Fakultät für Informatik
Technische Universität München

Machine Learning Methods for Detecting Rare Events in Temporal Data

Nikou Günnemann-Gholizadeh

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Uwe Baumgarten

Prüfende der Dissertation:
1. Prof. Dr. Jürgen Pfeffer
2. Prof. Dr. Eirini Ntoutsi,
Leibniz Universität Hannover

Die Dissertation wurde am 17.05.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.07.2018 angenommen.

Abstract

Time series, representing measurements of signals over time, are omnipresent in industrial and scientific applications. The recording of sensors in production processes, the evaluation of products in online system, or the dynamic behavior of users in social media are only a few examples where temporal data is collected and they show which tremendous amount of information they contain. As a consequence, to support these domains, automatic techniques that infer knowledge from temporal data are of crucial importance.

Analyzing temporal data, however, is highly challenging: often the data from real life use cases is polluted by observations which deviate from the normal behavior, potentially caused by failure events. A motivating example is the field of system diagnostics in the vehicle industry. Here, the functionality of a device can be investigated by looking for anomalous patterns in the data collected from the vehicles' heterogeneous sensors. An anomaly could hint to malfunctions and its early intervention could prevent serious consequences.

In this thesis, we address the anomaly and event detection challenge in temporal data by developing and extending intelligent data driven algorithms which learn from data. Specifically, by focusing on different temporal data domains, this thesis deals with the problem of detecting rare anomalous events, subdivided into two categories: First, we introduce methods for detecting *local* anomalies. The goal is to find time intervals whose values deviate from the remaining measured data points. Such intervals refer to rare events or anomalous behavior. We propose methods which enable an effective analysis of dynamic rating data as well as social media data over time. Second, we introduce methods for detecting *global* anomalies. Here, instead of finding intervals, whole instances, which statistically behave differently than the rest of the data, are assigned to the class of anomalies. Our developed algorithms tackle problems of early detection of rarely occurring failures in industrial sensor data. All methods are thoroughly evaluated in these different uses cases and we confirm their strong potential for rare event detection.

Zusammenfassung

Zeitreihen sind allgegenwärtig in industriellen und wissenschaftlichen Anwendungen und messen das Verhalten von Signalen über die Zeit. Beispiele umfassen das Aufzeichnen von Sensordaten in Produktionsprozessen, die fortschreitende Evaluierung von Produkten in online Systemen, oder das dynamische Verhalten von Benutzern in sozialen Netzen. Aufgrund der großen Verfügbarkeit dieser Daten und zur Unterstützung der entsprechenden Anwendungsdomänen sind automatische Methoden zur Datenanalyse unumgänglich.

Die Analyse temporaler Daten ist jedoch sehr komplex, da in Realanwendungen die Daten häufig verunreinigt sind und Fehler auftreten, die vom Normalverhalten abweichen. Als Beispiel sei das Feld der Diagnosesysteme in der Fahrzeugindustrie genannt. Um die Funktionalität einer Komponente zu testen, kann z.B. nach Anomalien in den heterogenen Sensordaten eines Fahrzeuges gesucht werden. Diese Anomalien geben Anzeichen für Fehlverhalten und deren frühzeitige Erkennung kann negative Konsequenzen vermeiden.

In dieser Arbeit adressieren wir das Problem der Anomalie- und Eventerkennung in temporalen Daten durch die Entwicklung und Erweiterung intelligenter datengetriebener Algorithmen. Unter Berücksichtigung verschiedener temporaler Datendomänen, behandelt diese Arbeit insbesondere das Problem von selten auftretenden Events, in zwei verschiedenen Kategorien: Zunächst betrachten wir die Erkennung von *lokalen* Anomalien, mit dem Ziel Teilintervalle innerhalb einer Zeitreihe zu finden die von den restlichen Messwerten abweichen. Solche Intervalle weisen häufig auf anormales Verhalten hin. Wir schlagen Methoden vor, welche eine effektive Analyse von dynamischen Bewertungsdaten und Daten aus sozialen Medien ermöglichen. Anschließend werden Verfahren zur Erkennung *globaler* Anomalien vorgestallt. Anstatt Teilintervalle zu identifizieren, werden hier gesamte Instanzen erkannt, welche von den verbleibenden signifikant abweichen. Diese werden den Anomalien zugewiesen. Die entwickelten Verfahren lösen Probleme im Bereich der frühzeitigen Erkennung von Fehlern im industriellen Umfeld basierend auf Sensordaten. Alle Verfahren sind auf einer Vielzahl von Daten und in verschiedenen Anwendungszenarien untersucht worden und es wird das große Potential zur Erkennung seltener Events dargestellt.

Publications

This thesis, "Machine Learning Methods for Detecting Rare Events in Temporal Data" includes five peer reviewed papers from the list of my publications. Table 1 shows this list including the conferences and the years at which the papers were accepted and presented. Substantially, the papers can be divided into two groups:

The three papers of the first group deal with the problem of detecting local rare events, looking for time intervals which point to anomalous behavior in the observed data. The developed approaches are related to unsupervised generative models explained in the Section 2.2.

Conversely, the two papers from the second group deal with detecting global rare events. In this context, supervised learning approaches from the area of discriminative models are used to classify instances as either anomalous or non anomalous. Section 2.3 gives an overview of the supervised discriminative models related to our published works.

Table 1: List of Selected Publications.

Published papers	Conference	Year
Part 1: Unsupervised Generative Models		
[20] Robust multivariate autoregression for anomaly detection in dynamic product ratings	WWW	2014
[25] Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution	SIGKDD	2014
[21] Finding non-redundant multi-word events on twitter	ASONAM	2015
Part 2: Supervised Discriminative Models		
[22] Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model	PAKDD	2017
[23] Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals	ECMLPKDD/LIDTA	2017

Contents

1 Introduction

Over the last decades, due to enormous improvements on big data infrastructures coupled with lower costs such as cloud computing, the potential of storing complex data from different sources has grown rapidly. Particularly, this means that there is no obstacle in storing observations for a longer period of time. This phenomenon has led to the fact that meanwhile a huge amount of temporal datasets are collected.

Analyzing such time series data over a long period creates ample opportunities and is, e.g., beneficial for discovering meaningful information or detecting specific events. In this regard, the latter is highly challenging, specifically when no prior knowledge about the event's statistical distribution is known or when and where such rare events could appear.

In this thesis, our goal is to clarify the associated challenges when analyzing data including anomalies as well as studying different solutions to detect specific behavior/events in time series. To this purpose, Section 1.1 first focuses on the different characteristics of the time series, while the challenges when analyzing anomalous temporal data are described in Section 1.2. In Section 2 we provide an overview of machine learning principles with focus on generative and discriminative models, each described with relevant details for this thesis in Sections 2.2 and 2.3. Based on this background, our contributions to new techniques are summarized in Section 2.4. These techniques are introduced as papers in Section 3. In this regard, the first three papers introduce new generative models to detect anomalies in temporal data such as in E-Commerce data or Twitter. The last two papers focus on extensions of existing discriminative models to classify industrial datasets in anomalous and non-anomalous groups.

1.1 Characteristics of Temporal Data

In general, temporal datasets possess different types of features which make their analysis a challenging task. As a result, depending on the nature of temporal data at hand, an individual model has to be used. Therefore, it is of importance to be able to characterize the observed data first. In the following, a description of the type of temporal datasets used in this thesis are explained. Likewise, Table 1.1 shows an overview of the different types of temporal datasets and their possible categorical subdivisions used in the published papers as listed in Table 1.

Definition 1.1 *A time series is a set of observations related to an instance j, denoted by $X_j = \{x^j_{t_1}, \ldots, x^j_{t_T}\}$ over a period of time $\{t_1, \ldots, t_T\}$ for the length of T, with $x^j_{t_i} \in \mathbb{D}$ representing the measurements' domain at $t_i \in \mathbb{R}$. For successive elements over time it is enforced that $t_i < t_{i+1}$.*

		Multivariate datasets	
	Datasets	*Irregular vs. Regular Time Series*	*Types of Statistical Data*
***P1**: Robust multivariate autoregression for anomaly detection in dynamic production ratings*	**E.g. Amazon**	*Irregular time series*	*Ordinal*
***P2**: Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution*	**E.g. Amazon**	*Irregular time series*	*Ordinal*
***P3**: Finding non-redundant multi–word events on twitter*	**Twitter**	*Regular time series*	*Categorical*
***P4**: Cost Matters: A new example dependent cost sensitive logistic regression model*	**Vehicle data set**	*Irregular time series*	*Numerical, Categorical and Ordinal*
***P5**: Predicting defective engines using Convolutional Neural Network (CNN) on temporal vibration signals*	**Engine data set**	*Regular time series*	*Numerical, Categorical and Ordinal*

Table 1.1: Different Types of Datasets Analyzed in the Published Papers.

In regard to this definition, we will discuss the two key characteristics of the observed datasets used in this thesis which involve a) the domain of the measured data points – which includes the aspect of dimensionality and the data's statistical type – and b) the measurements' temporal density.

The *dimensionality* of the observed data points can be roughly divided into the two categories of univariate and multivariate temporal data defined as follow:

Univariate temporal data presents one dimensional measurements at different time points. A typical example for such data would be the day temperature or product price changes over a year.

Multivariate temporal data describes multidimensional data points measured at different time points. Multivariate datasets include complex structures with latent dependencies which could be uncovered by machine learning models. Almost any dataset collected from the industrial use cases can be assigned to the class of multivariate data.

In this thesis, we mainly focus on multivariate datasets. For instance, referring to the papers [25] and [20], where the evolution of the rating data for a product or service in e.g. Amazon or Yelp is considered, the different star ratings represent the multidimensionality of the data. Another example is the measured multidimensional sensor dataset presented in the paper [22], where different features like the mileage, the speed, the engines category or the temperature for each instance (a vehicle) are considered on account of which a tire change should be predicted. Similarly, the collected cold vibration test for engines in [23] reflects the multidimensionality of each object. Interesting is also the twitter dataset in paper [21], where the multidimensionality is reflected by considering the distribution of words in hourly time intervals.

The *statistical data type* of the measured observations can be divided generally in the three classes of numerical, categorical and ordinal data.

Numerical data is also known as quantitative data which can be further broken into two categories of discrete and continues. In paper [22], we mainly focus on features with numerical values.

Categorical data known also as qualitative data, represent characteristics which can be grouped into classes/categories. The category of engines' models in the environment data used in paper [23] can be mentioned as an example for categorical data.

Ordinal data is in between numerical and categorical. This means, that the observations fall into categories which, however, allow for relative comparison between them, enabling, e.g., to create a ranking. This is the case in the e-commerce rating data considered in [25, 20].

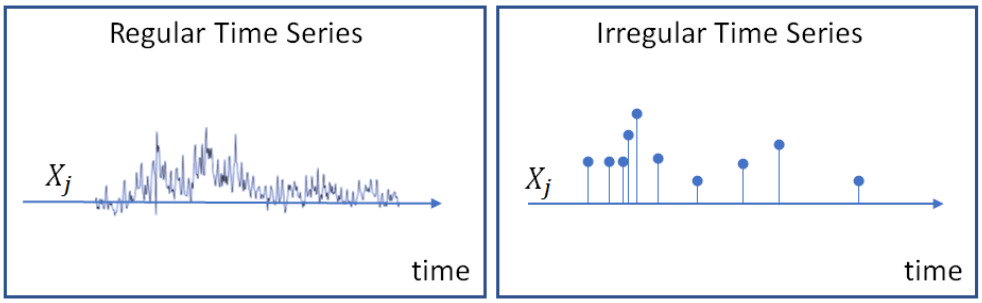

Figure 1.1: Regular Time Series vs. Irregular Time Series.

Beside the data domain, the number of considered time points and the spacing of their observation times (temporal density) have an influence on the information about the historical data. In this thesis, we consider mainly two types:

Regular time series present observations which are measured continuously over a predefined time interval. Figure 1.1 shows the clocked recording of observations along the time axis with constant continuous spacing within a time window for an instance X_j. Formally written, there exists a $c > 0$ such that for all $i \in \{1, \ldots, T-1\}$ the following term holds: $t_{i+1} - t_i = c$. The Twitter data in paper [21], or the measured cold test for engines in [23] can be assigned to this group of data, as shown in Table 1.1.

Irregular time series can be considered as a snapshot with discrete observations, where there is no cadence in time distances when recording the measurements. In this case, a static measurement gives information about a certain time point without presenting the evolution of observations continuously over time. Each collected data point itself can be considered as an event at a certain time point. However, the order of snapshots is important referring to the passage of time. Formally written, this means that $t_{i+1} - t_i$ might be different for every considered i. In Figure 1.1, lines facing upwards show the captured snapshot information. In paper [22], the measured features when requiring new tires or the rating data in [20, 25] can be classified as irregular time series.

1.2 Challenges When Analyzing Temporal Data

While collecting tremendous amounts of temporal data sounds encouraging, the analysis of such data faces substantial challenges. One core challenge is that the observed temporal data is often polluted with errors, making its analysis a difficult task. In this context, errors are defined as *anomalies* which are irregular behavior that follow a specific pattern and differ from normal behavior of the data. The appearance of errors caused by an abnormal incidence can change the natural evolution of the measured data points. Thereby, the ramification of such errors could be carried from one time period over into future time periods, influencing the entire observed time series. Detecting erroneous events over a long period of time is challenging due to the possible long term contextual changes in the data which make the distinction between the normal and the anomalous behavior very difficult.

In this thesis, it is helpful to attempt a categorization referring to the phenomena of anomalies in the analyzed data. Generally, we consider the following two categories:

Local Anomalous Behavior refers to intervals including single or multiple data points *within* a time series whose values deviate from the remaining data points of the same time series indicating possible suspicious behavior.

Global Anomalous Behavior refers to parts of a collected dataset, denoted as *instances*, whose measured temporal behavior deviates from the rest of the data, so that the *overall* instances can be categorized either as anomalous or non-anomalous.

Time points with consistently low ratings due to a change in the product's quality as in papers [20, 25] or an abnormal increase of tweets in a certain topic, reasoned by an event like in paper [21], are some examples of local anomalous behavior. The main challenge for such temporal datasets is not only to be able of detecting intervals where anomalous behavior could have been appeared but also to estimate how the evolution of measurements *without* anomalous intervals, known as base behavior, could look like (since in advance it is not known what the base behavior is). Accordingly, since the base behavior is not known, the used datasets are generally unlabeled.

In the context of global anomalous behavior, the work on malfunction engines [23] respectively the prediction of tire change [22] can be mentioned as representative examples. To be able of classifying such instances into the two classes of anomalous and non-anomalous, in this thesis, already labeled datasets are used.

Clearly, when analyzing temporal data such anomalous behavior has to be considered. Thus, one main question to deal with is:

> Challenge 1: How to design methods that are able of coping with time series data that includes anomalous behavior? In other words, how to distinguish between normal and abnormal behavior of data automatically?

Irrespective of whether local or global anomalies are considered, a general challenge affects both categories: the small amount of observed anomalous events, i.e. the events are *rare*. In many real world applications, the number of observed objects representing different classes are not equally distributed. As an example from the vehicle industry,

very often, the number of vehicles affected by a malfunction is in minority compared to the class of objects presenting functional vehicles. To stay with this binary classification example, using existing algorithms with the purpose of, e.g., predicting malfunction vehicles could lead to results with low sensitivity since most of the classical machine learning algorithms are biased towards the majority class. This is caused by loss functions which attempt to optimize the classification results based on the existing labels but often ignore the distribution of objects between the two classes. Likewise, local anomalies appear only at a few points in time, again being extremely rare. Detecting these rare occurrences is highly challenging. Overall, the second question is:

Challenge 2: How to design methods that are able of coping with *rare* events? In other words, how to increase the methods' learnability despite the poor existence of information about the anomalous instances or the polluted time points, respectively?

