlook

In this thesis an overlook over computer-aided methods and heuristics being able to analyse and comprehend a given sequence has been given. One of these methods was evaluated on real-world data. It was showed that during this evaluation involving Support Vector Machines a pattern was found that actually exists in the data. According to this the method represents a tool that is able to handle at least simple patterns. The question what kind of patterns the Support Vector Machine manages to detect and to what degree has to be analysed in more detail in future works.

Further engineering in feature representation might lead to better results as well as the application of sequential classifiers like Hidden Markov Models or Conditional Random Fields. Sequence Mining and Episode Mining algorithms might give good insights as well.

In the end it is the particular task which determines what methods should be applied and how features are to be represented. The ambiguity which machine states are of actual interest formed the greatest problem in evaluating the Support Vector Machine's performance. It has to be clearly defined what states should be predicted so that a proper evaluation is possible. The task is to be better formulated for further investigations.

A topic that has not been considered at all in this work is how stationary this problem is. This has to say that in the experiments the sequence of machine states has been considered as hardly not changing. This might be a problem when there are changes in machine states leading to others over time. This thought should be considered in further works.

Another main issue might be the different durations of machine states. Although in the experiments it has been tried to consider these durations it might be good to rethink how this is done. As for prediction there is a fixed number of machine states considered these unequal durations lead to different time spans in the features. This problem might be overcome by unfolding the machine states in time.

When the knowledge of related machine states is to be used as an early-warning system one has to think about the offset by which an prediction of a state is possible.

What might be another interesting point is an evaluation if test-labels around a drastic peak in the evaluation area-chart form a sequence which is of practical interest on its own. When there is such a drastic peak this means there has to be a significant difference to other parts of the test-data. It could be analysed what makes such peaks happen and if this reason has a particular meaning.

Facing the visual evaluation methods in this thesis it may be evaluated how well these are applicable in practical tasks. This means, are they suited for enabling non-datamining experts to solve concrete problems?

In the end this thesis lays the basis for further works and investigations and presented from which different angles the problem can be tackled.

Appendix A - Synthetic datasets

This chapter describes the two synthetic datasets used in chapter 6 in more detail. For each there is first given a chart visualising the distribution of occurrences for each label. In addition a table lists the exact figures of that distributions.

7.1 Equally distributed

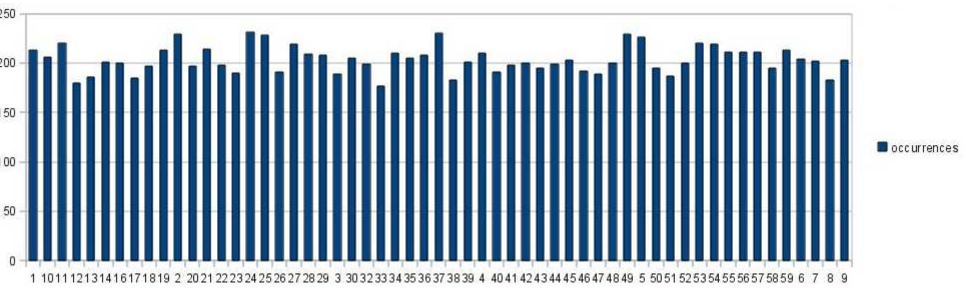

Figure 7.1: Distribution of labels of synthetic data set equally distributed

Figure 7.1 shows a chart which indicates the number of occurrences of each label. In the following the table with exact numbers is given. Label 31 is treated specially. It is appearing as almost every second label. For training and testing in classification instances mapping onto it are removed from the sets. This label exists because such a label and corresponding machine state exists in the real-world data and after all the synthetic data sets should adapt some properties of the real-world sets.

The occurrences of each label except label 43 are constructed randomly by applying the *rand-* function in JAVA. As one sees the distribution is far from being perfect equally. This might arise from the imperfect standard random function in JAVA. Label 43 is the one label for which there exists a pattern in the data. As being noted in the table below it appears 195 times. 185 times it is preceded by label 56 which appears 211 times in total. There is also a slight variation in the position of label 56 before 43.

label	number of occurrences	ratio to maximum occurrence
1	213	61.74647887323944
10	206	63.84466019417476
11	220	59.78181818181818
12	180	73.06666666666666
13	186	70.70967741935483
14	201	65.43283582089552
16	200	65.76
17	185	71.09189189189189
18	197	66.76142131979695
19	213	61.74647887323944
2	229	57.43231441048035
20	197	66.76142131979695
21	214	61.45794392523364
22	198	66.42424242424242
23	190	69.22105263157894
24	231	56.935064935064936
25	228	57.68421052631579
26	191	68.8586387434555
27	219	60.054794520547944
28	209	62.92822966507177
29	208	63.23076923076923
3	189	69.58730158730158
30	205	64.15609756097561
31	13152	1.0
32	199	66.09045226130654
33	177	74.30508474576271
34	210	62.628571428571426
35	205	64.15609756097561
36	208	63.23076923076923
37	230	57.18260869565217
38	183	71.8688524590164
39	201	65.43283582089552
4	210	62.628571428571426
40	191	68.8586387434555
41	198	66.42424242424242
42	200	65.76
43	195	67.44615384615385
44	199	66.09045226130654
45	203	64.78817733990148
46	192	68.5
47	189	69.58730158730158

label	number of occurrences	ratio to maximum occurrence
48	200	65.76
49	229	57.43231441048035
5	226	58.19469026548673
50	195	67.44615384615385
51	187	70.33155080213903
52	200	65.76
53	220	59.78181818181818
54	219	60.054794520547944
55	211	62.33175355450237
56	211	62.33175355450237
57	211	62.33175355450237
58	195	67.44615384615385
59	213	61.74647887323944
6	204	64.47058823529412
7	202	65.10891089108911
8	183	71.8688524590164
9	203	64.78817733990148

7.2 Non-equally distributed

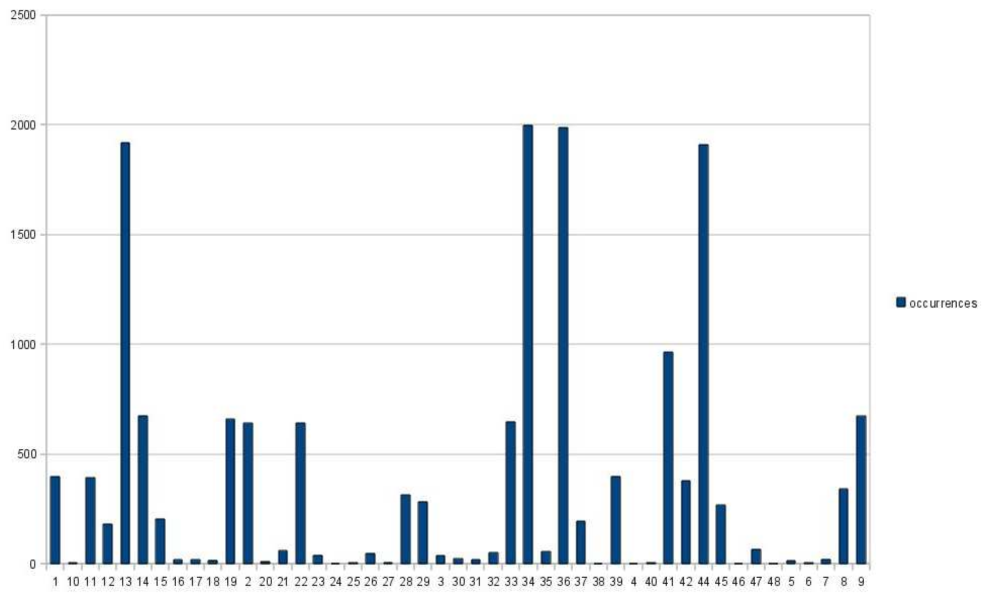

Figure 7.2: Distribution of labels of synthetic data set non-equally distributed

Figure 7.2 shows the chart of the labels' distributions for the synthetic data set non-equally distributed. Again the occurrences of each label are constructed with the help of the *rand* function in JAVA.