In addition to these two main challenges, there are of course other general aspects such as the potential high dimensionality or heterogeneity underlying the analyzed datasets shown in Table 1.1. However, the focus of this work will be mainly on contributing novel algorithms focusing on the above two challenges.

1.2.1 Example Applications

It is important to realize that in many application domains exploring multidimensional temporal data which address the mentioned challenges are essential requirements to understand the evolution of the whole process behind a business.

1) As an example, e-commerce portals (e.g., Amazon, Yelp), provide customers the opportunity to share their shopping experience by reporting and evaluating the products' quality online each time of a purchase. Evaluating such temporal data provides customers valuable information about competitive products but at the same time supports product producers to keep their service quality in view. In order to transmit such information, the challenge is to extract the base behavior of users regarding the product's quality over time as well as to discover time points at which the product's evaluation shows anomalous patterns. To this purpose, it is necessary to have available a method for analyzing rating data that incorporates the data's temporal characteristics.

2) Another example is social media (e.g., Facebook, Twitter) where huge amounts of data are shared every second. Which topics attract the sudden attention of people in social media and from which locations and for how long did these events happen? These are only a few questions of many that could be answered only based on method, being able of detecting informative events automatically from social media data.

3) Last but not least, time series data is of huge importance in the industrial domain. As an example, in the vehicle industry, the functionality of different parts of a car

could be checked by evaluating time based measurements collected from sensors which are integrated in cars. Having such data at hand, we are able to learn from the normal operation of sensors. In this context, a predominant task is to predict rare events which includes malfunctions in a vehicle to avoid negative consequences.

To tackle the described challenges in the different business use cases, the key contribution of this thesis concerns with the design of new algorithms, respectively applying and evaluating existing ones, on time based data with focus on its two predominant tasks: first, detecting rare local events/anomalies (Part I); and second, predicting rare global events which are categorized as anomalous instances (Part II).

Figure 1.2 summarizes the list of data analyzed in our published papers together with their temporal properties and the type of covered challenges. In the last column, relevant questions to be answered in relation to the different business use cases and techniques are listed.

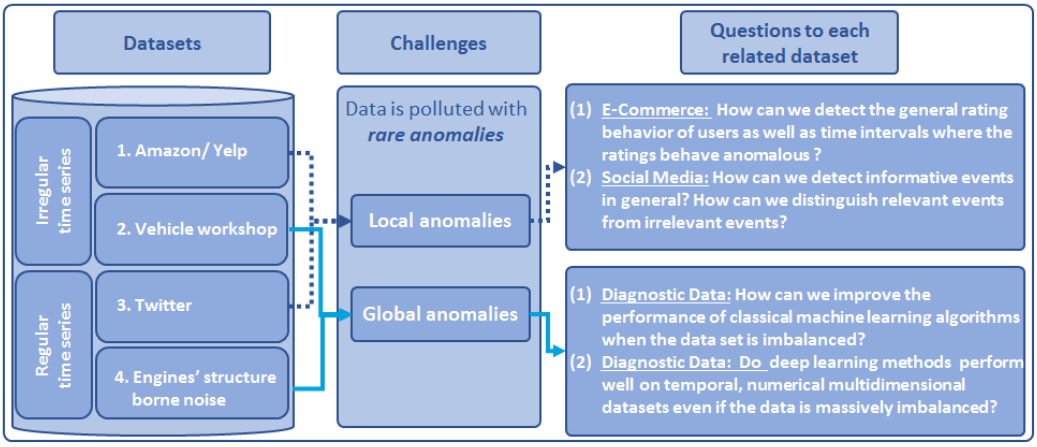

Figure 1.2: An Overview of Leading Questions Related to the Considered Data.

2 Existing Conceptual Consensus

Most of the existing conventional algorithms assume perfect data. Nevertheless, as aforementioned, real world datasets are affected by characteristics as, e.g., anomalies and rare events. To recognize and extract such informative patterns from the data, in this chapter, an overview of the Knowledge Discovery in Databases process [24, 26, 42] as well as the principles of machine learning relevant for this thesis are given.

2.1 Data Analytics and Machine Learning Foundations

2.1.1 KDD Process

Knowledge Discovery in Databases (KDD) [42] is an interdisciplinary research process consisting of the following steps: 1) First, the raw data is collected from one or multiple data sources. Since raw data is noisy and incomplete, data cleaning is expected. This includes the correction of missing values, removal of noises or inconsistencies. 2) The cleansed data from multiple sources are then integrated into a data warehouse which could subsequently be used for analytical uses. In this step, it is also possible to create new data dimensions from the existing data for the later analysis steps. 3) To this purpose, depending on the imminent analytical tasks, relevant data dimensions are first selected. To meet the necessary requirements by the following algorithms, the selected dimensions might need to be transformed. This step is also called data preprocessing. 4) After the preprocessing step, the actual knowledge discovery follows by applying machine learning methods and statistical models. 5) Finally, the achieved results are evaluated. For the purpose of understanding, the results are visualized. If the achieved results are not satisfactory, an improvement of results by executing steps four and five iteratively as well as taking a glance on the selected features in the preprocessing step can be helpful. Figure 2.1 illustrates the explained KDD process in 5 steps.

Without loss of generality, we assume for each use case in this thesis that the preprocessing step has been executed. While this includes basic cleaning, normalization, transformation or feature extractions, its effect on recognizing and extracting anomalies and errors is restricted. Specially, if the involved events in the data are rare, their recognition gets more difficult by human, without any supportive tool.

Thus, in this thesis, our goal is to design new algorithms in the step of machine learning and statistics to recognize (anomalous) patterns in level 4 and evaluating their correctness of extracted knowledge in level 5 of the KDD process.

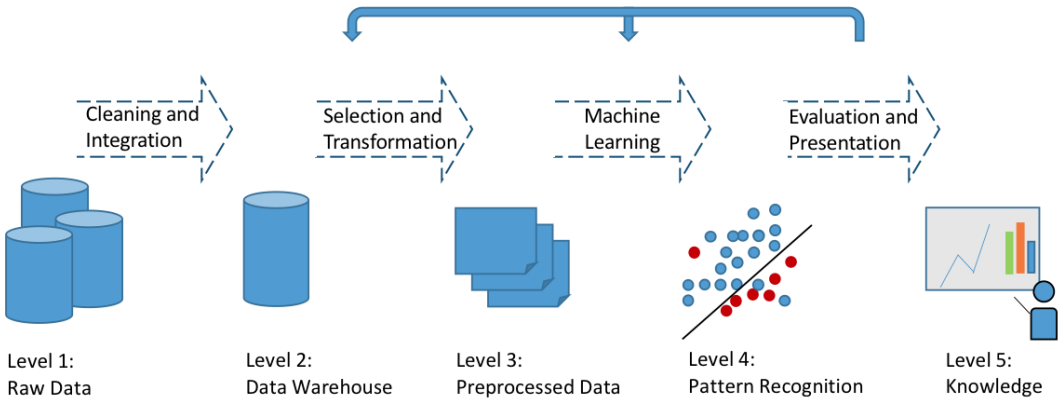

Figure 2.1: Knowledge Discovery in Databases (KDD).

2.1.2 Machine Learning Principles

Generally, machine learning algorithms can be distinguished in the following three categories [6, 18, 51]:

- *Supervised learning* covers those algorithms that learn based on labelled data. Each instance j can be expressed as a pair of X_j as input and y_j as target. The algorithm learns automatically a function f which maps the inputs to their desired targets by $f(X_j) = y_j$. If the target values belong to a set of categories, then the class of considered problem is called classification problem. Alternatively, in the case when the targets have continues values we have a regression problem.

- *Unsupervised learning* refers to those algorithms which learn based on unlabelled inputs X_j. The goal is to automatically learn and recognize patterns and structures hidden in X_j. One of the most important unsupervised learning tasks is clustering through which similar instances are grouped in the same cluster.

- *Semi-Supervised learning* falls between supervised and unsupervised learning. The algorithm learns based on a set of labelled and unlabelled instances. Depending on the task, methods from unsupervised and supervised learning are adapted to be used in semi-supervised learning. In doing so, semi-supervised learning does not only learn from labelled instances and their correlation to their desired target but hopefully also by considering the hidden correlation between labelled and unlabelled instances, if such correlation exists.

What is the best model? So far, the different types of considered data as well as the different types of learning concepts, relevant for this thesis, are described. However, how to assess the performance of the finalized model to be used?

The main reason for poor performance of a model is subjected to the phenomena of either overfitting or underfitting [6, 51]. If the estimated model performs well only

on observed data and degrades on new observations, then we talk about an overfitted model. Alternatively, if the estimated model can neither model the observed data nor it is bale to be generalized to the new observations, then the model is underfitted.

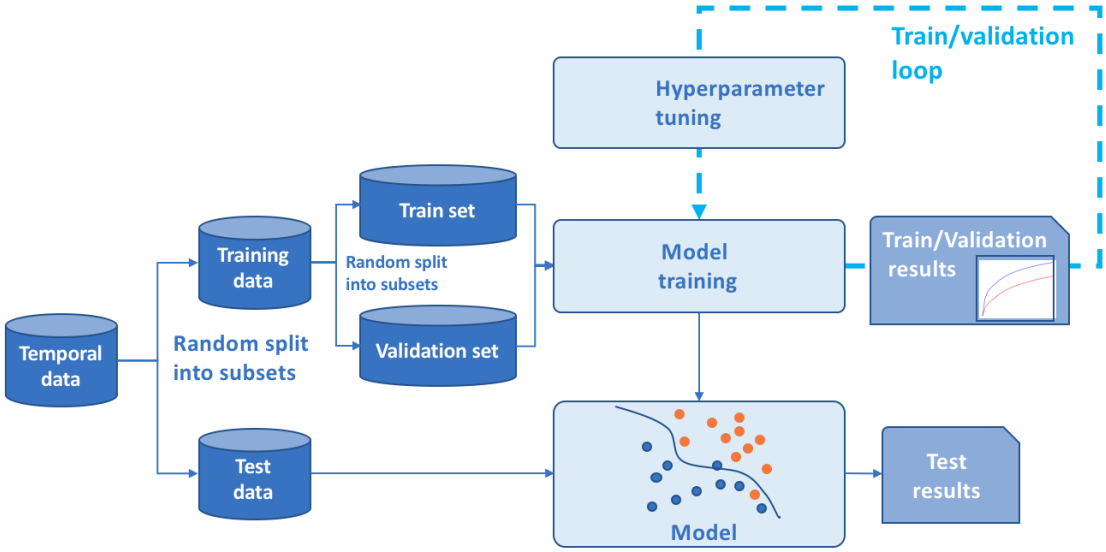

Figure 2.2: Evaluating Supervised Machine Learning Models.

Using model selection techniques supports the attempt to search for a model that achieves good fit on the training data but also performs well on new observed data. In the following, we give an overview of some standard approaches used in the papers [20, 22, 25] to select the best model.

Model selection in supervised learning: Regardless of the considered tasks and which supervised learning method is used, the reliability of the reached results can be evaluated based on the following steps.

1. First, the collected temporal data is divided into two subsets of randomly selected training and test instances. To avoid the problem of underfitting, the number of instances in the training data is much higher than the test data.

2. In the *learning phase*, the training data is in turn divided randomly into the two sets of train and validation data. First a model is trained on the train set that aims to fit the data best.

3. In the *validation phase*, we validate the reliability of the fitted model with its learned weights on the remaining validation set. In this phase, we use the data to tune the parameters to find a model that generalizes.

4. Finally, in order to confirm the performance of the fitted model, we evaluate the accuracy of the model based on the remaining test data which can be interpreted as unseen or new data. This phase is called *test phase*.

		Actual Value (confirmed by the experiments)		
		negatives	positives	
Predicted Value (predicted by the test)	negatives	True Negatives (TN)	False Negatives (FN)	Negative Predicted Rate = $\frac{\sum TN}{\sum Total\ Negative}$
	positives	False Positives (FP)	True Positives (TP)	Positive Predicted Rate **(Precision)** = $\frac{\sum TP}{\sum Total\ Positive}$
		True Negative Rate **(Specifity)** = $\frac{\sum TN}{\sum All\ Negative}$	True Positive Rate **(Sensitivity= Recall)** = $\frac{\sum TP}{\sum All\ Positive}$	**Accuracy** = $\frac{\sum TN + \sum TP}{\sum All\ Data}$

Table 2.1: Basic Measures Derived from the Confusion Matrix.

The introduced method to evaluate the reliability of the learned model is known as the *holdout* method. Figure 2.2 shows the partitioning of the raw data in the three mentioned subpartitions as well as training the model in the process of machine learning. Depending on the machine learning model hyperparameters can be set before the training phase which could influence the performance of the results.

In the papers [22, 23], the holdout method is used to train and select a model. Since the two datasets are time based, it if of importance to do not ignore the order of instances in time, meaning that the indices of timepoints in test dataset have to be higher than those in training data as well as the order should also be kept.

Alternatively, *cross validation* can be used, where the hold out method is used based on the three randomly selected partitions, repeatedly.

In order to select the best model (in the validation phase) and evaluate the performance of a classification model in general, a so called confusion matrix can be computed, allowing to derive many useful statistical measures. This is shown in Table 2.1. If the data is imbalanced, the two measures precision and recall reflect the performance of a classification model w.r.t. the prediction of rare events best.

Model selection in unsupervised learning: Evaluating the performance of unsupervised methods is not a trivial task, since the input data is unlabelled. Thus, depending on the learning method, the evaluation method can be different.

For the case of clustering e.g. methods such as Bayesian approaches, Minimum Description Length (MDI) or Cross Validation (CV) techniques [19, 41, 57] can be used to estimate the number of clusters. However, the authors in [17] discussed the imperfection of such methods.

For evaluating unsupervised generative models, the most common methods besides CV are:

- *Akaika information criterion (AIC):* It returns the model with the relative minimum information loss among all candidate models [4, 56]. To this purpose, let's first assume, we have a candidate model M_i and a true Model M^* each with a log-likelihood functions $l(\tilde{a}; X)$ and $l(a^*; X)$ where X denotes the data at hand and $\tilde{a}$ respectively a^*, a parameter vector to be learnt. To estimate the dissimilarity between the two models M_i and M^*, the well-known Kullback-Leibler divergence (KL) [11, 39], also known as relative entropy can be used. However, since the any parameter vector a^* is unknown, we approximate the minimized KL divergence by estimating the maximized $l(a; X)$. This implies:

$$AIC = -2l(\tilde{a}) + 2k \ , \tag{2.1}$$

  where $l(\tilde{a})$ is the maximum log-likelihood of the model and k denotes the number of parameters in vector $\tilde{a}$ to be estimated.

- *Bayesian information criterion (BIC):* An alternative to AIC is given by the BIC [3]. It estimates the performance of a model on a testing dataset by:

$$BIC = -2 \cdot l(\tilde{a}) + k + log(N) \ , \tag{2.2}$$

  where $l(\tilde{a})$ is the maximum log-likelihood of the model, k denotes the number of parameters in vector $\tilde{a}$ to be estimated and N represents the sample size of the training dataset. In [20, 25] we used the BIC model selection principle to select the optimal value for the number of anomalies in rating data.

2.1.3 Generative vs. Discriminative Models

Considering the two main tasks introduced in Chapter 1 – the attempt of detecting anomalies in temporal data respectively classifying anomalous and non-anomalous events – the two principles of supervised and unsupervised learning as described above can be used. In the case of supervised learning, given is $X_j = \{x^j_{t_1}, \ldots, x^j_{t_T}\}$ as a time series input (e.g. signals from a sensor) and a class label y_j (e.g. defect vs. not defect) for each instance $j \in N$. To classify the instances, meaning to learn a function $f : X_j \mapsto y_j$, generally, methods from the two categories of *generative* and *discriminative* models can be used [54].

Supervised generative models model the class which most likely generated the observed measurements by learning the joint probability distribution $p(X_j, y_j)$ between the given X_j and y_j. By using Bayes rule, the probability $p(y_j|X_j)$ can then be derived in order to assign each instance its most likely labelled value.

Supervised discriminative models learn directly the conditional probability distribution $p(y_j|X_j)$ for each j, i.e. they learn the boundaries between the existing classes by using the posterior distribution $p(y_j|X_j)$ directly.

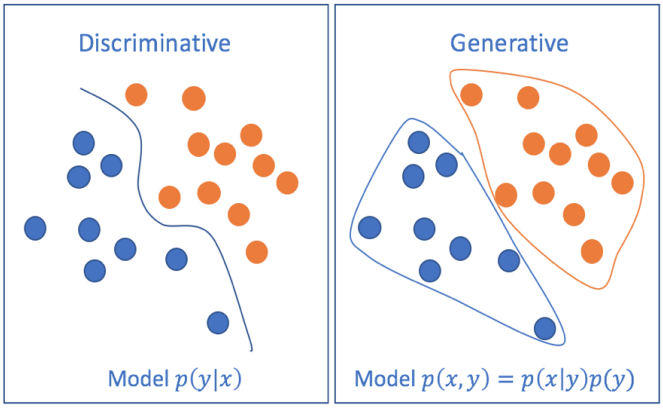

Figure 2.3: Discriminative vs. Generative Learning in Supervised Learning.

Figure 2.3 outlines the difference between the two models by presenting the discriminative model as a conditional probability $p(y_j|X_j)$ and generative model as a specification of the joint probability $p(X_j, y_j)$ for supervised learning. In this thesis, the developed methods in the papers [22, 23] are related to the supervised discriminative algorithms.

In contrast, in case of unsupervised learning where the given data is unlabeled, mainly the principle of generative models is used. The developed methods in the papers [20, 21, 25] are related to this principle.

Unsupervised generative models attempt to learn the probability distribution $p(X_j)$ which approximates the natural distribution of given data X_j at best.

The following two sections, Section 2.2 and 2.3, introduce the state of the art to unsupervised generative models as well as supervised discriminative models applied on temporal data which are relevant for the contribution of this thesis.

2.2 Modelling of Time Series: Generative Process

Detecting specific patterns in biological processes [16, 50, 60, 62], evaluating anomalous behaviour in product ratings [20, 25, 32], detecting topical trends [7, 15, 21, 63], or predicting human activities using body-worn sensors [46] are just a few examples of applying unsupervised generative models for temporal data in praxis.

Thereby, as mentioned before, the general goal is to find a model that describes the observed data best. To this purpose, in many time based applications, specially in signal processing, one of the most important and commonly used procedures is the *Autoregressive Moving Average* (ARMA) model [6, 9]. It assumes, that a future value of a variable of interest is linear dependent on its several past observations.

The ARMA model consists of mainly two submodels: the *Auto Regressive* (AR) model [2, 9, 61] and *Moving Average* (MA) model [6, 9, 62]. The submodels can be used as generative models which forecast the future data point by using the weighted sum of previous time points or sum of noises, respectively. In this regard, the assumed values for the weights are learnt based on historical data. The learnt weight parameters express

the influence of each data point from the past on the future forecasted data point. In the context of anomaly detection in temporal data, a clear deviation between the forecasted data and its corresponding observed data could hint to a suspicious event in the data. This conclusion, however, can only be drawn if the data where we learnt the model's parameters on itself was not corrupted – which in reality, however, is rarely given. Thus, the assumption of stationarity (see below) underlying the ARMA model is rarely true. Solving this is precisely the idea that have been used in [20, 25].

In the following, the two generative models AR and MA as well as the ARMA model are described in detail. Additionally, to generate non-stationary temporal data an extension of ARMA to ARIMA model is explained.

2.2.1 Stationary Models

A stationary model assumes that the statistical properties of the underlying time series $X_j = \{x^j_{t_1}, \ldots, x^j_{t_T}\}$ behave constant and particularly varies by a small amount around the constant mean for a long period of time [29, 53, 64]. Nevertheless, in reality most time based data behaves non-stationary [53], meaning that they are, e.g., polluted already with anomalies. The concept of analyzing stationary time series data based on a finite parameter model was first introduced by Yule [65]. The stochastic models introduced in the following, base on his work. For this purpose we first introduce the "backshift operator", a general notation used in time series analysis.

Notation 2.1 *Given is a time series $X_j = \{x^j_{t_1}, \ldots, x^j_{t_T}\}$, then $\boldsymbol{B}$ is a "backward shift operator" also known as lag operator on $x^j_{t_i}$ defined as: $\boldsymbol{B} \cdot x^j_{t_i} = x^j_{t_{i-1}}$, for all $i > 1$, meaning back up from x at time t_i by one time unit. Hence, by considering k applications of $\boldsymbol{B}$ to $x^j_{t_i}$, $\boldsymbol{B}^k \cdot x^j_{t_i} = x^j_{t_{i-k}}$.*

The backward shift operation can be used to construct the backward difference operator Δ, defined as $\Delta x^j_{t_i} = x^j_{t_i} - x^j_{t_{i-1}}$. In terms of $\mathbf{B}$,

$$\Delta x^j_{t_i} = x^j_{t_i} - x^j_{t_{i-1}} = x^j_{t_i} - \mathbf{B} \cdot x^j_{t_i} = (1 - \mathbf{B}) \cdot x^j_{t_i} \tag{2.3}$$

Linear Filter: Based on Yule's theorem [65], a time series signal with interdependent sequence of time points showing periodicities is subject to linearity. In this context, a linear filter can be used to generate such signal effectively by combining a series of randomly distributed superposed fluctuations ϵ_t also known as white noises with zero mean and constant variance σ^2_ϵ. Formally, such filter is described as:

$$x^j_{t_i} = \mu^j + \epsilon^j_{t_i} + \sum_{m=1}^{i-1} \alpha^j_m \cdot \epsilon^j_{t_{i-m}} = \mu^j + \sum_{m=1}^{i-1} \alpha^j_m \cdot \mathbf{B}^m \cdot \epsilon^j_{t_i} \tag{2.4}$$

where μ^j denotes a constant mean about which the signal varies and α^j_m's are some parameters to be learnt in the model. In other words, Equation 2.4 implies that the value of the current time point depends on a current white noise linearly combined with *all* its weighted noises in the past. Attention should be paid to the unbounded number of prior values; meaning the number of time points considered back in time is not fixed.

Moving Average Model: An alternative to the linear filter is the so called Moving Average (MA), which considers only a *finite* number of prior values to forecast $x^j_{t_i}$ using a regression model on its past noises [9]. This means that in an MA model, we consider mainly the resulting error occurred between present and some previous terms:

$$x^j_{t_i} = \mu^j + \epsilon^j_{t_i} + \sum_{q=1}^{Q} \Theta^j_q \cdot \epsilon^j_{t_{i-q}} = \mu^j + (1 + \sum_{q=1}^{Q} \Theta^j_q \mathbf{B}^q)\epsilon^j_{t_i} \tag{2.5}$$

where Θ_q denotes the parameters of the model with the order q, and ϵ represents independent random noise terms for the j-th instance. Here, the order is the finite number of preceding terms in the considered series that are used to predict x^j at time t_i. We assume, that the series of randomly extracted noises are normal distributed with a zero mean and a constant variance σ^2_ϵ. The sequence of uncorrelated variables $\{\epsilon_{t_1}, \epsilon_{t_2}, \ldots, \epsilon_{t_T}\}$ is also called a white noise process or a Gaussian white noise.

Auto Regressive Model: While MA forecasts the value of $x^j_{t_i}$ in a linear combination of its past noises within a predefined time window, the Autoregressive model (AM) considers the *value* of its prior time points. Formally, given is the time series $X_j = \{x^j_{t_1}, \ldots, x^j_{t_T}\}$, the approach of forecasting a value at a future time point e.g. t_i is defined as a linear combination of its p past observations:

$$x^j_{t_i} = v^j + \alpha^j_1 \cdot x^j_{t_{1-1}} + \ldots + \alpha^j_p \cdot x^j_{t_{i-p}} \tag{2.6}$$

Looking at 2.6, $x^j_{t_i}$ is obviously regressed on $\{x^j_{t_{i-1}}, \ldots, x^j_{t_{i-p}}\}$ to which the name "autoregressive model" refers to [9]. Since the estimated $x^j_{t_i}$ is only an approximation of its real value, a white noise denoted by ϵ has to be added such that $x^j_{t_i}$ is formally defined as:

$$x^j_{t_i} = v^j + \sum_{p=1}^{P} \alpha^j_p \cdot x^j_{t_{i-p}} + \epsilon^j_{t_i} \tag{2.7}$$

where α_p are model parameters at order p and v^j is a constant referring to the underlying trend in x^j. Using the backshift notation at the AR equation in 2.7:

$$x^j_{t_i} = v^j + \sum_{p=1}^{P} \alpha^j_p \cdot \mathbf{B}^p \cdot x^j_{t_i} + \epsilon^j_{t_i} \tag{2.8}$$

By moving the equation under the sigma sign to the other side of the equation, it gets clear that AR is a special form of Linear Filter whose input is a white noise.

$$(1 - \sum_{p=1}^{P} \alpha^j_p \cdot \mathbf{B}^p) \cdot x^j_{t_i} = v^j + \epsilon^j_{t_i} \tag{2.9}$$

In [20], to detect the potentially evolving base behavior of users, defined as an unknown parameter, an AR based model is used to generate the observed rating data as well as to spot anomalous points in time where the rating distribution differs from this base behavior.

Mixing Autoregressive and Moving Average Models: Often, parts of the observed data exhibits homogeneous behaviour while other parts show trends. A linear combination of AR Eq. 2.7 and MA Eq. 2.5 enables flexibility by modelling both behaviours. The resulting model is called *Auto Regressive Moving Average* (ARMA) model [9]:

$$x^j_{t_i} = v^j + \sum_{p=1}^{P} \alpha^j_p \cdot x^j_{t_{i-p}} + \epsilon^j_{t_i} + \sum_{q=1}^{Q} \Theta^j_q \epsilon^j_{t_{i-q}} \tag{2.10}$$

$$\Leftrightarrow (1 - \sum_{p=1}^{P} \alpha^j_p \cdot \mathbf{B}^p) x^j_{t_i} = v^j + (1 + \sum_{q=1}^{Q} \Theta^j_q \mathbf{B}^q) \epsilon^j_{t_i} \tag{2.11}$$

including $P + Q + 2$ unknown variables $v, \sigma_\epsilon, \alpha_1, \ldots, \alpha_p, \Theta_1, \ldots, \Theta_q$. ARMA belongs also to the important class of *stationary* stochastic models, assuming that the statistical properties of the data do not change over time. Though, as discussed in Section 2.2.1, many temporal datasets are not necessarily stationary in time.

2.2.2 Non-Stationary Models

Several studies in image recognition, speech recognition, language processing and many other areas have shown that time series data behave with no fixed mean and variance [5, 12, 35, 37, 38]. In particular, data collected in industry, e.g. the sensor data in a vehicle, shows that the collected time series exhibit non stationary behaviour. However, even if the data in general is non stationary covering different fluctuations, there may be still intervals within the time series that exhibit similar behaviours. To obtain such homogeneous non stationary statistical properties, in addition to AR, MA, and ARMA, the differences between consecutive observations denoted by d can be considered. This is known as *differencing*. In practice, d is usually equal to 0, 1 and 2 whereas $d = 0$ corresponds to stationary behaviour between the consecutive observations. The resulting model is called ARIMA(p, d, q) [9] where p and q represents each the order of the autoregressive respectively the moving average part. For simplicity, we use the backward shift notation explained in Section 2.1, which results in:

$$\overbrace{(1 - \sum_{p=1}^{P} \alpha^j_p B^p)}^{AR(P)} \cdot \overbrace{(1 - B)^d \cdot x^j_{t_i}}^{d-differences} = v^j + \overbrace{\left(1 + \sum_{q=1}^{Q} \Theta^j_q B^q\right) \cdot \epsilon^j_{t_i}}^{MA(Q)} \tag{2.12}$$

The temporal data considered in this thesis are likewise non stationary. However, the main reason there is the inclusion of abrupt changes caused by anomalies. Thus, alternative models are required. For this, in the papers [20, 21, 25], we present our new designed generative models mostly based on latent decomposition to estimate the base behavior as well as local anomaly behaviors within the data at the same time. The related experimental results show a substantial improvement of anomaly detection in temporal data in comparison to the existing methods.

2.3 Classification of Time Series: Discriminative Process

Discriminative processes are preferred over generative ones in many supervised learning tasks due to their theoretical properties. In [8, 54], the authors showed that a generative approach reaches its higher asymptotic error much faster than a discriminative approach. It is also shown that by increasing the number of data points, discriminative approaches perform better than the generative one.

Consider the classification tasks for predicting (global) anomalous and non-anomalous events: A discriminative approach learns a boundary which directly maps the observed data to their corresponding label by modelling the posterior distribution $p(y_j|X_j)$ [6, 18]. In other words, in discriminative approaches, the overall goal is to fit a model on the observed data X through which we predict y denoted by $\tilde{y}$ best.

The overall quality of the fitted model can be measured by minimizing the empirical risk which estimates the misfit between y and $\tilde{y}$.

$$E = \frac{1}{N} \sum_{j=1}^{N} l(y_j, \tilde{y}_j) \tag{2.13}$$

For computational ease the usage of a *loss function*, operating on the true class y_j and its predicted value $\tilde{y}_j$ is standard. More details to this are given in Section 2.3.1. Some examples for discriminative models are Support Vector Machines (SVM) [14, 59], Regression models [33], Random Forests [10, 47] or the class of Neural Networks [6, 18, 27]. In the following, some of these techniques which are used in our published papers [22, 23] are briefly introduced as a state of the art.

Notation 2.2 *Given is a set of observed data represented as tuples* $D = \{(X_j, y_j)\}_{j=1}^{N}$, *where each instance* $X_j = \{x_{t_1}^j, \ldots, x_{t_T}^j\}$ *reflects a T-dimensional vector which is assigned to a class label* $y_j \in \{0, 1\}$. *Parts of this thesis focus on detecting global anomalies by using binary classification [22, 23]. Here,* $y_j = 1$, *presents the class of instances manifesting anomalies while* $y_j = 0$ *is the opposite.*

2.3.1 Logistic Regression

Logistic Regression (LR) is one of the most frequently used methods in supervised learning for the purpose of prediction. Its popularity is due to its simplicity and interpretability of the model's parameters [55]. To predict the class labels of instances, the logistic regression assigns each instance a probability. The probability of an anomalous instance ($y_j = 1$) is denoted by $P(y_j = 1|X_j)$, and $P(y_j = 0|X_j) = 1 - P(y_j = 1|X_j)$, for $y_j = 0$. The estimated probability is learned based on the sigmoid basis function:

$$P(y_j = 1|X_j) = f(g(X_j, \beta)) = \frac{1}{1 + e^{-g(X_j,\beta)}} \ , \tag{2.14}$$

where $0 \leq f(g(X_j, \beta)) \leq 1$ and $g(X_j, \beta) = \beta_0 + \sum_{i=1}^{T} \beta_i \cdot x_{t_i}^j$ is a linear expression including the explanatory features and the regression coefficients β. In statistic terminology,

the Eq. 2.14 is known as *linear logistic regression*. Considering the sigmoid equation, the main challenge is to estimate β in $g(X_j, \beta)$ to make $f(g(X_j, \beta)) = \tilde{y_j}$ close to y_j, as mentioned in Eq. 2.13. Assuming that the N samples in the observed data are independent, we can write $P(y_j|X_j; \beta)$ as a product, leading to the following overall *Likelihood function*:

$$L(Y, X, \beta) = \prod_{j=1}^{N} \tilde{y}^{y_j} \cdot (1 - \tilde{y})^{1-y_j} \tag{2.15}$$

The β can be obtained by minimizing the negative log likelihood of Eq. 2.15 given by:

$$E = l(Y, X, \beta) = -\sum_{j=1}^{N} y_j \cdot \log \tilde{y} + (1 - y_j) \cdot \log(1 - \tilde{y}) \tag{2.16}$$

The result is the so called *logistic loss function* or *cross entropy error* which is used to measure the fit of the model. A standard technique to find the optimal values for β is to minimize the negative log likelihood via *gradient descent* [6, 18]. Gradient descent is an iterative process, where in each step one moves into the direction where the objective function decreases most. Gradient descent consists of two hyperparameters: the *learning rate*, which determines the size of steps, and the *iteration number* [18].