This time label 18 and 35 are the ones having patterns. Label 18 appears 17 times totally from which in 16 times it is preceded by label 31 which occurs 19 times in total. Label 35 appears 35 in total; 52 times it is preceded by label 21 which occurs 59 times in total.

In the following table the exact number of occurrences for each label are listed.

label	number of occurrences	ratio to maximum occurrence
1	398	46.15075376884422
10	5	3673.6
11	392	46.857142857142854
12	182	100.92307692307692
13	1919	9.571651902032308
14	675	27.21185185185185
15	205	89.6
16	18	1020.4444444444445
17	18	1020.4444444444445

label	number of occurrences	ratio to maximum occurrence
18	17	1080.4705882352941
19	658	27.914893617021278
2	640	28.7
20	11	1669.8181818181818
21	59	311.3220338983051
22	642	28.610591900311526
23	40	459.2
24	2	9184.0
25	4	4592.0
26	48	382.6666666666667
27	7	2624.0
28	314	58.496815286624205
29	284	64.67605633802818
3	38	483.36842105263156
30	22	834.9090909090909
31	19	966.7368421052631
32	50	367.36
33	648	28.34567901234568
34	1999	9.188594297148574
35	55	333.96363636363634
36	1988	9.23943661971831
37	195	94.1948717948718
38	1	18368.0
39	399	46.03508771929825
4	1	18368.0
40	6	3061.3333333333335
41	965	19.03419689119171
42	379	48.46437994722955
43	18368	1.0
44	1910	9.61675392670157
45	267	68.7940074906367
46	1	18368.0
47	67	274.14925373134326
48	1	18368.0
5	15	1224.5333333333333
6	4	4592.0
7	21	874.6666666666666
8	340	54.023529411764706
9	672	27.333333333333332

Appendix B - Real datasets

In this chapter the real-world data sets from the OEE-Analyser^{TM}are discussed in detail. This is done the same way as with the synthetic data sets in the previous chapter; first a chart is given which visualises the distribution of each label and second the detailed numbers of that distributions are listed.

7.3 Machine 62

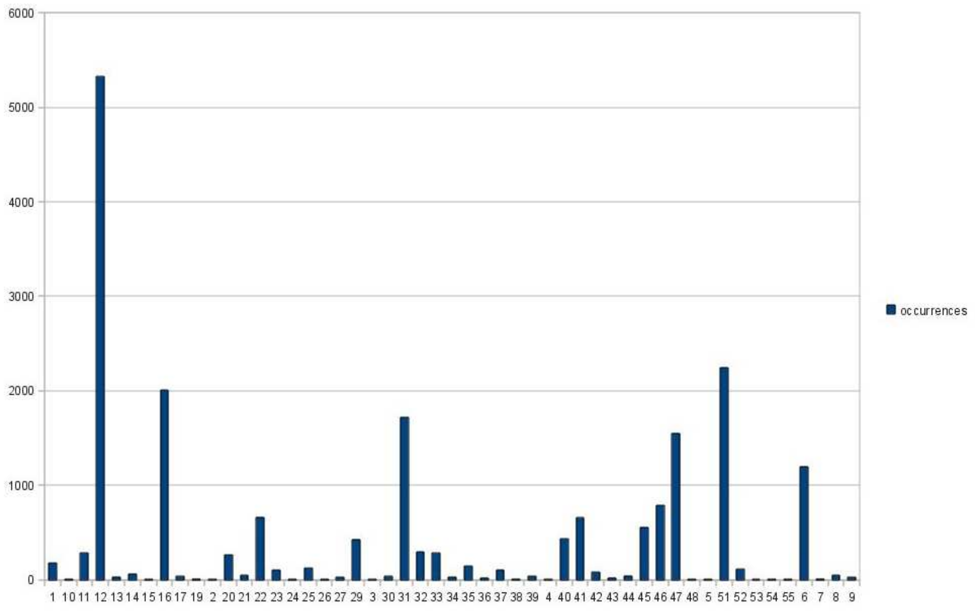

Figure 7.3: Distribution of labels of machine 62

Figure 7.3 shows the distribution of numbers of labels from machine 62. Except for that one label occurring more than 5000 times it seems to be very similar to the distribution of the synthetic data set non-equally distributed. In the following the detailed numbers of occurrences are given.

label	number of occurrences	ratio to maximum occurrence
1	177	30.073446327683616
10	7	760.4285714285714
11	282	18.875886524822697
12	5323	1.0
13	30	177.43333333333334
14	63	84.4920634920635
15	3	1774.3333333333333
16	2006	2.653539381854437
17	34	156.55882352941177
19	10	532.3
2	1	5323.0
20	264	20.16287878787879
21	46	115.71739130434783
22	663	8.028657616892911
23	104	51.18269230769231
24	4	1330.75
25	122	43.631147540983605
26	6	887.1666666666666
27	26	204.73076923076923
29	427	12.466042154566745
3	1	5323.0
30	34	156.55882352941177
31	1719	3.096567771960442
32	296	17.98310810810811
33	280	19.010714285714286
34	29	183.55172413793105
35	143	37.22377622377623
36	21	253.47619047619048
37	105	50.695238095238096
38	8	665.375
39	36	147.86111111111111
4	1	5323.0
40	431	12.350348027842227
41	654	8.13914373088685
42	79	67.37974683544304
43	22	241.95454545454547
44	42	126.73809523809524
45	557	9.556552962298024
46	787	6.7636594663278276
47	1551	3.4319793681495807
48	4	1330.75
5	5	1064.6
51	2247	2.368936359590565

label	number of occurrences	ratio to maximum occurrence
52	109	48.8348623853211
53	2	2661.5
54	7	760.4285714285714
55	7	760.4285714285714
6	1200	4.435833333333333
7	10	532.3
8	47	113.25531914893617
9	25	212.92

The data of machine 62 is used in different experiments. Amongst others there is a variation of the parameter N in some experiments. This variation leads to different labels being associated with the machine states. In the following an overview is given in which the associated labels for each value of N gets apparent. The labels given in the table above are the one from N=1.

	N=1	N=2	N=3
AALG	49	286	2375
ANFNST	24	184	1573
ANFST E	44	504	1923
AUSW	19	102	798
BESP	35	423	505
BetU St	4	231	964
DAB	7	166	884
DGMA	16	100	2129
ENTG	22	482	1212
EROB	51	296	357
FAUS	17	324	2419
FNSP	1	158	201
FOEDB	29	415	1143
FRTG Start	14	172	2226
GRK	30	575	1313
GSTAB	8	8	12
GUHB	27	566	1306
IH	20	242	1643
IHWZ	42	360	1770
KEBR	36	202	926
KEMA	54	378	454
KEML	48	368	444
KOB GST	43	597	2018
KOLB	34	270	1669
OFEN	40	125	2169

	N=1	N=2	N=3
OFF	25	26	29
OPTI	23	331	1733
PAUS	33	198	242
PRES	46	430	1848
PRGU	15	405	1820
PRGUE	3	456	2571
PROD	28	569	1986
REIN	41	207	1595
RUE/W/R	9	397	1124
RUE/W/R En	38	589	666
RUES	5	232	1627
RUESE	2	455	537
SAN E	10	167	1552
SAN ST	55	532	604
SCHI	11	546	1964
SCHM	12	550	1967
SCHMR	6	5	702
SCHT	21	555	1971
SORG	45	134	2182
SROB	37	47	1412
STEI	18	406	2504
STSA	31	34	2075
TABE	32	267	997
UBSP	47	432	517
UNBE	50	69	78
WAEi	26	337	403
WAIN	39	353	431
WALG	52	439	525
WART	13	240	304
WARTE	53	83	92