In the paper [22], a logistic regression model is used to predict a necessarily needed tire change for observed vehicle data. However, the classical LR does not meet the real world use cases since it intuitively tries to minimize the so called zero-one loss with the assumption that all misclassified instances have equal cost. This means the correct classifications lead to a cost of zero and misclassification gets a cost of one. Additionally, instances with an issue are in minority which reflects the challenge of rarely occurring anomalies as introduced in Section 1. To this purpose, we upweighted the occurrence of rare events by adapting the logistic loss function of LR to a new cost-sensitive-one, introduced in the paper [22].

2.3.2 Neural Networks

Logistic regression is a very useful and easily interpretable model, however it can quickly reach its limits by the curse of dimensionality [6]. Besides, nonlinearity within the data significantly limits the performance of such models since LR learns linear decision boundaries. This could be caused by, e.g., anomalous behavior within the data which causes non stationary properties. The set of engines' data we analyzed in paper [23] pertain to such scenarios. In order to solve the classification problem in such datasets, it is necessary to adapt the basis function on nature of nonlinear observed data. This can be warranted by constructing multiple layers of basis functions one upon the other in form of a network.

Neural networks are related to LR insofar, that if the network consists of only one layer with one basis function, namely the sigmoid function (Eq. 2.14), then the network reflects the form of a LR classifier. This is shown in Figure 2.4. In the following, we will first give an overview of a general neural network architecture.

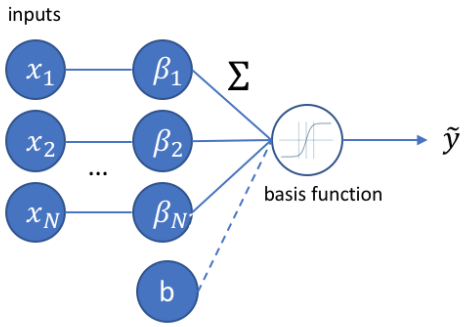

Figure 2.4: Illustration of Classical Logistic Regression in a 1-Layer Network.

Referring to Notation 2.2, to map a given input X_j to its corresponding label y_j, first a mapping function $y_j = f(X_j; \beta)$ has to be learned. In neural networks, this process is known as *feedforward*, since information flows from the input data X, through the intermediate computations used to learn f, forwarded to the final step where the output $\tilde{y_j}$ is estimated [6, 18, 34]. The intermediate computations are designated to be the core of neural networks, since different functions for different purposes are composed together like a connected chain building different layers [18]. These intermediate computations build one layer of the overall three layers in a neural networks for the purpose of classification. These three layers are introduced in the following in detail with an overview of their architecture, shown in Figure 2.5:

The input nodes which represent the first layer, forward the observed data X by duplicating them to next layer.

The intermediate nodes, include different functions which are connected to each other as a chain structure, applied on the input data. Technically, this means that e.g. f^1, f^2 and f^3 are composed together in the following sequence: $h_1 = f^1(X; \beta_1), h_2 = f^2(h_1; \beta_2)$ and $h_3 = f^3(h_2; \beta_3)$, where X, h_i represent vectors, β_i represent weights to be learnt and f_is represent some non-linear functions which transform the input data into a new scale which in turn can be used as the input for the next non-linear function at the next layer. The used functions to describe the non-linearity in the data are called *activation functions*.

An example for nonlinear functions is the logistic sigmoid function $f(g(X, \beta))$ presented in Section 2.3.1. In contrast to LR as a convex problem, which guaranties a distinct minimum global optimum as a solution, in a neural network, by using multiple layers of such basis functions, the convexity gets lost. Consequently, to minimize the error for the whole network, meaning to find an optimal value for β, the local optima is estimated by computing recursively the gradient of Eq. 2.16 with respect to each activation node.

Since during the feedforward process, the desired labels y to each input data X are not shown, the intermediate nodes are also known as *hidden layers*. The learning algorithm decides by minimizing the loss, how to use each layer to get the desired label [18].

The output nodes represent the predicted y denoted by $\tilde{y}$. The last layer can be written as $\tilde{y} = f_{last}(h_{last-1}; \beta_{last})$

The solid lines between the nodes refer to the flow of information from one layer and from each node to every node on the next layer in the feedforward network.

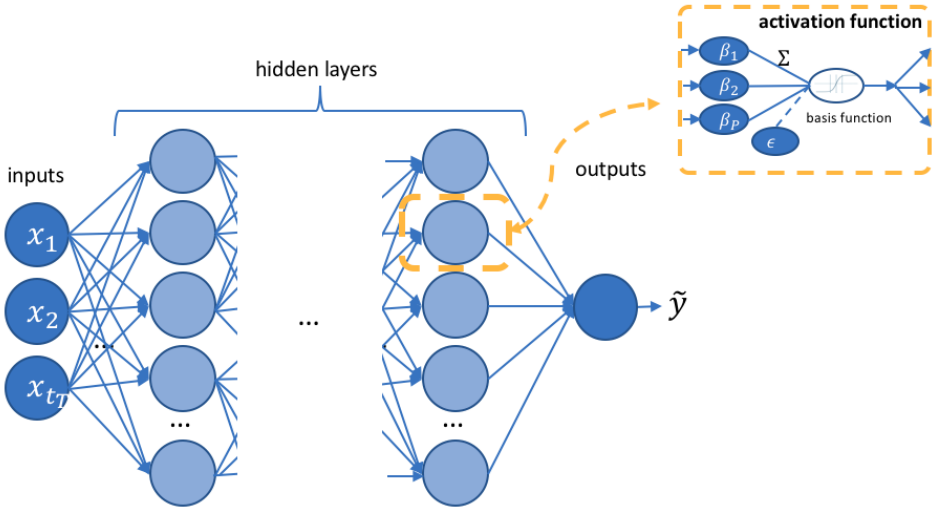

Figure 2.5: Illustration of an N-Layer Neural Network.

Besides the sigmoid function, there are several other possible activation functions which can be encountered in a neural network with different effects. Figure 2.6 illustrates the characteristic of commonly used activation functions in practise. A very frequent side effect of most used activation functions is the so called *saturation*, meaning the error of the neural network stays at a high constant for some period of time during the learning [28, 45, 58]. In case of e.g. the sigmoid function, this would mean that the gradient is almost zero at these regions. A saturation is known to be caused by an inappropriate set of initial weights [45]. Thus, depending on the input data, it is important to select the appropriate activation function.

2.3.3 Specialized Architectures

In classical neural networks, depending on the size of the input units and the number of hidden layers, the number of weights to be learned in the intermediate nodes can increase rapidly, causing high complexity. This applies in particular to temporal data having often many time points. Likewise, the caused significant non-linearity within the temporal data limits the performance of classical machine learning models as LR, introduced in Section 2.3.1 which learns linear decision boundaries. Also, considering one of the most widely used models for time series data, the ARIMA model in Section 2.2.2, shows that the performance is again limited due to its lack of flexibility by extracting non-linear correlation structures from the data.

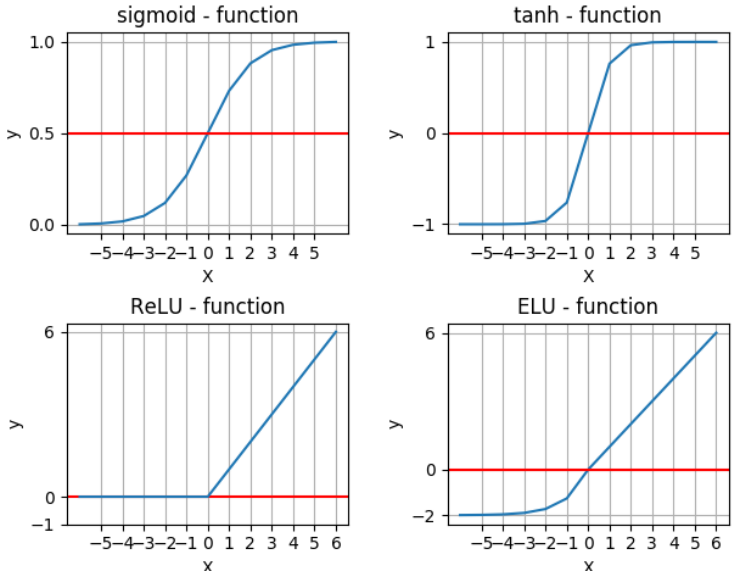

Figure 2.6: Common Activation Functions Used in Neural Networks.

Thus, as an alternative to these methods, the usage of so called Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) are suggested. In the following, the structure of such neural networks is explained. Since the focus of this thesis is on CNN and not RNN, first only a broad overview of matters of general interest is given. Afterwards a detailed explanation to the CNN is presented.

Recurrent Neural Networks Recurrent Neural Networks operate on sequential data. They include the three different layers mentioned in Section 2.3.2. However, the main advantage of RNN gets visible by sharing parameters across different parts of the model in the hidden layers, learning long term dependencies. This is done by applying a set of computations recursively mapping an input and the parameters on an output [18].

$$h_{t_i} = f(h_{(t_{i-1})}, x_{t_i}; \beta) \tag{2.17}$$

Hereby, as shown in Equation 2.17, the applied functions f as well as the learned parameters β are always the same. The weights represent how the importance of the hidden states according to the present as well as past is.

Although the main purpose of RNN is to capture long-term dependencies, recent researches show that it is difficult to store these information for a very long period of time [44]. On the other hand, instead of considering the overall sequence, it could be of advantage to focus on local variations and behaviors within the time series. Such intervals could be a potential indicator for anomalous events. To extract such salient features within a certain time interval, the usage of convolutional neural networks is recommended [18, 44].

Long Short Term Memory Typically, as mentioned before, RNNs are not suited to solve problems with learning long-term temporal dependencies because of the vanishing gradient problem [30]. An advanced version of RNNs are Long Short Term Memories (LSTMs) which use special blocks (or units) in addition to standard blocks. Each block forms a memory cell which consists of three components: a) an input gate, b) an output gate and c) a forget gate. This set of gates are used to control for how long the incoming information should be maintained in memory [31]. However, one of its main disadvantages is its slow training.

Convolutional Neural Network Convolutional networks are a specialized kind of neural network often used for data with known grid topology [18]. While originally proposed for image-like data where the spatial locality of neighboring pixels is exploited, it has recently been proven very successfully for temporal data as well. Examples include temporal data with equally spaced time series as in engine data [23] or in speech recognition [1, 43]. The essential feature of CNN is the inclusion of built invariance properties into the structure of CNN [43, 49].

Considering the specific task of predicting malfunctioned engines in our published paper [23], the resulting data is a set of matrices $S_i \in R^{N \times L_i}$, where each $i \in M$ represents one signal measured for all instances. Here, N denotes the number of instances/engines and L_i the length of frequency spectra regarding signal i. Accordingly, the $j-th$ row of S_i comprises the frequency spectrum collected for engine $j \in N$ regarding signal i. Relating to Notation 2.2, $S_{j,i} = X_j$, whereas the length of the considered time points is $L_i = T$.

A key property of such data, similar to images, is the assumption that nearby frequency measures are more strongly correlated than distant ones. Extracting such local information and processing them in later steps are incorporated into CNN through three mechanisms [6]: (i) local respective fields, (ii) weight sharing and (iii) subsampling. The idea of local respective fields were first inspired by [36]. The "discovered" local information are the results of convolutions between the input signal row X_j and weights $\beta = (\beta_1, \ldots, \beta_P)$, described as:

$$f_{map}(i) = (X_j * \beta)(i) + \epsilon = \sum_{p=1}^{P} \beta_p \cdot x^j_{t_{i-p}} + \epsilon \tag{2.18}$$

The benefit of extracting f_{map}, also known as *feature maps*, where the weights β are shared by shifting them along the $j-th$ row, is the reduced number of parameters to be learned. The structure of a CNN is shown in Figure 2.7

The different layers used to estimate feature maps in CNN are defined as follows:

- Convolutional layer: It convolves learnable kernel filters across the input data X^j to compute a dot product between the entries of the filter and the input. For each signal (and layer) we learn different kernel weights, thus, accounting for the multi-view nature of the data. As a result, the network learns different filters which aim to detect specific type of features at certain positions. To extract maximum local

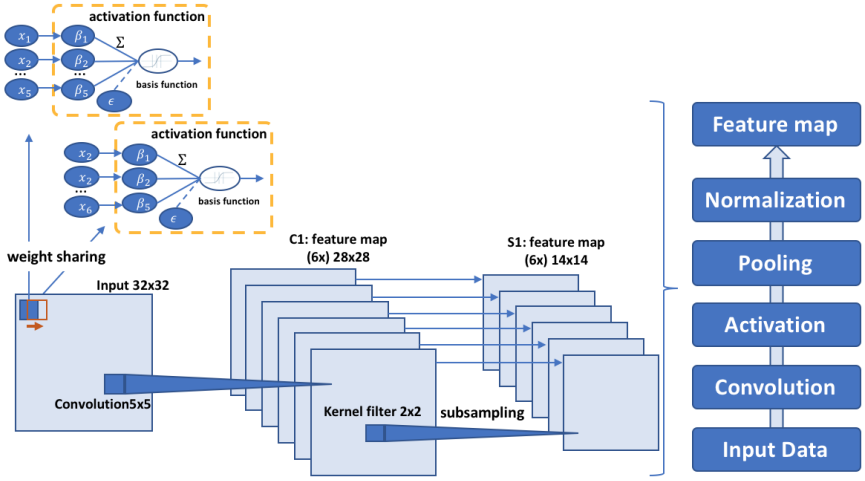

Figure 2.7: The Architecture of a Standard CNN.

temporal information from the whole data, it is beneficial to restrict the size P_l of the filters, where l is the corresponding layer.

- Activation layer: The results generated by the convolutional layer is given as an input to the activation function to capture some of the non-linearities from the input data. In Section 2.3.2, we already illustrated the different effects of frequently used non-linear functions in Figure 2.6. In [23], we used the exponential linear unit (ELU) activation function [13] as well as the frequently used rectified linear units (ReLUs), designed by Nair and Hinton [52].

- Pooling Layer: Finally, to down-sample the resolution of the input from the previous layer, the so called pooling also known as subsampling layer is used. The pooling layer can follow after a convolution or activation layer. In general, it generates translation invariant features by estimating statistics of the previous layer which in paper [23] are the resulting non-linearities of convolution activation from a small receptive field of the feature map. Some of the frequently used functions to generate invariant features are the mean or max functions, i.e. computing the average value in each neighborhood within the selected kernel pooling window, respectively the maximum value. The intuition that the exact location of features is not important but only its rough location relative to other features [66].

One interesting aspect gets visible by comparing the Equation 2.18 and the one of AR (Eq. 2.7), presented in Section 2.2.1. As it can be seen, both equations are very similar to each other by considering the preceding values in the considered time series – which explains the usefulness of CNNs for temporal data. However, in contrast to AR, the CNN learns in a discriminative fashion using labels assigned to each instance. In case of AR, such information are not available. In addition to this, a CNN is able of learning

non linearities from the data. These aspects explain the power of CNNs for classifying temporal data [23, 1, 43].

2.4 Summary and Contributions

Section 2, as an introduction to the state-of-the-art of this thesis, provided an overview of the principles of machine learning used to master the two challenges of a) detecting anomalous intervals in temporal data without labels and b) classifying instances to the two categories of anomalous and non-anomalous considering labeled data. For this purpose, the two classes of generative and discriminative models are introduced.

Section 2.2 focused on the class of unsupervised generative models such as autoregressive models (AR) with the assumption that the considered temporal data behaves stationary. This mindset can be used to draw conclusions about anomalous intervals when local variations and behaviors are noticed. These ideas are picked up in our papers [20, 25, 21]. Below, the contribution for each of these papers is summarized.

Our Contribution

1. In [20], an extension of the AR model is introduced by modeling the observed data as a mixture of a latent, autoregressive model representing the base behavior with a sparse anomaly signal. Based on this generative process, we derived an efficient algorithm, called Robust Latent Autoregression *(RLA)* which invokes a sequence of quadratic programs. The experiments carried out with RLA on E-Commerce data show how well it can be used to detect the potentially evolving base behavior of users as well as to spot local anomalies where the rating distribution differs from this base behavior. Since our method exploits the temporal characteristics of the data, it further allows to predict the rating distributions of future points in time, which subsequently allows to detect newly occurring anomalies.

2. In [25], a different approach, called SpotRate, based on a sound Bayesian framework is proposed which represents the rating data via temporally constrained categorical mixture models. The constraints enable to model local anomalies only occurring in specific time intervals. In contrast to RLA, SpotRate operates on the time-stamped rating data, thus, requiring no aggregation. As in [20], SpotRate accounts for the temporal evolution of the base behavior and enables us to predict the rating behavior for newly occurring ratings. Our experiments have shown the potential to detect local anomalies appearing at certain time intervals.

3. As the third paper in the field of unsupervised models, our work [21] focuses on the detection of rare anomalies/events in Twitter data. Considering the frequency of words appearing over time, we derived a statistical model for detecting significant deviations in the words' occurrences. To increase the interpretability of the results, our method not only uses single words to describe rare events, but derives descriptions consisting of multiple words. At the same time it ensures that the results do

not capture redundant information, thus, providing a compact summarize of all events in the data.

Section 2.3 introduced the principles of supervised machine learning methods enabling us to separate rare events from the rest. In this context, two different industrial dataset are analyzed. The contributions to this topic are listed below.

1. Classical models such as Logistic Regression (LR) do not perform well when considering rare events. Thus, extensions are required. In [22] we extended LR to handle rare events by incorporating cost-sensitive learning ideas. To this purpose multiple extended loss functions each having different theoretical properties and interpretations are designed. The evaluation on various datasets has shown that incorporating these cost-sensitive loss functions leads to increase performance for classification.

2. In [23], we exploited the principle of neural networks to detect malfunctioned engines as one form of global anomalies. Instead of using standard methods such as LSTM or RNN, we analyzed the performance of CNNs for temporal data since they capture similar to AR models local variations within time points. We designed our model on top of wavelet transformed temporal sensor signals, additional including non-temporal external data sources and accounting for the imbalance of rarely occurring malfunctions.

Overall, the summarized machine learning methods in this section are used and extended to solve different practical applications with focus on detecting rare events in temporal data. The following section presents these results.

3 Peer Reviewed Publications

This chapter includes the listed published papers of Table 1. All published papers are peer reviewed publications. The structure of this chapter is as follow: The first part covers papers related to the topic of unsupervised generative models which focus on detecting local anomalies. In this context, novel techniques are demonstrated. The papers *"Robust Multivariate Autoregression for Anomaly Detection in Dynamic Product Ratings"* [20], and *"Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution"* [25] deal with the challenge of detecting anomalous as well as normal behavior within rating data from E-commerce platform polluted by anomalies. *"Finding non-redundant multi-word events on Twitter"* [21] focuses on detecting rare events in social media by considering the users' behaviour based on their used vocabularies in tweets. An anomalous event is observed, if there is a deviation from the expected generated behavior and the observed data.

The second part of published papers, *"Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model"* [22], and *"Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals"* [23] deal with the challenge of classifying malfunction instances which are considered as global anomalies in this thesis. Since the class of anomalous instance are rare events, the observed data becomes massively imbalanced which makes generally the process of supervised learning very difficult. To this purpose, the first paper [22] introduces a new cost sensitive function which deals directly with the problem of imbalanced data classification. The second paper [23] presents the application of deep learning to use also the local behavior within temporal data in the new field of predicting malfunction engines.

3.1 Robust Multivariate Autoregression for Anomaly Detection in Dynamic Product Ratings

Peer-reviewed Conference Paper

Autors **Nikou Günnemann**, Stephan Günnemann, Christos Faloutsos

In *Proceedings of the 23rd international conference on World Wide Web, pp. 361-372, 2014, [20].*

Abstract User provided rating data about products and services is one key feature of websites such as Amazon, TripAdvisor, or Yelp. Since these ratings are rather static but might change over time, a temporal analysis of rating distributions provides deeper insights into the evolution of a products' quality. Given a time-series of rating distributions, in this work, we answer the following questions: (1) How to detect the base behavior of users regarding a product's evaluation over time? (2) How to detect points in time where the rating distribution differs from this base behavior, e.g., due to attacks or spontaneous changes in the product's quality?

To achieve these goals, we model the base behavior of users regarding a product as a latent multivariate autoregressive process. Using an autoregressive model enables us to realize a smooth base behavior by generating the current normal data dependent on the weighted values of previously measured ones combined with a vector which represents the trend. The latent base behavior generated in such a way is subsequently mixed with a sparse anomaly signal finally leading to the observed data. Since the anomaly signal is assumed to be sparse, we capture the idea of handling rare anomalies which appear only locally at some time points.

To infer the values of the hidden variables which best describe the observed data, our method uses a block coordinate-descent approach to alternatively optimize the variables. As we show, in each step our problem can be reduced to a quadratic programming, thus, ensuring efficient execution by using state-of-the-art quadratic program solvers.

The experimental results show that our method can be used to detect the potentially evolving base behavior of users as well as to spot anomalous points in time where the rating distribution differs from this base behavior. Since our method exploits the temporal characteristics of the data, it further allows to predict the rating distributions of future points in time, which subsequently allows to detect newly occurring anomalies.

Contribution of thesis author: Algorithmic design, discussions and implementation, experimental analysis, majority of manuscript writing, revision and editing.

Robust Multivariate Autoregression for Anomaly Detection in Dynamic Product Ratings

Nikou Günnemann Stephan Günnemann Christos Faloutsos
Carnegie Mellon University, USA
{nguennem, sguennem, christos}@cs.cmu.edu

ABSTRACT

User provided rating data about products and services is one key feature of websites such as Amazon, TripAdvisor, or Yelp. Since these ratings are rather static but might change over time, a temporal analysis of rating distributions provides deeper insights into the evolution of a products' quality.

Given a time-series of rating distributions, in this work, we answer the following questions: (1) How to detect the base behavior of users regarding a product's evaluation over time? (2) How to detect points in time where the rating distribution differs from this base behavior, e.g., due to attacks or spontaneous changes in the product's quality? To achieve these goals, we model the base behavior of users regarding a product as a *latent* multivariate autoregressive process. This latent behavior is mixed with a sparse anomaly signal finally leading to the observed data. We propose an efficient algorithm solving our objective and we present interesting findings on various real world datasets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications —*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

anomaly detection; robust autoregression; sparsity

1. INTRODUCTION

Rating scores about products and services are ubiquitous in today's websites such as Amazon, Yelp, or TripAdvisor. These ratings supply new customers information about the products' quality and support the decision making process which product to buy or service to use. In today's era of fast-paced developments in industry and growing competitions between companies and manufacturers, these ratings can also influence a companies production line. In fact, the ratings of customers can be considered as a benchmark for future sales performance of a product. To avoid negative results, companies can derive benefits from customer ratings by detecting functional weaknesses as well as deficiencies of products for improving the detected lacks.

In this work, we propose a method for the temporal analysis of rating distributions. Given a time-series of rating distributions, our goal is to extract the *base behavior* of users regarding the product's quality over time as well as to discover time points at which the product's evaluation shows *anomalous patterns*. The base behavior represents the general quality of a product accounting for temporal effects like, e.g., decreasing quality due to technical progress of competing products. The anomalies, in contrast, represent irregularities where the observed ratings deviate from the base behavior. These anomalies might occur, for example, due to spammers trying to push the success of a product or due to changes in the product's manufacturing process.

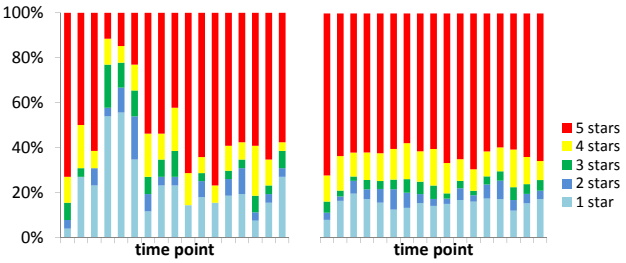

Figure 1: Left: Observed rating distributions of the product "coconut-water" over time. Right: Detected base behavior without anomalies.

A real world example for such an effect is illustrated in Figure 1 (left). The diagram illustrates the distribution of ratings over time for a "coconut water" sold on Amazon (more details about the data are given in the experimental section). The different colors represent the fraction of the star ratings – from 1 star (light blue) to 5 stars (red) – at a certain time. At the very beginning and at the majority of the later time points, the product received good ratings. During the time period 4-6, however, the rating distributions behave differently: one observes a high increase of negative ratings. As we will show in Section 5, these anomalies occurred due to a change in the products packaging – the previously used paper bottles were replaced by plastic bottles leading to an unpleasant aftertaste.

With our method, we aim at detecting such anomalies as well as the corresponding base behavior if the data would be "anomaly free". Let us anticipate and present the result of our method: As shown in Figure 1 (right), our method successfully detects the base behavior of users regarding the

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568008.

product, and, as we will see, it will provide further information about the anomalies and the points in time when they have been occurred.

Besides using our method to detect weaknesses in products and services, it can generally be used to find points in time where irregular ratings have been given. Thus, by filtering out these information new customers might be provided with a more precise evaluation of the product or – as another extreme – these irregular ratings can specifically be used to provide the whole picture on a product (since these anomalous ratings are otherwise hidden in the much larger set of normal behavior). Furthermore, our method allows to predict the rating distributions of future time steps exploiting the underlying base behavior. Thus, when new ratings arrive one can assess whether they match the predicted values, i.e. normal behavior, or whether they deviate, thus, indicating an anomaly.

The general idea of our method is to consider the base behavior as a *latent* multivariate autoregressive process, thus, incorporating the temporal dynamics of the data. The base behavior is then mixed with a latent, sparse anomaly signal to finally generate the observed data.

The contributions of our work are:

- *Novel mining task:* We present a technique for the temporal analysis of product ratings that detects the base behavior of users as well as potential anomalies enriched with their points in time when they occurred. Furthermore, our technique can be used to predict the rating distributions at future points in time.
- *Theoretical soundness:* We base our method on a sound generative process that models dynamic rating distributions by mixing a latent autoregressive process with a latent and sparse anomaly signal.
- *Algorithm design:* We develop an efficient algorithm that solves our objective by invoking a sequence of quadratic programs. As a further benefit, our method does not require user-defined parameters.
- *Effectiveness:* We evaluate our method on different real world datasets and we show its effectiveness by presenting interesting findings.

While the rating of products or services has been studied extensively in other research areas such as, e.g., recommender systems and opinion mining (cf. Section 4), the goals of these areas are completely different to our work. To the best of our knowledge, there are no works that consider the temporal analysis of rating distributions incorporating potentially anomalous behavior.

2. THE RLA MODEL

In this section, we introduce our RLA model (Robust Latent Autoregression) for detecting the users' base rating behavior and anomalies in rating distributions. Following convention, we do not distinguish between a random variable X and its realization $X = x$ if it is clear from the context. Vectors of (random) variables are written in bold font, e.g. $\boldsymbol{b}$, while the entries of the vectors are given in standard font, e.g. b_i. The number of time steps is denoted with T and the number of dimensions with D.

Before formally introducing our objective, we discuss the data, used as the input for our model, in more detail.

Preliminaries - Data Representation. In our work, we aim at analyzing the temporal evolution of rating distributions. We assume that users can choose ratings based on an ordinal rating scale with M different ratings (e.g. star ratings from 1 to M). Correspondingly, a distribution over these ratings can be represented by a M-dimensional vector $\mathbf{r} \in [0 \dots 1]^M$ with $\sum_{d=1}^{M} r_d = 1$, where r_i denotes the fraction of ratings with value i. The raw data we process is a time-series $\mathbf{R} = (\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(T)})$ of such rating distributions, i.e. a multivariate time-series of length T.

Since ratings are given on an ordinal scale, we use the established principle of analyzing the *cumulative distributions* instead of the raw data [2]. That is, instead of considering $\mathbf{r}^{(t)}$, we analyze the cumulative distributions $\mathbf{x}^{(t)}$ with $x_d^{(t)} = \sum_{d'=1}^{d} r_{d'}^{(t)}$. By analyzing the cumulative distributions we preserve the ordering of the ratings and we can better describe differences in rating distributions.

Consider, e.g., the (non-cumulative) distributions $\mathbf{a} = [1, 0, 0, 0]$, $\mathbf{b} = [0.5, 0.5, 0, 0]$, and $\mathbf{c} = [0.5, 0, 0, 0.5]$. Intuitively, the distributions $\mathbf{a}$ and $\mathbf{b}$ are more similar to each other than $\mathbf{a}$ and $\mathbf{c}$, since $\mathbf{a}$ represents only 1-star ratings, $\mathbf{b}$ 1&2-star ratings and $\mathbf{c}$ 1&4-star ratings. By considering the cumulative distributions $\mathbf{a}' = [1, 1, 1, 1]$, $\mathbf{b}' = [0.5, 1, 1, 1]$, and $\mathbf{c}' = [0.5, 0.5, 0.5, 1]$ this similarity structure is directly visible (even without doing cross-dimension comparisons).

Since the last entry of the cumulative distribution is always 1, we can safely ignore it from our considerations. Additionally, to be a valid cumulative distribution, the values of $x_d^{(t)}$ have to be non-decreasing in d. As an abbreviation for later use, we define

$$\mathcal{C}_D := \{\mathbf{x} \in [0 \dots 1]^D \mid \forall i : x_i \leq x_{i+1}\}$$

Overall, in the remainder of this paper, we consider the data $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)})$ where each $\mathbf{x}^{(t)}$ has dimensionality $D := M - 1$ and $\mathbf{x}^{(t)} \in \mathcal{C}_D$.

For easier interpretation, when plotting data, we always use the non-cumulative distribution.

Generative Process. Given the observed time series $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)})$, our aim is to extract the base behavior of the users and the corresponding points in time where anomalies occur. The challenge of this task is that the observed data $\mathbf{X}$ is already polluted by anomalies, thus, directly using it for estimating the base behavior might be misleading.