7.4 Machine 65

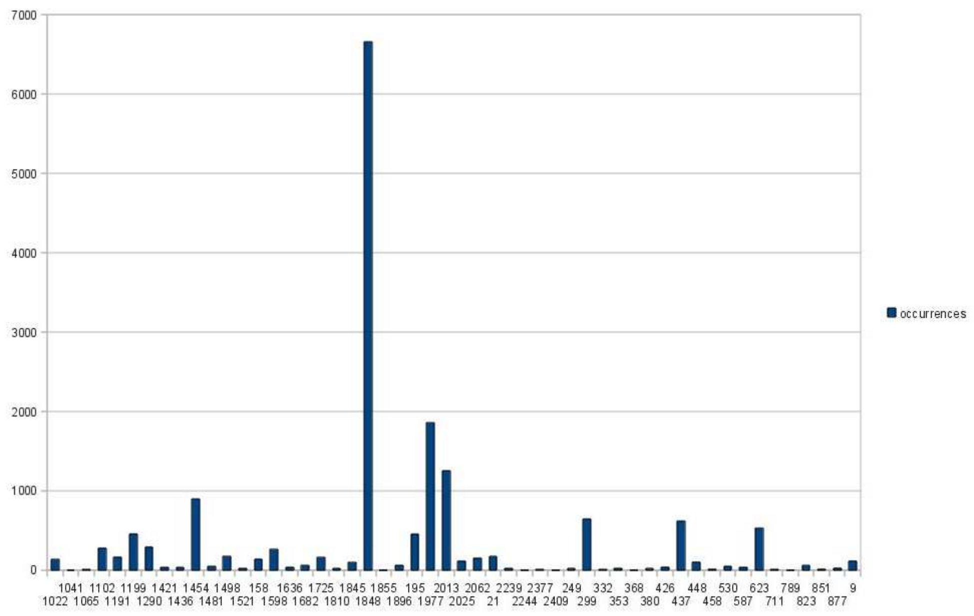

Figure 7.4: Distribution of labels of machine 65

In the following the exact numbers are listed again.

label	number of occurrences	ratio to maximum occurrence
1022	133	50.07518796992481
1041	2	3330.0
1065	9	740.0
1102	280	23.785714285714285
1191	167	39.880239520958085
1199	460	14.478260869565217
1290	296	22.5
1421	36	185.0
1436	35	190.28571428571428
1454	899	7.4082313681868746
1481	48	138.75
1498	171	38.94736842105263
1521	22	302.72727272727275
158	142	46.901408450704224

label	number of occurrences	ratio to maximum occurrence
1598	259	25.714285714285715
1636	38	175.26315789473685
1682	58	114.82758620689656
1725	158	42.151898734177216
1810	21	317.14285714285717
1845	95	70.10526315789474
1848	6660	1.0
1855	4	1665.0
1896	62	107.41935483870968
195	457	14.573304157549234
1977	1857	3.5864297253634896
2013	1256	5.302547770700637
2025	118	56.440677966101696
2062	154	43.246753246753244
21	171	38.94736842105263
2239	23	289.5652173913044
2244	1	6660.0
2377	7	951.4285714285714
2409	3	2220.0
249	19	350.5263157894737
299	648	10.277777777777779
332	6	1110.0
353	25	266.4
368	2	3330.0
380	18	370.0
426	41	162.4390243902439
437	621	10.72463768115942
448	105	63.42857142857143
458	16	416.25
530	51	130.58823529411765
587	37	180.0
623	529	12.589792060491494
711	12	555.0
789	1	6660.0
823	58	114.82758620689656
851	15	444.0
877	29	229.6551724137931
9	119	55.96638655462185

As there are only experiments made with N=3 for machine 65 there is no variation in labels. Nevertheless in the following the label assignment to machine states is given.

AALG	2199
ANFNST	1436
ANFST E	1810
AUSW	711
BESP	426
BetU E	2409
BetU St	851
DAB	789
DGMA	1977
ENTG	1102
EROB	299
FAUS	2239
FNSP	158
FOEDB	1041
FRTG Ende	1065
FRTG Start	2062
GRK	1199
GSTAB	9
GUHB	1191
IH	1498
IHWZ	1636
KEBR	823
KEMA	380
KEML	368
KOB GST	1896
KOLB	1521
OFEN	2013
OFF	21
OPTI	1598
PAUS	195
PRES	1725
PRGU	1682
PRGUE	2377
PROD	1872
REIN	1454
RUE/W/R	1022
RUE/W/R En	587
RUES	1481
RUESE	458
SAN E	1421
SAN ST	530
SCHI	1845
SCHM	1848

SCHMR	623
SCHT	1855
SORG	2025
SROB	1290
STAUS	2244
STEI	2324
TABE	877
UBSP	437
UNBE	64
WAEi	332
WAIN	353
WALG	448
WART	249
WARTE	68

Appendix C - Results in detail

In this appendix the exact numbers of experiments are given. Therefore the rankings and probabilities are given. Thereby

- *count* indicates how often a label occurred for a test-item,
- *rnk hit* is the average rank of a label when it appeared for a test-item,
- *rnk rest* is the overall average rank of a label,
- *prob hit* is the average probability of a label when it appeared for a test-item,
- *prob rest* is the overall average probability of a label
- *hi rnk hit* is the highest rank a label took when it appeared for a test-item,
- *hi rnk* is the highest rank a label tool when it not appeared for a test-item,
- *lo rnk* is the lowest rank a label took.

7.5 Synthetic data equally distributed

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	41	16.24	16.48	0.018418401	0.018424843	8	3	50
10	31	10.65	12.4	0.018951211	0.018796338	3	1	41
11	46	19.15	21.36	0.018539267	0.018336559	1	1	57
12	35	54.54	54.28	0.015410052	0.015500172	47	37	57
13	33	46.3	46.63	0.01645671	0.016456626	35	22	57
14	38	30.58	30.25	0.017508669	0.017490327	17	8	52
16	39	34.79	34.32	0.017231306	0.01725645	24	18	55
17	34	50.15	49.07	0.016228065	0.016282171	45	27	57
18	39	38.87	39.75	0.017016616	0.016917475	24	20	57
19	41	14.85	17.97	0.018623557	0.018415468	1	1	57
2	53	11.87	11.27	0.018867947	0.01894908	1	1	48
20	41	42.15	42.56	0.016803818	0.016776022	32	27	56
21	39	11.46	11.57	0.018793937	0.018837443	5	2	36
22	30	22.2	21.58	0.01793862	0.018041812	14	7	47
23	33	44.45	40.01	0.016614674	0.016899042	27	13	57
24	54	9.44	9.38	0.01900932	0.019045338	3	2	28
25	40	1.85	1.9	0.020282488	0.020228963	1	1	12
26	30	36.03	34.16	0.017178541	0.017267616	27	17	51
27	42	9.9	10.16	0.01903218	0.018998098	3	1	42
28	46	30.17	30.78	0.017426264	0.017458893	13	7	56
29	46	31.61	32.89	0.017418843	0.017338179	17	13	54
3	29	31.41	33.59	0.017423196	0.017293364	21	13	55
30	35	18.37	17.77	0.018329868	0.018313214	12	3	37
32	43	43.33	43.03	0.016744535	0.016766297	35	26	54
33	29	51.69	52.37	0.016027443	0.01587993	44	37	57
34	44	24.82	25.17	0.017758748	0.01779985	18	12	47
35	31	15.35	14.0	0.018520467	0.018647518	6	1	45
36	45	31.82	30.37	0.017417578	0.017474243	15	10	54
37	42	2.29	2.26	0.020117208	0.020146249	1	1	11
38	31	47.03	48.18	0.016448127	0.016363077	37	23	56
39	39	33.33	31.7	0.017325014	0.01740358	21	16	55
4	45	27.33	27.61	0.017685296	0.017656157	11	6	52
40	31	34.81	34.6	0.01727643	0.01724684	20	17	52
41	33	26.52	26.28	0.017820766	0.017757362	12	6	56
42	41	38.29	38.22	0.017049495	0.017018035	20	17	56
43	37	10.95	52.94	0.035496723	0.009532571	1	1	57
44	39	37.9	37.1	0.01707625	0.017112128	31	23	54
45	39	28.85	27.81	0.017646896	0.017627535	21	14	48
46	43	51.19	50.93	0.015905695	0.015962299	37	23	57
47	33	42.73	43.48	0.016759798	0.016732804	36	21	54
48	48	48.63	48.63	0.016416023	0.016340293	39	32	56

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
49	47	4.89	5.19	0.019536382	0.019535705	1	1	18
5	53	14.6	15.03	0.0186495	0.018599227	2	1	45
50	37	41.41	42.18	0.016916687	0.016808182	28	25	56
51	39	52.69	52.69	0.015963653	0.01591417	45	36	57
52	31	23.0	21.79	0.017983874	0.018020436	15	5	41
53	38	5.03	4.98	0.019522801	0.019575214	2	1	20
54	42	9.36	9.7	0.019033356	0.018996233	5	2	28
55	39	16.46	15.52	0.018404597	0.01848627	10	4	35
56	41	18.98	18.5	0.018241193	0.01825494	14	4	38
57	44	25.18	24.87	0.017851295	0.017847072	14	6	55
58	47	50.55	51.23	0.015966546	0.015843289	30	21	57
59	34	6.68	7.55	0.019412838	0.019260956	2	1	26
6	42	30.21	31.78	0.017477447	0.017406985	20	13	55
7	29	12.93	14.16	0.018685017	0.018593268	8	4	32
8	30	47.17	47.52	0.016372116	0.016410286	34	23	57
9	43	35.37	35.27	0.017226212	0.017212374	28	18	48

7.6 Synthetic data non equally distributed

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	64	13.98	13.96	0.02347191	0.023458097	13	12	17
11	45	12.76	12.76	0.024749856	0.024729885	12	12	16
12	26	21.88	21.91	0.010035277	0.009998254	20	19	24
13	260	2.95	2.88	0.116870366	0.11724668	1	1	5
14	97	7.18	7.38	0.04094426	0.040815804	6	6	11
15	33	20.85	20.71	0.010901642	0.011039787	20	19	23
16	3	35.33	33.08	6.2630937E-4	0.0010405619	31	21	46
17	1	42.0	34.89	2.34735E-4	9.261885E-4	42	20	46
18	3	16.0	38.24	0.022868035	7.084805E-4	9	9	46
19	96	7.48	7.44	0.040890608	0.04071502	6	6	11
2	89	10.69	10.75	0.037874725	0.03770167	6	6	12
20	2	34.0	34.57	6.97135E-4	6.2831695E-4	33	29	39
21	13	24.15	24.22	0.00377679	0.0038204333	24	23	28
22	103	9.76	9.78	0.038403396	0.03854183	6	6	12
23	4	28.25	27.84	0.0023390176	0.0023661645	27	25	31
26	6	26.33	26.46	0.0026098231	0.0026639048	26	25	31
27	2	37.0	39.28	4.4918148E-4	3.5831245E-4	36	32	44
28	47	17.06	17.01	0.019050214	0.019075356	17	16	19
29	54	18.61	18.64	0.014644032	0.0146345785	18	17	22
3	3	27.67	27.72	0.0023512335	0.0023570715	27	24	34
30	6	32.33	33.76	8.6383056E-4	7.3523185E-4	31	26	40
31	3	30.33	30.16	0.0011769066	0.001279235	30	29	34
32	8	25.38	25.54	0.0030355875	0.0029850886	25	24	29
33	107	8.86	8.86	0.03933485	0.039342187	6	6	12
34	272	1.51	1.5	0.120906785	0.12080914	1	1	5
35	12	4.33	30.68	0.09712201	0.0028474857	1	1	46
36	293	1.82	1.82	0.12068765	0.120589435	1	1	5
37	24	20.5	20.46	0.011196534	0.011167285	20	19	23
39	54	12.76	12.52	0.02491871	0.02523719	12	12	17
41	137	5.0	5.01	0.05818919	0.058194384	5	4	6
42	53	14.92	14.91	0.02268741	0.022690216	14	12	17
44	256	3.79	3.81	0.11451875	0.11429803	2	1	5
45	33	18.48	18.49	0.014808574	0.01485331	18	18	20
47	8	23.13	23.11	0.0055160336	0.0054167295	23	22	25
5	1	31.0	32.91	0.0013242	8.406924E-4	31	26	39
7	4	33.5	33.34	7.658145E-4	7.7279826E-4	30	28	40
8	41	16.1	15.98	0.020827515	0.020933904	15	12	18
9	83	6.76	6.89	0.041399833	0.04123792	6	6	12

7.7 Machine 62

7.7.1 K=30, N=1, no IDF, E = 1

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	59	16.22	17.79	0.011015005	0.008507028	5	4	34
11	65	11.0	12.63	0.018880881	0.016587222	5	3	37
12	1203	1.19	1.39	0.30906543	0.24994291	1	1	9
14	2	25.5	35.11	0.009159606	0.0013156764	16	4	50
16	281	3.53	4.35	0.110846445	0.08801599	1	1	17
19	3	45.0	43.11	3.964537E-4	6.3150085E-4	43	6	50
20	44	8.98	14.88	0.06539071	0.015048013	1	1	30
21	18	24.61	27.4	0.0024907666	0.0022900633	17	12	40
22	155	7.19	8.3	0.0453102	0.03411928	4	2	19
23	20	18.15	19.89	0.009182677	0.0061671766	10	5	32
24	1	42.0	40.31	9.09156E-4	4.920507E-4	42	34	46
25	18	16.06	19.07	0.013292966	0.0070127905	3	1	32
27	3	25.67	30.61	0.0048272433	0.0021007294	20	4	43
29	357	18.23	18.66	0.0077314125	0.0074325316	7	2	39
30	3	25.33	25.72	0.00231901	0.0025429158	23	17	36
31	486	3.84	4.25	0.10445312	0.086587735	1	1	24
32	85	11.82	12.55	0.019773368	0.016190587	5	4	32
33	74	8.14	15.66	0.04528512	0.012191676	1	1	33
34	3	25.67	27.29	0.0026369735	0.002245601	23	16	41
35	56	16.68	18.38	0.010899559	0.006790939	9	6	28
36	6	30.33	31.83	0.0014563538	0.0013618151	29	19	41
37	34	18.26	18.41	0.0064905826	0.0061988197	9	8	36
38	1	28.0	38.56	0.00216489	6.45747E-4	28	13	46
39	12	25.0	28.42	0.003912834	0.0021892993	15	12	40
40	84	9.62	9.68	0.026184997	0.025100073	6	4	20
41	165	6.78	10.16	0.058391802	0.030025177	1	1	34
42	15	20.87	22.55	0.0058888993	0.004514415	16	8	37
43	4	29.25	30.37	0.002638425	0.0016041243	27	19	42
44	5	15.4	31.48	0.021003464	0.0023319414	6	2	47
45	295	6.83	15.98	0.073821574	0.016860258	1	1	38
46	173	6.4	8.13	0.059553925	0.039157495	1	1	34
47	441	3.77	6.61	0.15517029	0.07462353	1	1	23
48	1	38.0	40.42	6.1671E-4	4.913858E-4	38	31	47
51	577	2.8	2.98	0.13570529	0.118561305	1	1	11
52	21	17.14	17.4	0.01014407	0.0068716328	9	7	32
54	3	37.33	38.82	0.00139455	5.591995E-4	34	31	50
6	277	5.07	5.84	0.07612375	0.05722388	1	1	21
7	1	50.0	42.5	1.35897E-4	5.849211E-4	50	11	50
9	5	34.0	34.79	0.0013329316	0.0012455418	33	3	50