In our model, we solve this issue by assuming that the observed time series $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)})$ is obtained by mixing the base (but unknown) user behavior $\mathbf{A} = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(T)})$ with an (also unknown) anomaly behavior $\mathbf{y}$. Thus, the base behavior $\mathbf{A}$ acts as a *latent* variable which is not directly observed but inferred by our technique.

An overview of our generative process showing the used variables and their dependencies is illustrated by the graphical model in Figure 2. More formally, we assume that the observed data follows the random generative process

$$\mathbf{x}^{(t)} = p_t \cdot \mathbf{a}^{(t)} + (1 - p_t) \cdot \mathbf{y} + \boldsymbol{\varepsilon}_t \quad \forall t = 1 \dots T \qquad (1)$$

where $p_t \in [0 \dots 1]$ is the mixing coefficient at time t and $\boldsymbol{\varepsilon}_t$ corresponds to white noise (e.g. a normal distribution). The higher p_t the stronger the effect of the base user behavior at a certain point. Thus, the vector $\mathbf{p}$ is effectively the indicator where the anomalies have been occurred. To ensure the interpretability of the model, we require that the base

behavior as well as the anomaly behavior are valid cumulative distributions, i.e. $\mathbf{a}^{(t)}, \mathbf{y} \in \mathcal{C}_D$.

At this point it is important to highlight the difference between outliers and anomalies – two different kinds of irregular data (since this difference is not consistently handled in the literature, we describe our notion here). *Outliers* are irregular behavior that can be attributed to mostly random corruptions of the data (like, e.g., measurement errors). *Anomalies*, in contrast, are irregular behavior that follow a specific pattern (like, e.g., time points with consistently low ratings due to a change in the product's quality). In our method, we consider anomalous behavior, whose characteristic is described by the distribution $\mathbf{y}$.

While the observed data $\mathbf{X}$ might show abrupt changes due to the anomalies, we assume that the base behavior changes smoothly over time. An established principle to model such behavior is the use of (vector) autoregressive models [20, 21], where the values observed at time t depend on the values of previously observed time points allowing slight perturbations. Based on this general idea, we define the base behavior to be generated by the random process

$$\mathbf{a}^{(t)} = w \cdot \mathbf{a}^{(t-1)} + (1-w) \cdot \mathbf{b} + \hat{\boldsymbol{\varepsilon}}_t \quad \forall t = 1 \dots T \qquad (2)$$

Here, $w \in [0 \dots 1]$ determines the importance of the previous time step, $\mathbf{b} \in \mathcal{C}_D$ is a vector, which represents the trend, and $\hat{\boldsymbol{\varepsilon}}_t$ is again white noise.[1] The value $\mathbf{a}^{(0)}$, required for a valid recursion, represents the base behavior of the users at (the virtual) time point 0.

We want to mention two important differences of Equation 2 in contrast to usual autoregressive models: First, we use a convex combination of the vectors $\mathbf{a}^{(t-1)}$ and $\mathbf{b}$, i.e. we use weights w and $w' = 1 - w$. In general, autoregressive models allow unbounded values for w and w'. The convex combination leads to the huge advantage of generating only "valid" data, i.e. if $\mathbf{a}^{(t-1)}$ and $\mathbf{b}$ represent valid (cumulative) distributions, the convex combination is also a valid (cumulative) distribution. This property is in particular useful, when using the autoregressive model for predicting future time steps: in this case, the error term will be equal to zero and, thus, the convex combination leads to a prediction representing a valid distribution. As we will see in the experimental analysis, this aspect is neglected by all competing approaches making their results difficult to interpret.

Second, unlike to traditional autoregressive models, where the values of $\mathbf{a}$ are observed (and thus fixed), we consider a *latent* autoregressive model. Thus, similar to the values of w, and $\mathbf{b}$, the values of $\mathbf{a}$ are variables.

Even though Equation 1 and 2 look similar at a first sight, their underlying principle is very different. While the weights p_t might change over time to allow different strength of anomalies, the weights w are constant reflecting the smooth base behavior. While the value of $\mathbf{x}^{(t)}$ depends only on values of the same time point, the value of $\mathbf{a}^{(t)}$ depends on the values of the previous time point. And most important, while the values of $\mathbf{x}^{(t)}$ are observed, the values of $\mathbf{a}^{(t)}$ are latent.

Sparsity. So far, we assumed that the values of p_t are independent of each other. Thus, scenarios where we might observe "anomalies" at every point in time ($p_t < 1$ for all t) are possible. Since such a scenario would contradict the idea of an anomaly, it is reasonable to require that anomalies are rare events. Technically speaking, we want to enforce the vector $\mathbf{1} - \mathbf{p}$ to be sparse.

[1]Equation 2 is based on a first-order autoregressive process. Extensions to higher order processes are straightforward, though, not focus of this work.

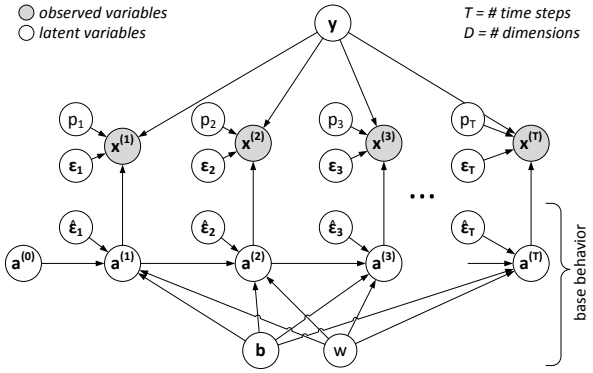

Figure 2: Graphical model of RLA. The observed data is a mixture of the (potentially evolving) base behavior $\mathbf{a}^{(t)}$ with the anomaly behavior y.

Let λ denote the maximal number of anomalies a user expects in the data. We require $\|\mathbf{1} - \mathbf{p}\|_0 \leq \lambda$ where $\|.\|_0$ denotes the L_0 pseudo-norm and $\mathbf{1}$ is the vector consisting of only ones. In this case, the variables p_t are no longer independent since we effectively use the prior distribution

$$p(\mathbf{p}) \propto \begin{cases} const & \text{if } \|\mathbf{1} - \mathbf{p}\|_0 \leq \lambda \\ 0 & \text{else} \end{cases} \qquad (3)$$

from which the values of $\mathbf{p}$ are drawn. For illustration purposes we do not show this dependency in our graphical model in Figure 2.

The above prior distribution is beneficial due to multiple reasons: First, it is easy to interpret. By using the L_0 norm, the value of λ represents simply an upper bound on the number of anomalies and one can precisely spot the time steps where an anomaly has occurred. Second, the distribution is essentially the maximum-entropy distribution taking into account the upper bound of λ anomalies. That is, it is a non-information prior which is not biased to certain $\mathbf{p}$ vectors. Finally, as we will show later, by using the above prior we can use a model selection approach to determine the parameter λ automatically. Thus, the user does not need to provide it by hand.

Objective. The ultimate goal of our method is to infer the values of the hidden variables which best describe the observed data. There are multiple ways to formulate this objective. In this work, we are interested in finding the maximum likelihood solution w.r.t. the complete data. That is, we aim at maximizing the joint probability $p(\mathbf{X}, Z)$ where Z denotes the set of all variables. To make this problem well-defined we finally have to select appropriate prior distributions. We simply assume independent, identically distributed Gaussian error, i.e. $\boldsymbol{\varepsilon}_t, \hat{\boldsymbol{\varepsilon}}_t \sim \mathcal{N}(0, \sigma^2 \cdot I)$. In this case, the variables $\boldsymbol{\varepsilon}_t, \hat{\boldsymbol{\varepsilon}}_t$ can be absorbed into Equation 1 and Equation 2 by writing

$$\mathbf{x}^{(t)} \sim \hat{\mathcal{N}}(p_t \cdot \mathbf{a}^{(t)} + (1-p_t) \cdot \mathbf{y}, \sigma^2 \cdot I)$$
$$\mathbf{a}^{(t)} \sim \hat{\mathcal{N}}(w \cdot \mathbf{a}^{(t-1)} + (1-w) \cdot \mathbf{b}, \sigma^2 \cdot I)$$

Here, $\hat{\mathcal{N}}$ denotes the Normal distribution restricted to the domain $\mathcal{C}_D$ since we require $\mathbf{a}^{(t)}, \mathbf{x}^{(t)} \in \mathcal{C}_D$.

Additionally, we select non-informative priors for the values of $\mathbf{y}$, $\mathbf{a}^{(0)}$, $\mathbf{b}$, and w. That is,

$$p(\mathbf{a}^{(0)}) \propto \begin{cases} const & \text{if } \mathbf{a}^{(0)} \in \mathcal{C}_D \\ 0 & \text{else} \end{cases}$$

and accordingly for $\mathbf{y}$, $\mathbf{b}$, and w. Note that these are valid priors since the domain of the vectors is bounded.

Using this setting, we can derive that the likelihood, if all variables are in their valid domains, is proportional to

$$p(\mathbf{X}, Z) \propto \prod_{t=1}^{T} p(\mathbf{a}^{(t)}|\mathbf{a}^{(t-1)}, \mathbf{b}, w) \cdot p(\mathbf{x}^{(t)}|\mathbf{a}^{(t)}, \mathbf{y}, \mathbf{p}) \tag{4}$$
$$\propto \frac{1}{\sigma^{D \cdot T \cdot 2}} \cdot exp(-\frac{1}{2 \cdot \sigma^2} f(\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, w, \mathbf{p}))$$

where the function f is given as

$$f(...) := \sum_{t=1}^{T} \Big(\Big\| \mathbf{x}^{(t)} - p_t \cdot \mathbf{a}^{(t)} - (1 - p_t) \cdot \mathbf{y} \Big\|_2^2 + \Big\| \mathbf{a}^{(t)} - w \cdot \mathbf{a}^{(t-1)} - (1 - w) \cdot \mathbf{b} \Big\|_2^2 \Big)$$

Solving for the optimal value of σ^2 that maximizes this equation leads to $\sigma^2 = \frac{f(...)}{2 \cdot D \cdot T}$ (assuming $f \neq 0$, otherwise $\sigma^2 \to 0$ is the optimal solution). Using this result in Equation 4, the likelihood finally becomes proportional to $1/f(\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, w, \mathbf{p})^{D \cdot T}$. Thus, overall, maximizing Equation 4 corresponds to solve the following objective

$$\min_{\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, w, \mathbf{p}} f(\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, w, \mathbf{p}) \tag{P1}$$

subject to

$$\|\mathbf{1} - \mathbf{p}\|_0 \leq \lambda \ \wedge \ \mathbf{a}^{(t)}, \mathbf{b}, \mathbf{y} \in \mathcal{C}_D \ \wedge \ 0 \leq w \leq 1 \ \wedge \ 0 \leq p_t \leq 1$$

This formulation intuitively states that the optimal solution is the one that minimizes the (squared) residuals regarding the observed data and the base user behavior.

Model Selection. Solving problem P1 leads to the optimal solution w.r.t. a specific upper bound λ on the number of anomalies. To determine the value of λ, we use a model selection principle. Since we use the L_0 pseudo-norm as our constraint, λ effectively controls the number of free parameters in our model. Increasing λ by a value of one, increases the number of free parameters by two: We allow one additional p_t value to vary, plus we have the freedom to decide at which time t this happens.[2] One exception represents the step from $\lambda = 0$ to $\lambda = 1$, which increases the number of free parameters by $2 + D$. The additional term D accounts for the vector $\mathbf{y}$ which is not used in the case of $\lambda = 0$.

Given these observations, we use the Bayesian information criterion [5] to determine the optimal value of λ. That is, we choose the λ minimizing

$$BIC(\lambda) = -2 \cdot \ln L_\lambda + k_\lambda \cdot \ln(D \cdot T)$$

where $k_\lambda = m + 2 \cdot \lambda + D \cdot \min(\lambda, 1)$ is the number of free parameters determined based on the observations made above. Here, the term m represents the remaining parameters of our model which are not effected by the choice of λ and which, therefore, do not effect the choice of the optimal λ (i.e. for the model selection step we can simply set $m = 0$). The term L_λ is the likelihood of the data which in our case simplifies to $L_\lambda = 1/f(\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, w, \mathbf{p})^{D \cdot T}$ (cf. above).

Special Cases. We briefly want to discuss two special cases of our model. First, when fixing $p_t = 1$ for all t, our model reduces to a special case of Kalman filtering [5], which is not able to detect the anomaly signal $\mathbf{y}$. Thus, if anomalies are present in the data, the estimates of the Kalman Filter might be highly corrupted. Second, when fixing $w = 0$, we ignore the temporal effects of the data modeled via autoregression. In this case, only the vectors $\mathbf{b}$ and $\mathbf{y}$ can be used to generate the data. When additionally dropping the sparsity constraint, the data corresponds to a mixture of two static components as similar done by classical matrix factorization techniques. Further relations to existing methods can be found in Section 4.

Prediction. Besides detecting the base behavior and anomalies for the given data, our method can also be used to predict the rating distributions of future points in time. Assuming that anomalies are rare, it is natural to exploit only the base behavior for the prediction step. Thus, the most likely (cumulative) distribution at time $T + 1$ is

$$\tilde{\mathbf{x}}^{(T+1)} = w \cdot \mathbf{a}^{(T)} + (1 - w) \cdot \mathbf{b} \tag{5}$$

Here, since a-priori no information about the error at the next time step is known, the original error terms have been replaced with their expected value of 0. Comparing the observed distribution at time $T + 1$ against the predicted one gives an indicator whether a new anomaly has been observed.

3. ALGORITHM

Finding an exact solution to problem P1 is infeasible due to two reasons: First, albeit natural, the sparsity constraint via the L_0 norm is NP-hard to optimize in general [8, 24]. Second, the objective function is neither convex nor concave, which prevents the (direct) application of efficient convex optimization solvers. Thus, in the following, we present an algorithm computing a near-optimal solution. An overview of our method is given in Algorithm 1.

Sparsity. One standard solution to tackle the complexity of the L_0 norm is to replace it with the L_1 norm. As shown in [7], however, an iterative reweighting of the L_1 norm outperforms the simple (unweighted) L_1-norm in many situations. In preliminary experiments, this effect has also been observed in our scenario: by using the unweighted L_1-norm some values in the $\mathbf{1} - \mathbf{p}$ vector were close to (but not exactly) 0 leading to a bad approximation of the L_0-norm.

Thus, we follow the study of [7] and we replace the L_0 norm with the weighted L_1 norm, where the weights are iteratively updated based on the previously determined solution. This principle leads to a sequence of optimization problems (cf. line 12 of Algorithm 1). Technically, we use the constraint

$$\|Z \cdot (\mathbf{1} - \mathbf{p})\|_1 \leq \tau \tag{6}$$

as a proxy for $\|\mathbf{1} - \mathbf{p}\|_0 \leq \lambda$, where $Z = diag(z_1 \ldots, z_T)$.

Initially, all weights z_t are set to 1, corresponding to the unweighted L_1 norm. After solving our objective using this constraint, the weights are recomputed based on $\mathbf{p}$:

$$z_t = \frac{1}{1 - p_t + \delta} \cdot \max_t (1 - p_t + \delta) \tag{7}$$

[2] It is fair to mention that the value 2 is a slight overestimate: Due to the inequality constraint $\leq \lambda$, we essentially sample the timepoints where anomalies occur with replacement. Thus, with some probability we sample an already drawn point. Being aware of this effect, we can conclude that our model selection approach might slightly underestimate the optimal value of λ.

where δ is a very small constant to ensure that the fraction is always well defined (default $\delta = 10^{-4}$). The idea is that values p_t which are close to 1 will get higher weights z_t. Therefore, in the next iteration of our method, they are stronger penalized when deviating from 1 (effectively steering these values to become exactly 1). An analytical justification for this principle is given in [7].