7.7.2 K=30, N=1, IDF, E = 1

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	59	15.54	18.51	0.011730533	0.008018025	9	6	36
11	65	11.52	12.56	0.01828558	0.01604745	6	4	30
12	1203	1.19	1.38	0.32170987	0.26770505	1	1	9
14	2	21.0	33.09	0.0145543	0.0014095363	10	7	43
16	281	3.7	4.71	0.110662736	0.078428	1	1	20
19	3	43.0	44.24	3.9262566E-4	8.8502135E-4	39	1	50
20	44	8.18	13.77	0.053203795	0.016203132	1	1	34
21	19	21.42	26.94	0.004111712	0.0024147236	14	12	35
22	155	7.1	8.43	0.046892665	0.033370823	2	1	21
23	20	17.5	20.41	0.010118714	0.006333709	9	1	35
24	1	40.0	41.46	8.45182E-4	4.9063016E-4	40	31	46
25	18	18.11	18.71	0.009311404	0.0071306117	6	2	34
27	3	30.0	32.0	0.002359417	0.001716267	24	8	45
29	357	16.63	18.74	0.00970973	0.006935291	6	6	39
30	3	23.67	26.07	0.0023411198	0.002589651	20	17	37
31	486	3.36	4.0	0.1112861	0.08710421	1	1	15
32	85	11.84	12.7	0.019042017	0.015993519	4	3	32
33	74	7.62	15.15	0.050360996	0.013784075	1	1	40
34	3	27.33	27.66	0.0022453668	0.002176236	25	18	37
35	56	16.27	17.86	0.010535993	0.007213206	8	6	30
36	6	31.17	30.77	0.0019037226	0.0015557829	28	16	40
37	34	18.94	19.19	0.0063962718	0.0057369545	13	8	30
38	1	7.0	38.88	0.0364003	6.3077617E-4	7	7	45
39	12	22.58	28.62	0.0048805554	0.0022957462	15	7	40
40	84	9.63	9.81	0.028393194	0.02595225	3	1	24
41	165	7.79	10.12	0.047873624	0.02859599	2	1	27
42	15	20.07	21.41	0.0066953176	0.0049833464	16	9	37
43	4	29.75	29.18	0.0019126075	0.0018557394	27	20	38
44	5	20.2	32.72	0.012997238	0.0020747713	10	3	50
45	295	7.52	15.17	0.09855068	0.01799841	1	1	36
46	173	6.49	7.62	0.059662137	0.040031403	1	1	23
47	441	4.2	7.19	0.13701287	0.063616686	1	1	25
48	1	43.0	40.58	5.01199E-4	5.1050965E-4	43	35	45
51	577	2.74	3.2	0.14138682	0.110899925	1	1	13
52	21	15.67	19.75	0.016728716	0.0059114317	4	2	34
54	3	38.33	38.47	0.001238339	6.189348E-4	34	27	49
6	277	4.5	5.25	0.08263445	0.0627188	1	1	16
7	1	46.0	46.23	2.41841E-4	3.7439176E-4	46	15	50
9	5	32.6	31.84	0.0017543469	0.0016594426	30	4	45

7.7.3 K=30, N=1, no IDF, E = 2

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	59	16.0	17.91	0.011908074	0.008480641	5	4	37
11	65	11.09	12.54	0.018702293	0.016274761	5	3	36
12	1203	1.19	1.42	0.31351262	0.25008926	1	1	9
14	2	25.5	35.89	0.009716357	0.0011782077	16	4	50
16	281	3.41	4.26	0.11424325	0.09003476	1	1	18
19	3	45.0	42.77	4.4540735E-4	7.011049E-4	43	2	50
20	44	8.05	15.04	0.08331654	0.014739875	1	1	31
21	19	24.21	27.08	0.0025349348	0.0022922896	16	12	40
22	155	7.16	8.3	0.04535632	0.033961516	3	1	21
23	20	18.25	19.63	0.009711733	0.0061831027	10	5	32
24	1	42.0	39.97	0.00108051	4.9798796E-4	42	32	46
25	18	15.94	19.7	0.0153695345	0.006685966	3	1	33
27	3	25.67	30.69	0.00513968	0.0020941196	20	4	44
29	357	18.06	18.64	0.007603182	0.007246921	7	4	38
30	3	24.33	25.58	0.0023342867	0.0025381248	21	16	36
31	486	3.84	4.29	0.10450554	0.08634935	1	1	25
32	85	11.79	12.57	0.01973992	0.01602516	5	5	33
33	74	7.46	16.55	0.05939559	0.011570003	1	1	35
34	3	24.67	27.14	0.002698213	0.002234079	23	16	42
35	56	16.05	18.16	0.011744863	0.00691063	7	5	27
36	6	30.83	31.56	0.0014530554	0.0013572738	30	20	42
37	34	18.12	18.34	0.006657267	0.006170752	9	8	35
38	1	25.0	38.8	0.00406661	6.325889E-4	25	17	47
39	12	23.42	28.69	0.0049950033	0.0022041118	11	9	41
40	84	9.36	9.62	0.026726456	0.025197523	6	4	22
41	165	6.74	10.29	0.062332857	0.02883097	1	1	30
42	15	20.0	22.29	0.0066317995	0.0045232503	15	8	37
43	4	29.5	30.11	0.0025602377	0.0016062683	26	19	43
44	5	14.2	32.97	0.031265795	0.0021423155	4	2	49
45	295	6.85	15.68	0.07344967	0.016579712	1	1	37
46	173	6.31	8.15	0.059486713	0.03915854	1	1	34
47	441	3.69	6.64	0.15616381	0.0733867	1	1	24
48	1	36.0	40.06	5.62797E-4	5.047543E-4	36	30	47
51	577	2.78	2.99	0.1368579	0.11915179	1	1	18
52	21	17.1	17.27	0.010781474	0.0069441106	9	5	33
54	3	36.0	38.66	0.0013165455	5.68184E-4	32	31	50
6	277	5.06	5.85	0.077054694	0.057458356	1	1	24
7	1	49.0	42.27	1.18946E-4	5.6846347E-4	49	11	50
9	5	32.0	34.17	0.0018762766	0.0012736904	27	3	48

7.7.4 K=30, N=1, no IDF, E = 3

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1	59	15.97	17.94	0.012459078	0.008440086	5	4	38
11	65	11.18	12.54	0.018825617	0.016026836	7	3	35
12	1203	1.19	1.44	0.31644994	0.25070927	1	1	12
14	2	25.0	36.42	0.010864495	0.0011154306	15	5	50
16	281	3.38	4.24	0.11480725	0.09041501	1	1	19
19	3	45.33	42.27	4.6560704E-4	7.1863364E-4	42	2	50
20	44	7.45	15.19	0.09205907	0.014216684	1	1	32
21	19	24.16	26.85	0.0025897664	0.0023049926	16	13	38
22	155	7.13	8.28	0.045291837	0.03393828	3	1	22
23	20	17.95	19.46	0.010108447	0.0062047327	11	6	32
24	1	43.0	39.75	0.0012379	4.987051E-4	43	32	46
25	18	15.61	19.88	0.016574135	0.0066180117	2	1	34
27	3	26.0	30.55	0.004994443	0.0020803967	21	5	44
29	357	17.93	18.54	0.007639746	0.0072302544	7	5	38
30	3	23.33	25.49	0.00236207	0.0025431712	21	15	38
31	486	3.84	4.27	0.104551405	0.0867221	1	1	25
32	85	11.8	12.57	0.019742414	0.015937114	4	4	34
33	74	7.0	17.28	0.068681166	0.011136994	1	1	37
34	3	24.33	27.07	0.0028342668	0.002223589	23	16	42
35	56	16.21	18.18	0.011670574	0.0068670567	7	6	28
36	6	30.5	31.42	0.0014879679	0.0013607763	28	19	42
37	34	17.85	18.37	0.0066917674	0.0060975575	7	7	35
38	1	16.0	39.47	0.0116955	6.122715E-4	16	11	50
39	12	23.33	28.96	0.006187161	0.0022157372	9	7	42
40	84	9.21	9.57	0.027124973	0.025338797	6	4	24
41	165	6.67	10.35	0.06576489	0.02852807	1	1	29
42	15	19.27	22.23	0.0070971576	0.004471399	15	8	36
43	4	29.75	29.97	0.0025441626	0.0016023314	26	20	44
44	5	14.2	33.97	0.042299647	0.0020996553	3	1	49
45	295	6.79	15.21	0.073376104	0.016839173	1	1	37
46	173	6.22	8.09	0.059534565	0.039380383	1	1	34
47	441	3.7	6.69	0.15705676	0.07255858	1	1	28
48	1	38.0	39.73	4.93882E-4	5.1176344E-4	38	29	47
51	577	2.77	3.0	0.1376296	0.118903376	1	1	20
52	21	17.81	17.14	0.010216745	0.006973415	11	6	32
54	3	36.33	38.36	0.0012213796	5.730431E-4	33	31	50
6	277	5.05	5.87	0.077225804	0.057393175	1	1	26
7	1	48.0	42.09	1.0739E-4	5.569632E-4	48	12	50
9	5	30.2	33.82	0.002447713	0.0012851959	26	4	46