It is worth mentioning that Equation 7 differs from the equation proposed in [7] by introducing the multiplicative factor $\max_t(1-p_t+\delta)$. Since in [7] the weighted norm is used as the *objective function*, any (positive) multiplicative factor would lead to the same result. In our scenario, however, we use the weighted L_1 norm as a *constraint*. If we would allow arbitrary small values for z_t, the constraint effectively becomes useless, i.e. it is no longer guaranteed that $\mathbf{1}-\mathbf{p}$ is sparse. In contrast, by using our definition it holds that $z_t \geq 1$ and, therefore,

$$\|Z \cdot (\mathbf{1}-\mathbf{p})\|_1 \geq \|(\mathbf{1}-\mathbf{p})\|_1 \tag{8}$$

Indeed, the factor $\max_t(1-p_t+\delta)$ leads to the tightest possible solution that fulfills the above equation. Combining Equation 8 with Equation 6, it becomes clear that during each iteration of our method we guarantee

$$\|(\mathbf{1}-\mathbf{p})\|_1 \leq \|Z \cdot (\mathbf{1}-\mathbf{p})\|_1 \leq \tau$$

and, therefore, the sparsity of the vector $\mathbf{p}$ is realized.

Since the values of p_t (and thus $1-p_t$) are bounded between 0 and 1, Equation 6 can directly be written as

$$\sum_{t=1}^{T} z_t \cdot (1-p_t) \leq \tau \quad \Leftrightarrow \quad \sum_{t=1}^{T} -z_t \cdot p_t \leq \tau - \sum_{t=1}^{T} z_t \tag{9}$$

which corresponds to a simple, linear constraint on the values of $\mathbf{p}$.

Reduction to quadratic programs. While the objective function of our original problem definition is not jointly convex in all variables, we can observe the following: Given the values of $\mathbf{p}$ and w, the objective function is convex in $\mathbf{a}^{(*)}$, $\mathbf{y}$ and $\mathbf{b}$; and simultaneously, given $\mathbf{a}^{(*)}$, $\mathbf{y}$, and $\mathbf{b}$, the objective function is convex in $\mathbf{p}$ and w.

We show an even stronger result: both problems (i.e. conditioned either on $\mathbf{p}, w$ or $\mathbf{a}^{(*)}, \mathbf{y}, \mathbf{b}$) are instances of a quadratic program.

THEOREM 1. *Given* $\mathbf{p}$ *and* w. *Let* $(\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_T, \hat{\mathbf{b}}, \hat{\mathbf{y}}, \mathbf{p}, w)$ *be the optimal solution of problem P1 (when fixing the values of* $\mathbf{p}$ *and* w*). Set* $\hat{\mathbf{u}} = stack(\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_T, \hat{\mathbf{b}}, \hat{\mathbf{y}})$*, where stack denotes the stacking of multiple vectors to a single one.*

There exist vectors $\mathbf{c}^\bullet$, $\mathbf{d}^\bullet$ *and (sparse) matrices* $\mathbf{Q}^\bullet$, $\mathbf{A}^\bullet$ *such that*

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u} \in \mathbb{R}^{(T+3)\cdot D}} \frac{1}{2} \cdot \mathbf{u}^T \cdot \mathbf{Q}^\bullet \cdot \mathbf{u} + (\mathbf{c}^\bullet)^T \cdot \mathbf{u} \text{ subject to } \mathbf{A}^\bullet \cdot \mathbf{u} \leq \mathbf{d}^\bullet$$

PROOF. See appendix □

THEOREM 2. *Given* $\mathbf{a}^{(*)}, \mathbf{y}$, *and* $\mathbf{b}$. *Let* $(\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(T)}, \mathbf{b}, \mathbf{y}, \hat{\mathbf{p}}, \hat{w})$ *be the optimal solution of problem P1 (when fixing the values of* $\mathbf{a}^{(*)}, \mathbf{y}, \mathbf{b}$*). Set* $\hat{\mathbf{v}} = stack(\hat{\mathbf{p}}, \hat{w})$.

There exist vectors $\mathbf{c}^\circ$, $\mathbf{d}^\circ$ *and (sparse) matrices* $\mathbf{Q}^\circ$, $\mathbf{A}^\circ$ *such that*

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v} \in \mathbb{R}^{T+1}} \frac{1}{2} \cdot \mathbf{v}^T \cdot \mathbf{Q}^\circ \cdot \mathbf{v} + (\mathbf{c}^\circ)^T \cdot \mathbf{v} \text{ subject to } \mathbf{A}^\circ \cdot \mathbf{v} \leq \mathbf{d}^\circ$$

PROOF. See appendix □

Data: series of rating distributions $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)})$
1 $z_1 = \ldots = z_T = 1$;
2 initialize $\mathbf{p}, w$;
3 **until** *convergence* **do**
4 **until** *convergence* **do**
5 recompute $\mathbf{Q}^\bullet, \mathbf{c}^\bullet$;
6 determine $\hat{\mathbf{u}}$ via quad. programming (cf. Theo. 1);
7 set $\mathbf{a}^{(*)}, \mathbf{b}, \mathbf{y}$ based on $\hat{\mathbf{u}}$;
8 recompute $\mathbf{Q}^\circ, \mathbf{c}^\circ$;
9 determine $\hat{\mathbf{v}}$ via quad. programming (cf. Theo. 2);
10 set $\mathbf{p}, w$ based on $\hat{\mathbf{v}}$;
11 **end**
12 update weights z_t based on Eq. 7
13 **end**

Algorithm 1: The RLA algorithm

The above two results make it possible to use highly efficient solvers for quadratic programming in our scenario, in particular by exploiting the fact that the matrices are sparse.

Overall method. As shown in Algorithm 1, our method uses a block coordinate-descent to alternatively optimize the variables. In each step it invokes a quadratic programming solver (line 6 & 9). After updating one set of variables, we recompute the corresponding matrices/vectors according to Theorem 1/2, which subsequently are used to compute the optimal solution for the other set of variables. Per default, we initialize $\mathbf{p}$ with high values since we expect only few anomalies in the data. w is initialized randomly. We assume convergence if the relative change in the objective function value is less than 0.01%. Overall, our method allows to efficiently compute a near-optimal solution of problem P1.

4. RELATED WORK

The main objective of our work is to spot anomalies in dynamic rating data by simultaneously detecting the users' base behavior evolving through the time. In the following, we discuss methods related to our approach. We first discuss methods that are related to our method from a technical perspective (i.e. the methodology of modeling the data is related). Afterwards, we give a broader perspective on methods whose application domain is related to ours.

Related Techniques. *Autoregression* (AR) models [21] represent data as a random process based on previous time points as well as noise. An AR model can be written as $x_t = \sum_{m=1}^{M} w_m \cdot x_{t-m} + \epsilon$ where x is the point series under investigation, M is the order of the AR model, w_m are some coefficients taking arbitrary values and finally ϵ is noise. While traditional autoregression models only consider univariate data, multivariate extensions (so called *vector autoregression* (VA) models) have been proposed [20, 18].

In contrast to our method, where the autoregression model is used as a latent process, the existing techniques consider directly the observed data. Thus, these approaches are not robust to anomalies in the data. It is a well known fact that in these models an anomaly (or in general a so-called shock) effects all time points infinitely far into the future. Thus, these models fail to find good approximations of the data corrupted by anomalies.

Kalman Filter/Smoother [5] can be considered as a generalization of first-order vector autoregressive models. While Kalman Filters are very flexible in generating the data, they share a similar drawback as AR/VA models. They are sensitive to anomalies in the data. The relation to our model to Kalman Filter has already been discussed in Section 2.

Robust extensions of vector autoregression [9] and Kalman Filtering [25] to handle outliers have been proposed. These methods try to circumvent errors in the data by, e.g., down-weighting the effect of high residuals (which usually occur due to outliers) via, e.g., the Huber loss function.

Again we want to highlight the difference between outliers and anomalies as already described in Section 2. *Outliers* are abnormal behavior attributed to mostly independent, random corruptions of the data, while *anomalies* are abnormal behavior following a specific pattern. It is clear that outliers and anomalies are two different concepts which should be handled differently. While all the above mentioned methods handle outliers, our method is specifically designed to handle anomalies.

Finally, as already described in Section 2, our model uses convex combinations of the involved variables, thus ensuring that the final results are valid distributions and leading to easy interpretability. The above techniques do not enforce such a constraint leading to potentially questionable results.

We compare our method against Kalman Filtering, a vector autoregression technique [18] and an extension to handle outliers [9] in our experimental section.

Matrix decomposition techniques (e.g., PCA) are designed to decompose a data matrix into products or sums of other matrices exhibiting special structure (e.g. low-rank or sparsity). These techniques do not consider the temporal characteristics of the data but they treat each time point as an independent observation, thus, ignoring highly important information. In particular, these techniques can not be used to predict future points in time based on the current observations. Similar as the temporal methods described above, the standard versions of these methods are not robust to anomalies or outliers. While *robust extensions* considering outliers have been proposed [6, 26], these methods are still not aware of the data's temporal characteristics.

In the experimental section, we compare our RLA method against ICA (Independent Component Analysis) [10], PCA (Principal Component Analysis) [13], NNMF (Non-Negative Matrix Factorization) [4], and robust NNMF [26], the latter one being a robust extension to handle outliers.

Related Applications. Our method can be used to detect anomalies in a time-series of rating distributions and to describe the base behavior of users regarding the considered product. In the following we discuss related applications.

Multiple techniques, considering various data types, have been proposed in the area of *outlier detection* [1]. While the majority of techniques tackles the case of independently distributed data, time-series outlier detection and outlier detection for streaming data are also an active field of research [1]. Both areas differ from our work. In time-series outlier detection one considers a set of time-series, where some of the time-series might be outliers. In contrast, we operate on a single multivariate time-series where some points in time might be anomalous. Streaming outlier detection assumes that the data under considering is not completely given but successively arrives over time – additionally coping with further aspects as limited storage of the data. In our work, however, we assume the complete history of data is given, thus, allowing enhanced analysis capabilities. Again, we have to note that most existing techniques consider outlier in the sense of independent, random errors in the data. We, in contrast, assume the anomalies to be generated from a specific anomalous behavior.

Change detection techniques try to detect points in time where the state of the underlying system has changed [16, 14]. A change might not generally indicate anomalous behavior. Indeed, since we model the base behavior as an autoregressive model, even the base behavior might change over time. Thus, change detection techniques cannot directly solve the problem tackled in this work.

The study of product ratings has been done in multiple research areas, all following different goals and objectives. *Recommender Systems* often incorporate ratings and their temporal information [12, 15, 22] to improve the prediction performance. Their goal is to recommend products or services to users which they most likely are going to use. *Opinion mining* or sentiment analysis aims at extracting the sentiment of users regarding specific products or features of a product [19, 3, 23]. This way, opinion mining can be considered as a technique to infer the ratings. *Modeling* of social rating networks, e.g. to compactly describe the underlying mechanism driving the network or to generate synthetic data, has been studied in, e.g., [11, 17].

None of the existing methods is designed to detect anomalies in a time-series of rating distributions under consideration of a potentially evolving base behavior.

5. EXPERIMENTAL ANALYSIS

In the following, we empirically analyze our RLA method.

Datasets. We applied our system on over hundred thousands of product ratings representing a variety of categories: 1) We used an extract of the Amazon website evaluating different food products[3] (400k ratings). 2) We analyzed ratings of restaurants/services in the area of Phoenix based on an extract of the Yelp website[4] (230k ratings). 3) Using an extract of the TripAdvisor website[5] (250k ratings), we used our method for studying hotel ratings.

The data consists of the IDs of the products/services to be rated as well as the related user IDs who evaluated them with star rating scores from 1 up to 5 at different timesteps (in the case of TripAdvisor, the rating scores range from 0 up to 5). Additionally, these datasets contain textual reviews, which we used to understand and describe the results of our method. Since our method is designed to handle rating distributions, we can flexibly process the raw data by, e.g., using equi-width binning or equi-depth binning. We decided to use equi-depth binning (each 25 temporally successive ratings are aggregated) to avoid sparsity effects.

Besides these real world datasets we used synthetic data generated based on the presented process to analyze the scalability and robustness of our method.

Runtime analysis. We start with a brief runtime analysis. The runtime of our method is primarily affected by two characteristics: the length T as well as the dimensionality D (i.e. the number of different ratings) of the time-series. To analyze these effects, we generated time-series of different length (100 to 10,000) with a small percentage of anomalies (5%) according to our model. We used the dimensionalities 4, 5, and 6 since these values reflect the properties of the frequently used rating schemes. All experiments were conducted on commodity hardware with 3 GHz CPU's and 4 GB main memory.

[3] `http://snap.stanford.edu/data/`
[4] `http://www.yelp.com/dataset_challenge/`
[5] `http://sifaka.cs.uiuc.edu/~wang296/Data/`

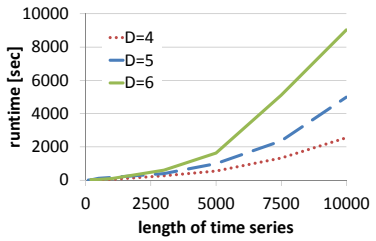

Figure 3: Runtime vs. length of time-series

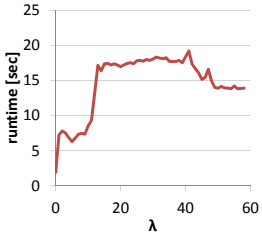

Figure 4: Runtime vs. λ (sparsity)

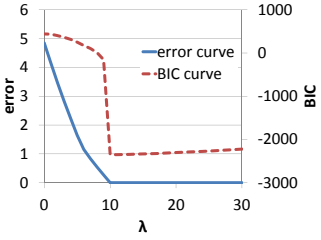

Figure 5: Error & BIC vs. λ (synthetic data)

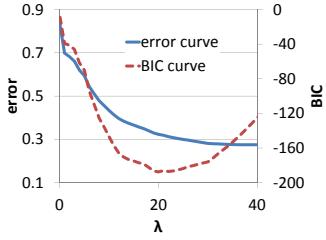

Figure 6: Error & BIC vs. λ (real world data)

The results are illustrated in Figure 3. The runtime increases approximately quadratic in the length of the time-series; similar to (or even better than) many matrix factorization techniques. While the quadratic complexity might be theoretically a problem for very long time-series, we want to mention that the absolute runtimes are small. For a length of 10,000 (which would already correspond to more than 27 years of data when measured on a daily basis – longer than most rating websites exist), the absolute runtime for, e.g., 5 ratings is around 80 minutes on commodity hardware.

In Figure 4, we analyze the effect of a varying λ on the runtime. In the figure, we plotted the results for an exemplary hotel from the TripAdvisor database. We observe an interesting behavior: Starting from very small values of λ, an increase in λ also increases the runtime. This effect might be attributed to the larger amount of variables the model now needs to fit. Given a certain value of λ, however, the runtime behaves almost constantly. Even more interestingly, when selecting a very high value, the runtime starts decreasing again. This last effect might be explained as follows: when choosing very large λ values, the p_t values can be optimized almost independently. Decreasing the value of λ, however, increases the dependency, thus, leading to higher runtimes. We observed a similar behavior across the various datasets on all of the three studied databases.

Effectiveness and Robustness. Next, we analyze the effectiveness and robustness of our method. We again start with synthetic data (100 time points, 10 anomalies, 5 dimensions). Figure 5 shows the minimal sum-of-squared-errors value (i.e. the value of the function f; cf. P1) obtained by our method when varying the parameter λ. Obviously, when increasing the value of λ, the error decreases since more flexibility is provided. We observe a high decrease in the error until reaching the value of $\lambda = 10$, corresponding to the true number of anomalies in the data. Afterward, the gain of allowing further anomalies decreases. The absolute error obtained by our method is small, showing that we well describe the underlying data.