7.7.5 K=30, N=2, no IDF, E = 1

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
100	281	3.22	4.19	0.11809773	0.08895014	1	1	19
102	3	43.33	43.5	5.18922E-4	8.1905606E-4	42	9	50
125	84	9.6	9.7	0.02834346	0.027408708	4	2	28
134	295	9.61	14.61	0.04274919	0.015855314	1	1	29
158	59	14.37	15.83	0.014203187	0.011833607	4	1	35
166	1	47.0	43.59	0.00305522	7.3070073E-4	47	23	50
172	2	31.5	36.31	0.0052873464	0.0013985289	26	9	50
184	1	43.0	41.13	0.00104971	7.641148E-4	43	32	47
198	74	6.57	16.53	0.08093588	0.014567796	1	1	35
202	6	30.17	31.7	0.001640515	0.0017446481	29	20	44
207	165	6.82	10.79	0.07405537	0.031742062	1	1	29
242	44	10.3	14.8	0.05155094	0.016430281	1	1	31
26	18	13.94	18.62	0.02538245	0.008289937	2	1	35
267	85	12.41	13.29	0.017601991	0.015610943	5	4	30
270	3	25.67	26.74	0.0031909433	0.0028506245	23	15	40
296	577	2.93	3.3	0.12931678	0.11110032	1	1	15
331	20	16.6	20.71	0.016786072	0.0066650286	3	3	35
34	486	3.93	4.47	0.10488326	0.08416824	1	1	23
353	12	24.0	28.22	0.0070748893	0.00297113	10	7	45
360	15	21.33	21.99	0.0064983848	0.006898382	14	1	37
368	1	39.0	40.23	9.26176E-4	7.923522E-4	39	32	46
378	3	36.0	38.63	0.0017131713	8.87273E-4	31	27	49
397	5	32.8	34.19	0.002279915	0.0016304638	31	4	49
415	357	18.55	18.82	0.008784693	0.007971122	6	4	38
423	56	16.25	18.36	0.0120978	0.0073747328	4	3	29
430	173	7.22	8.29	0.047905408	0.037033137	2	1	29
432	441	4.19	7.24	0.12511289	0.06988443	1	1	31
439	21	20.48	18.71	0.00766144	0.0072714984	12	6	35
47	34	17.68	19.61	0.008315065	0.0065158615	6	3	34
482	155	7.85	8.68	0.04115494	0.0330036	3	1	23
5	277	5.34	5.89	0.07563554	0.059547786	1	1	18
504	5	17.0	31.69	0.019589465	0.0031780053	7	2	49
546	65	11.97	13.03	0.018325845	0.017657433	6	1	35
550	1203	1.28	1.54	0.29409552	0.24129374	1	1	17
555	19	23.84	26.53	0.00456284	0.0030132225	16	10	36
566	3	25.0	30.02	0.006047456	0.002666049	18	6	44
575	3	23.67	25.57	0.0023512067	0.0031698092	23	14	38
589	1	20.0	37.62	0.00951089	0.0010347115	20	12	46
597	4	31.25	30.34	0.0034324075	0.002013553	29	15	47

7.7.6 K=30, N=2, IDF, E = 1

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
100	281	3.55	4.41	0.10867579	0.084287256	1	1	14
102	3	43.0	44.15	5.3614E-4	7.89279E-4	38	5	50
125	84	9.38	9.58	0.028424624	0.027330354	4	1	21
134	295	7.77	14.8	0.122698575	0.018624207	1	1	36
158	59	14.83	17.39	0.013294931	0.008703658	7	6	33
166	1	50.0	48.22	0.00171733	5.252571E-4	50	24	50
172	2	31.0	34.33	0.0049827853	0.0015514193	27	7	47
184	1	41.0	42.4	0.00200958	6.9849833E-4	41	30	48
198	74	5.88	17.48	0.096767314	0.013438139	1	1	36
202	6	32.67	31.01	0.0015483489	0.0019178591	29	16	42
207	165	7.98	11.72	0.06597308	0.02679263	1	1	28
242	44	8.95	13.78	0.054641984	0.018105894	1	1	33
26	18	14.22	17.56	0.019106098	0.008466635	2	2	33
267	85	12.41	13.06	0.017166331	0.015083741	7	5	31
270	3	27.33	27.15	0.0026899998	0.0026318328	26	17	37
296	577	2.97	3.33	0.12858644	0.108136	1	1	11
331	20	16.6	21.67	0.017304119	0.0056918464	3	3	37
34	486	3.6	4.06	0.10449362	0.08719588	1	1	18
353	12	22.5	27.8	0.008751279	0.0030723098	7	4	42
360	15	19.53	20.66	0.0074383756	0.0061165765	14	3	37
368	1	41.0	40.13	0.00114895	7.702137E-4	41	35	47
378	3	38.0	38.15	0.0013097808	9.1250247E-4	34	25	46
397	5	31.6	31.11	0.003476356	0.001972233	29	9	43
415	357	15.12	18.04	0.016024066	0.009216924	2	2	39
423	56	16.39	18.13	0.011107482	0.007385604	10	7	29
430	173	7.02	7.93	0.046645943	0.03856049	2	1	24
432	441	4.4	7.13	0.13402574	0.0692834	1	1	27
439	21	18.62	20.27	0.008857663	0.0059504462	10	4	36
47	34	18.03	19.25	0.0065150405	0.0059223888	11	7	30
482	155	7.85	8.64	0.04125815	0.032602858	3	1	23
5	277	5.04	5.55	0.07613949	0.061599575	1	1	16
504	5	17.8	31.41	0.02010085	0.002630011	7	1	48
546	65	11.85	12.5	0.017717006	0.01742221	6	1	26
550	1203	1.25	1.52	0.2982157	0.24832918	1	1	11
555	19	19.84	27.2	0.0053759585	0.002793828	12	8	36
566	3	30.0	31.73	0.00294801	0.002185916	23	7	44
575	3	23.67	25.71	0.0026705898	0.0030666322	20	13	35
589	1	4.0	39.43	0.0631743	9.6031243E-4	4	4	47
597	4	28.25	28.62	0.0023512999	0.002303836	25	19	38

7.7.7 K=30, N=2, no IDF, E = 3

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
100	281	3.15	4.17	0.12305855	0.091351636	1	1	16
102	3	43.0	42.39	8.487967E-4	8.497441E-4	41	8	50
125	84	9.5	9.71	0.029312849	0.026957272	3	2	28
134	295	8.69	14.25	0.05065411	0.016968904	1	1	30
158	59	13.83	16.0	0.01651966	0.011796968	4	1	33
166	1	46.0	43.18	0.00329692	7.3313975E-4	46	23	50
172	2	28.0	37.15	0.0076396456	0.0013352928	21	9	50
184	1	44.0	41.1	0.0013904	7.7659753E-4	44	25	49
198	74	6.0	18.1	0.0925243	0.01289229	1	1	36
202	6	30.5	31.57	0.0016693302	0.001754109	28	20	45
207	165	6.87	11.05	0.08534574	0.029990852	1	1	29
242	44	10.02	15.09	0.05611849	0.01600734	1	1	32
26	18	14.89	19.17	0.022901863	0.007934132	2	2	33
267	85	12.32	13.19	0.01753913	0.01586016	5	4	35
270	3	25.67	26.51	0.0035360835	0.0029078405	25	14	39
296	577	2.98	3.38	0.13108051	0.1114676	1	1	13
331	20	16.4	20.24	0.017338986	0.006855482	3	3	36
34	486	3.82	4.37	0.10467109	0.084446065	1	1	19
353	12	23.17	29.18	0.0098855365	0.0029159223	5	4	46
360	15	20.53	21.86	0.0067048655	0.0063688033	15	2	36
368	1	38.0	39.17	8.51817E-4	8.3796127E-4	38	30	46
378	3	35.67	38.61	0.0019364739	9.044016E-4	29	26	49
397	5	29.2	33.09	0.0029770941	0.0017477149	25	3	47
415	357	18.58	18.92	0.008873643	0.007930863	6	4	38
423	56	15.75	17.95	0.012781391	0.0076094335	7	3	28
430	173	6.86	8.08	0.049549438	0.038162418	2	1	28
432	441	4.11	7.14	0.12890391	0.0703989	1	1	31
439	21	19.52	18.08	0.0081280125	0.007490893	12	6	34
47	34	17.44	19.46	0.008079087	0.006535057	6	5	33
482	155	7.96	8.79	0.041757178	0.03258782	3	2	24
5	277	5.26	5.75	0.07751434	0.061278258	1	1	18
504	5	14.2	33.37	0.03802264	0.00261048	3	1	49
546	65	11.75	12.91	0.019027803	0.017482001	6	1	32
550	1203	1.29	1.62	0.29441527	0.23598726	1	1	17
555	19	23.79	26.27	0.0046214275	0.0030678038	16	10	35
566	3	25.0	30.07	0.0057487036	0.002609242	18	2	47
575	3	22.67	25.51	0.0026834833	0.0032069038	22	13	39
589	1	13.0	38.64	0.0246346	9.54635E-4	13	9	50
597	4	31.5	29.93	0.003315855	0.0020665878	27	18	45

7.7.8 K=30, N=2, IDF, E = 3

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
100	281	3.48	4.3	0.11163984	0.08677512	1	1	13
102	3	42.33	43.81	7.2832167E-4	7.902896E-4	37	7	50
125	84	9.38	9.57	0.028346712	0.026871793	3	2	20
134	295	7.52	14.41	0.1113041	0.018327171	1	1	36
158	59	13.92	17.39	0.014784195	0.008855222	7	5	34
166	1	50.0	48.05	0.00172229	5.2964065E-4	50	20	50
172	2	26.5	34.58	0.007055145	0.001646624	21	6	49
184	1	40.0	42.18	0.00266875	6.9905183E-4	40	24	48
198	74	5.81	18.53	0.10083153	0.012155976	1	1	38
202	6	32.67	31.1	0.0017365563	0.0018562059	28	18	42
207	165	7.98	11.69	0.0694123	0.02577325	1	1	27
242	44	8.45	13.92	0.058313895	0.017956747	1	1	32
26	18	15.0	17.8	0.020595727	0.008449989	2	1	32
267	85	12.27	12.87	0.016968546	0.015431252	8	5	31
270	3	28.0	27.44	0.002886527	0.002602861	27	17	36
296	577	2.85	3.22	0.12987745	0.110340096	1	1	12
331	20	15.8	21.06	0.016967917	0.00588038	3	3	35
34	486	3.53	4.06	0.10456436	0.087948345	1	1	18
353	12	22.75	28.35	0.010110551	0.0028990728	4	4	42
360	15	19.2	20.81	0.0074146553	0.006040017	12	3	40
368	1	40.0	39.53	0.00106661	7.751752E-4	40	33	45
378	3	36.0	38.59	0.0018249099	9.11448E-4	33	20	48
397	5	30.2	30.55	0.0038363896	0.0020890369	29	8	41
415	357	15.13	18.36	0.016042853	0.008843856	4	2	38
423	56	16.11	17.78	0.0115700355	0.007586866	10	7	29
430	173	6.85	7.9	0.04773054	0.039139733	2	1	24
432	441	4.3	7.06	0.1327201	0.06659905	1	1	27
439	21	18.24	19.6	0.009278352	0.006165326	10	5	34
47	34	18.29	19.5	0.006430832	0.0057018986	12	9	33
482	155	7.85	8.66	0.04131722	0.032210622	2	2	22
5	277	4.99	5.44	0.07811064	0.06409566	1	1	15
504	5	16.6	33.33	0.03678	0.0024400337	3	1	48
546	65	11.72	12.73	0.018468231	0.016796105	3	1	29
550	1203	1.25	1.53	0.2977843	0.24626236	1	1	11
555	19	19.95	26.63	0.0056829345	0.002932737	10	8	36
566	3	32.67	31.63	0.0027090332	0.0021493398	27	9	43
575	3	23.33	25.74	0.0026451934	0.0030691621	19	15	36
589	1	1.0	40.41	0.180557	8.396533E-4	1	1	47
597	4	30.0	28.85	0.00370809	0.0022465906	28	19	38

7.7.9 K=30, N=3, no IDF, E = 1

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1124	5	32.0	33.33	0.0027243102	0.001888115	30	9	47
1143	357	20.39	19.59	0.007745415	0.008021714	4	1	36
1212	155	8.44	8.87	0.03529335	0.03145204	2	1	23
1306	3	28.67	30.9	0.0036627932	0.0025902053	21	11	48
1313	3	24.33	24.6	0.00248589	0.0036984598	23	15	34
1412	34	17.97	18.73	0.008078982	0.007797263	7	2	35
1573	1	43.0	41.39	0.00104473	9.505028E-4	43	33	46
1595	165	7.08	10.43	0.060398713	0.031275287	1	1	29
1643	44	11.55	14.5	0.035650857	0.015705172	1	1	30
1669	3	25.67	26.04	0.0034514733	0.0033260554	20	14	39
1733	20	16.6	20.57	0.015385384	0.0073712626	5	2	35
1770	15	20.53	22.04	0.006875855	0.00679303	15	3	35
1848	173	7.43	8.53	0.044951063	0.036249474	2	1	29
1923	5	20.6	31.97	0.015886622	0.0033250265	4	1	50
1964	65	12.42	13.08	0.016806116	0.016288461	6	3	27
1967	1203	1.37	1.5	0.2694871	0.24414629	1	1	13
1971	19	24.05	26.03	0.0040362626	0.0034228275	16	11	37
201	59	13.76	15.8	0.016149152	0.011312345	4	3	30
2018	4	31.5	29.73	0.0031936134	0.0024145406	29	15	42
2075	486	4.09	4.32	0.092172146	0.084570535	1	1	15
2129	281	3.68	4.17	0.11186209	0.0946677	1	1	16
2169	84	9.67	9.63	0.026500825	0.026872994	2	1	26
2182	295	11.29	16.36	0.028948225	0.013571558	2	1	34
2226	2	32.0	37.47	0.0029192201	0.0014225753	30	11	50
242	74	7.38	16.2	0.072567455	0.015769364	1	1	35
29	18	15.5	18.14	0.017657261	0.009052532	4	1	32
357	577	3.35	3.43	0.117494926	0.11778512	1	1	17
431	12	23.83	28.58	0.0063702655	0.0029440396	10	9	44
444	1	40.0	40.16	7.09125E-4	9.952817E-4	40	31	46
454	3	35.33	39.98	0.002660037	0.0010718377	27	25	50
505	56	16.46	18.59	0.011619484	0.0074681845	7	5	31
517	441	5.09	7.16	0.08391473	0.0580192	1	1	27
525	21	20.1	18.98	0.0063701486	0.0077697597	16	5	35
666	1	18.0	36.72	0.00839649	0.0013146203	18	11	46
702	277	5.35	5.4	0.07067999	0.06973554	1	1	20
798	3	43.33	42.7	7.736163E-4	9.646069E-4	41	17	50
884	1	47.0	45.75	0.00106159	8.112167E-4	47	32	50
926	6	30.5	31.69	0.001869565	0.002007122	29	16	46
997	85	12.96	13.19	0.014637523	0.015751766	6	2	27

7.8 Machine 65

7.8.1 K=30, N=3, IDF, E = 3

label	count	rnk hit	rnk rest	prob hit	prob rest	hi rnk hit	hi rnk	lo rnk
1022	38	15.39	19.71	0.021340128	0.009506838	1	1	37
1041	1	49.0	49.25	0.00102251	6.351222E-4	49	35	50
1102	42	9.64	10.95	0.026142478	0.021051712	2	2	36
1191	34	11.21	15.46	0.022888443	0.012489685	4	3	33
1199	140	8.04	9.05	0.028964233	0.026764065	3	2	28
1290	31	10.32	11.08	0.023497265	0.020622117	2	2	27
1421	16	5.38	36.85	0.2838963	0.0042724586	1	1	43
1436	3	23.67	27.85	0.012332614	0.0041290675	13	2	44
1454	185	3.64	5.62	0.1009643	0.059781272	1	1	34
1481	9	23.22	30.22	0.018983705	0.004007688	4	1	45
1498	21	16.05	19.75	0.018381711	0.011689994	5	1	36
1521	11	33.0	33.06	0.0017244699	0.0020605864	28	21	48
158	52	16.37	19.13	0.01155432	0.008543999	6	5	38
1598	58	11.6	12.39	0.024203015	0.017829731	4	2	31
1636	6	22.83	25.31	0.008466663	0.0043920637	15	7	44
1682	14	18.36	41.82	0.038498156	0.0016859878	1	1	50
1725	37	14.41	14.93	0.012032895	0.012116789	7	1	36
1810	5	22.8	34.37	0.03811567	0.0029142639	2	1	44
1845	19	21.89	22.49	0.0063998443	0.0061303293	12	3	45
1848	1156	1.06	1.7	0.42140484	0.30316722	1	1	14
1896	16	20.69	21.46	0.005590157	0.0058255414	14	9	37
195	91	5.26	10.37	0.08742543	0.027002785	1	1	30
1977	427	2.57	2.79	0.120043054	0.109552324	1	1	14
2013	306	3.26	3.97	0.098816685	0.07964777	1	1	26
2025	40	17.93	21.22	0.013083331	0.007317561	4	2	37
2062	65	8.48	28.67	0.07035491	0.0039577154	2	2	45
21	36	12.39	14.39	0.023008816	0.013987809	2	2	35
2239	3	29.67	28.21	0.0030967232	0.0031732686	24	17	39
2377	1	1.0	46.22	0.111964	0.0010755095	1	1	49
2409	1	48.0	48.01	2.70358E-4	7.462332E-4	48	10	50
249	1	36.0	38.38	0.00114652	0.0019383429	36	3	50
299	139	6.43	6.33	0.044088043	0.042637445	1	1	18
332	2	45.0	41.0	0.004090145	0.0010290567	45	17	50
353	8	27.25	32.12	0.005238265	0.0024403469	20	7	43
380	2	36.5	31.69	0.004338	0.0025762804	30	8	46
426	12	27.67	26.02	0.0047025373	0.0040082554	21	12	39
437	168	6.95	8.97	0.05444488	0.035537668	1	1	27
448	34	19.76	18.81	0.0072006905	0.00811605	15	4	32
458	5	16.4	43.98	0.08141647	0.0012979427	1	1	48
530	26	10.12	37.15	0.0629543	0.0036351774	1	1	45
587	14	12.79	38.94	0.05677954	0.0027252273	1	1	50
623	127	7.0	7.37	0.036979515	0.034276165	2	2	21
823	25	22.52	24.13	0.005502004	0.0045518624	18	14	40
851	3	32.33	37.62	0.0045917197	0.0016382508	32	9	50
877	2	27.0	28.34	0.0028885999	0.0032403003	26	12	43
9	13	18.31	19.63	0.010193065	0.007846659	5	3	37

Bibliography

[AGYF02] Jay Ayres, Johannes Gehrke, Tomi Yiu, and Jason Flannick. Sequential pattern mining using a bitmap representation. pages 429–435. ACM Press, 2002.

[AKJ04] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *In Proceedings of the 15th European Conference on Machine Learning (ECML*, pages 39–50, 2004.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns, 1995.

[ATH03] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines, 2003.

[BDLR06] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of highly variable functions for local kernel machines. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 107–114. MIT Press, Cambridge, MA, 2006.

[Ben96] Yoshua Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1996.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, October 2007.

[CG03] Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In *PKDD*, pages 83–94, 2003.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[GRC$^+$07] John F. Gantz, David Reinsel, Christopher Chute, Wolfgang Schlichting, John Mcarthur, Stephen Minton, Irida Xheneti, Anna Toncheva, and Alex Manfrediz. Idc - the expanding digital universe: A forecast of worldwide information growth through 2010. Technical report, March 2007.

[GRS99] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 223–234, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[HPMA$^+$00] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359, New York, NY, USA, 2000. ACM.

[HPY99] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation, 1999.

[Jay57] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review Online Archive (Prola)*, 106(4):620–630, May 1957.

[JHY05] J.Pei J. Han and X. Yan. *Sequential Pattern Mining by Pattern-Growth: Principles and Extensions*, volume Volume 180/2005 of *Studies in Fuzziness and Soft Computing*, pages 183–220. Springer Berlin / Heidelberg, 2005.

[Joa10] Thorsten Joachims. Svmhmm sequence tagging with structural support vector machines, May 2010.

[Kar39] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939.

[KM02] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training, 2002.

[KT50] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on mathematical Statistics and Probability*, pages 481–492. Berkeley, U. of Calif. Press, 1950.

[KT07] Roman Klinger and Katrin Tomanek. Classical Probabilistic Models and Conditional Random Fields. Technical Report TR07-2-013, Department of Computer Science, Dortmund University of Technology, December 2007.

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[LSU07a] Srivatsan Laxman, P. Sastry, and K. Unnikrishnan. Discovering frequent generalized episodes when events persist for different durations. *IEEE Trans. on Knowl. and Data Eng.*, 19(9):1188–1201, 2007.

[LSU07b] Srivatsan Laxman, P. S. Sastry, and K. P. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 410–419, New York, NY, USA, 2007. ACM.

[LZO99] Neal Lesh, Mohammed J. Zaki, and Mitsunori Ogihara. Mining features for sequence classification. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 342–346, New York, NY, USA, 1999. ACM.

[McC02] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[MR04] Nicolas Méger and Christophe Rigotti. Constraint-based mining of episode rules and optimal window sizes. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04*, pages 313–324. Springer, 2004.

[MTV97] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences, 1997.

[Nak89] Seiichi Nakajima. *TPM Development Program: Implementing Total Productive Maintenance*. Productivity Press, New York, December 1989.

[PHW07] Jian Pei, Jiawei Han, and Wei Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *J. Intell. Inf. Syst.*, 28(2):133–160, 2007.

[SB87] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[SM06] C. Sutton and A. Mccallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[SS06] Luai Al Shalabi and Zyad Shaaban. Normalization as a preprocessing engine for data mining and the approach of preference matrix. In *DEPCOS-RELCOMEX '06: Proceedings of the International Conference on Dependability of Computer Systems*, pages 207–214, Washington, DC, USA, 2006. IEEE Computer Society.

[Tab10] Yasuo Tabei. Prefixspan: An implementation of prefixspan, May 2010.

[TL09] Vincent S. Tseng and Chao-Hui Lee. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Syst. Appl.*, 36(5):9524–9532, 2009.

[WF02] I. H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):76–77, 2002.

[Zak00] Mohammed J. Zaki. Sequence mining in categorical domains: incorporating constraints. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429, New York, NY, USA, 2000. ACM.

[Zak01] Mohammed J. Zaki. Spade: an efficient algorithm for mining frequent sequences. In *Machine Learning Journal, special issue on Unsupervised Learning*, pages 31–60, 2001.