On the second y-axis of Figure 5, we illustrate the BIC values. As shown, the minimal BIC value is obtained for the value of 10. Thus, the BIC gives a good guideline which λ parameter to choose.

Figure 6 shows the same type of plot for a real world dataset (hotel Punta Cana Princess, TripAdvisor – the original data is shown in Figure 7, left). Again, the absolute error converges to a very small value; our method leads to a good approximation. Additionally, the BIC curve clearly indicates which λ to choose. In Figure 7, the middle diagram shows the detected base behavior for this product. The behavior evolves smoothly over time and shows the general trend for this hotel containing of primarily high ratings. Mixing this base behavior with the detected anomalies leads to the right plot in Figure 7, which almost perfectly recovers the original data.

In Figure 8 we analyze for the same dataset how the error decreases with an increasing number of iterations of our algorithm until convergence (i.e. on the x-axis we count how often the inner-loop of Algorithm 1 has been executed, while the y-axis shows the obtained error). We parametrized our method with different λ values to show different effects. As expected, for all λ values, the first iterations (around 10) lead to the highest decrease of the error. Interesting to note are the sudden increases in the error for some of the curves at some points in time. At these points in time, the inner-loop of Algorithm 1 has been converged, thus, leading to an update of the weights enforcing the sparsity (Eq. 7). After updating the weights, our method is forced to select other (more sparse) solutions for $\mathbf{p}$, which obviously increase the error. After each sudden increase, the error term again decreases corresponding to the iterations of the inner-loop of our algorithm. Indeed, these increases show that the outer loop of our method is actually effective, i.e. it steers the $\mathbf{p}$ vector to other solutions. Furthermore, the figure shows

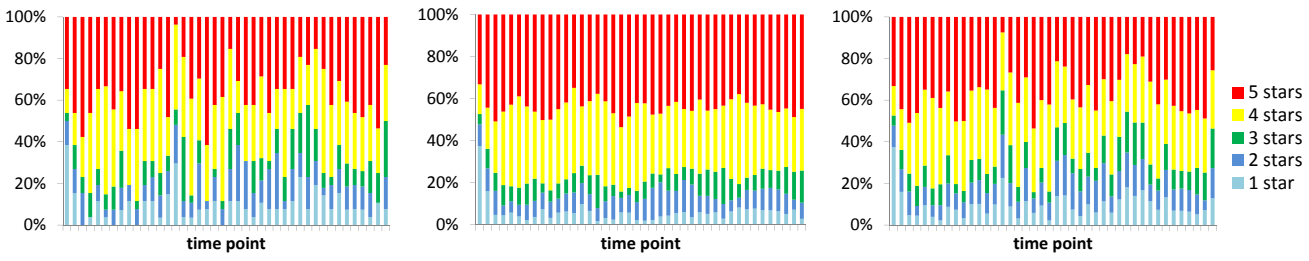

Figure 7: Left: Rating distributions over time for the hotel Punta Cana Princess (TripAdvisor data). Middle: Detected base behavior. Right: Reconstructed data after mixing with the anomaly signal.

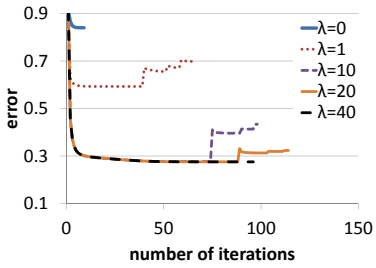

Figure 8: Convergence analysis

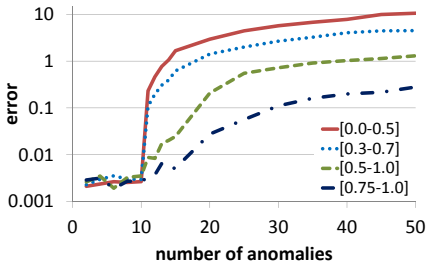

Figure 9: Robustness (here: $\lambda = 10$)

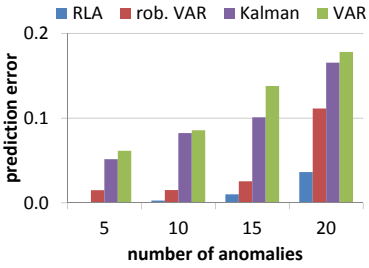

Figure 10: Prediction error

that only few iterations of the outer-loop are necessary to reach convergence. All variants converge in less than 120 iterations. In particular, the variant allowing no anomalies (λ=0) converges very quickly; though, of course, with a high error. As already shown in Fig. 6, increasing the value of λ lowers the obtained error since more flexibility is obtained.

In the next experiment, we analyze the robustness of our technique. For this experiment, we generated synthetic data with a varying number of anomalies (from 2-50). Additionally, we varied the strength of the anomalies by allowing the vector $\mathbf{p}$ to take only values from a specific domain: from very strong anomalies (p values between 0-0.5) to weak anomalies (p values from 0.75 to 1). For this experiment, we fixed the value of λ to 10, to show how our algorithm copes with data which is corrupted more than expected.

Figure 9 shows the result of this experiment. The x-axis shows the number of anomalies in the data, while the y-axis shows the reconstruction error of our method. As expected, if the number of anomalies in the data is less than 10, our method almost perfectly recovers the data. When exceeding this threshold, the error successively increases. The stronger the anomalies, the stronger the increase in the error. If the anomalies are weak, the errors can somehow be compensated by the base behavior.

Comparison with related techniques. In the following, we compare our RLA method against competing approaches. Due to the heterogeneity of the methods, we first describe the setup. As mentioned in Sec. 4, we compare against the (non-temporal) approaches PCA, ICA, and NNMF. To make these techniques applicable in our scenario – distinguishing between normal and irregular behavior –, we parametrize these methods to detect two components. For the temporal vector-autoregression approaches, we use an order of one, as our model does, too. The transition model and noise model of the Kalman Filter are learned via the EM algorithm.

We compare the methods based on their reconstruction error (i.e. how well these techniques approximate the data). For PCA, ICA, Kalman Filtering, and the non-robust variants of NNMF and vector autoregression, computing the approximated data is straightforward. Similarly, it is easy to compute the approximation using the result of our technique by simple mixing the base behavior with the anomaly signal. The robust NNMF and robust VAR, however, do not allow such an easy approach: Informally, these approaches remove points from the data when they are regarded as outliers, and the outliers do not need to follow a specific pattern (in contrast to our anomalies). Thus, given the results of these techniques one cannot simply reconstruct the original data. The principle we follow here is to replace the vectors (or entries) that have been regarded as outliers by these techniques with the original data. Thus, assuming for these points optimal reconstruction. Obviously, this gives a huge advantage to the competing methods.

Given the approximated data $\hat{\mathbf{X}}$, we compute the values $err = \|\hat{\mathbf{X}} - \mathbf{X}\|_F^2/T$, i.e. the approximation error per time stamp. This relative error allows us to compute the average over multiple time-series with varying length.

One advantage of our approach is the focus on convex combinations of valid distributions, thus, leading to only valid distributions for the approximated data (cf. Section 2 and 4). None of the competing methods considers this aspects. In many cases, we observed that the approximated vectors did not sum up to 1, thus, representing invalid distributions. An interpretation of such results is difficult or even misleading. Thus, in a second step, we additionally normalize the results of the competing methods to become valid distributions, leading to the normalized approximated data $\hat{\mathbf{X}}_{norm}$. Accordingly, we also compute the error $err_{norm} = \|\hat{\mathbf{X}}_{norm} - \mathbf{X}\|_F^2/T$.

		synth. data		real world data	
		err	err_{norm}	*err*	err_{norm}
temporal	RLA	0.0006	0.0006	0.0177	0.0177
	VAR	0.0098	0.0098	0.0368	0.0368
	rob. VAR	0.0060	0.0060	0.0330	0.0330
	Kalman	0.0091	0.0091	0.0273	0.0273
non-temp.	NNMF	0.0040	0.0044	0.0239	0.0252
	rob. NNMF	0.0027	0.0030	0.0213	0.0223
	PCA	0.0040	0.0044	0.0239	0.0252
	ICA	0.0102	0.0102	0.0242	0.0242

Table 1: Reconstruction error of the competing methods on synthetic and real world data.

Table 1 shows the results of all methods. In the left two columns ones sees the results on a synthetic dataset with 10 anomalies and 100 time points, the right two columns show the error averaged over multiple real world datasets. Considering the synthetic data one sees that RLA clearly outperforms the other temporal algorithms. In particular, the (non-robust) VAR and Kalman Filtering are sensitive to the anomalies in the data. Considering the non-temporal methods, ICA is not able to recover the hidden structure in the data. The error of the robust NNMF method is the smallest among the non-temporal approaches; though, as explained above, this method has an advantage when computing the error. Still, the error of this technique is larger than the one of RLA.

For real world data, we can infer similar results: Our RLA technique obtains the smallest error and, in particular, outperforms the other temporal techniques.

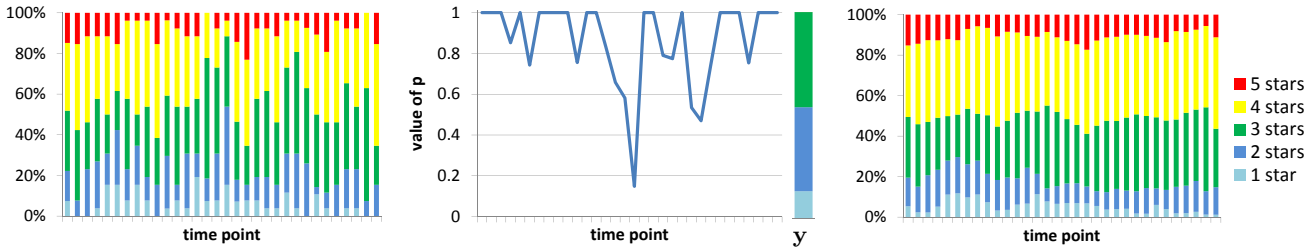

Figure 11: Left: Rating distributions over time for the Phoenix Airport (Yelp data). Middle: p vector and detected anomaly signal y. Right: Detected base behavior.

Comparison via prediction. The temporal techniques allow to predict the rating distribution at future points in time. In the following experiment, we analyze the methods' prediction accuracy. We generated synthetic data with 100 time points and an increasing number of anomalies, where we ensured that the rating distribution $\mathbf{x}$ at the last point in time is not an anomaly. We then removed $\mathbf{x}$ from the data, we applied the techniques on the remaining data, and we predicted the future rating distribution (e.g. in our model according to Equation 5). Similar as above, we compute the error $err_{norm} = \|\hat{\mathbf{x}}_{norm} - \mathbf{x}\|_F$, where $\hat{\mathbf{x}}_{norm}$ is the (normalized) predicted distribution.

Figure 10 shows the results. The non-robust VAR and Kalman Filtering show the highest error. In particular, for a high number of anomalies their error increases quickly since the hidden structure of the data can no longer be detected. RLA is more robust to the anomalies and also outperforms the robust VAR method. Obviously, for all methods a higher number of anomalies is more challenging when predicting the future rating distribution.

Discoveries. So far, we surveyed the efficiency and accuracy of our algorithm. In the following, we will demonstrate the application of our algorithm by illustrating some of our interesting discoveries from the three investigated real world datasets. Our approach for extracting products with noticeable rating irregularities relies on strong fluctuations in the $\mathbf{p}$ vector. As described in Section 2, small values in $\mathbf{p}$ weight the anomaly behavior strongly, thus, reflecting irregularities in the rating behavior. In contrast, the higher p_t the stronger the effect of the base user behavior at a certain point.

(1) As described in the introduction, one of our discoveries from the Amazon-Food data is the product "coconut-water"[6], presented in Figure 1. Figure 12 shows the values of the product's $\mathbf{p}$ vector at different timesteps t (x-axis). It is clear that during the timesteps $4-6$, the elements of $\mathbf{p}$ take small values which reflect the irregularities in the rating behavior. Additionally, Figure 12 shows the detected anomaly distribution $\mathbf{y}$. While the base behavior (Figure 1 (right)) shows the general trend of the product with primarily good ratings, the anomaly clearly explains the high amount of negative ratings observed during the time $4-6$. In particular, the anomaly shows a zero percentage of 5 star ratings. A closer look at the product's reviews during this time explains this behavior: Most of the customers complain about the *"new plastic bottle"* and *"a bad after taste"*. Since the manufacturer has recently changed his production from paper bottles to plastic bottles, many customers were disappointed. However, reviews after the anomalous time points hint to an improvement in the production (e.g., *"I can understand a lot of the initial bad reviews as I thought the new plastic bottle had a bad after taste. ... I can say that the taste is much improved ..."*).

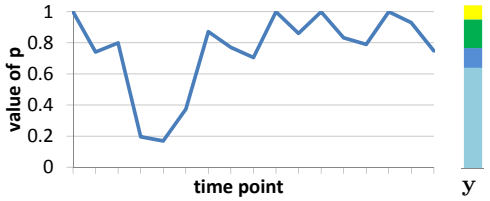

Figure 12: p vector and anomaly behavior y for the product "coconut-water" (Amazon data).

(2) Another interesting discovery of our algorithm from the Amazon-Food dataset is the detection of high anomalies in the rating of a dog food product[7]. The studied reviews at the detected time points show that the anomalies occurred due to the change of the country of manufacturing from the USA to another country: *"Originally, these were made in US. Now made in xxx ... advertised as natural, cage free, no hormones, no fillers, no antibiotics ...HUH? Not in xxx... I will not trust these as safe treats"*.

(3) Analyzing the dataset Yelp, an interesting pattern is found for the Sky Harbor Airport in Phoenix. Figure 11 (middle) shows the $\mathbf{p}$ vector for the rating behavior. Clearly, the elements of the $\mathbf{p}$ vector reach small values at the two intervals $[15, 18]$ and $[23, 25]$. Comparing this plot with the original rating data in Figure 11 (left), we are able to spot the sub-populations which were identified as irregularities by our method. Ignoring these irregularities, the base rating behavior looks as presented in Figure 11 (right). It is clear, that the rating behaves relatively robust with mostly 3-4 star ratings. The anomalous behavior, in contrast, shows a much higher fraction of 2 stars, and almost zero percentage of 4-5 stars. Mixing the base behavior with the anomalous signal using the $\mathbf{p}$ vector, our algorithm supplies an approximation to the real data with a very small residual error of $5.75 \cdot 10^{-4}$ per time stamp.

Analyzing the different reviews at yelp, it is noticeable that the main reasons for the high anomalies are, e.g., *"not many choices for food"* and *"long walkway between terminals"*. However, reviews shortly after the second anomaly note that *"... they are now doing some construction to try and fix things ..."*, giving hint to an improvement of the airport which may have caused the better evaluation (almost no 1 star ratings) at the end of the rating behavior.

(4) In Figure 7 we have already illustrated the distribution of ratings over time for the hotel Punta Cana Princess

[6] `http://www.amazon.com/dp/B000CNB4LE`

[7] `http://www.amazon.com/dp/B00141OU50`

evaluated on TripAdvisor. While manually detecting irregularities for this data might be difficult, examining the distribution of the p_t values (cf. Figure 13), clearly shows that there are 5 time intervals at which most of the anomalies are positioned. For these points the generally very good rating of the hotel (cf. Figure 7 (middle)), heavily drops. This behavior is reflected in the anomaly signal $\mathbf{y}$. Comparing the noticeable p_t values with the corresponding reviews, it is recognizable that the number of negative criticizes such as *"lack of communication"* and *"poor service and bad quality of food"* highly increases. Using such information in dependency of their time of occurrence can be used by the hotel's management to improve the quality of their service.

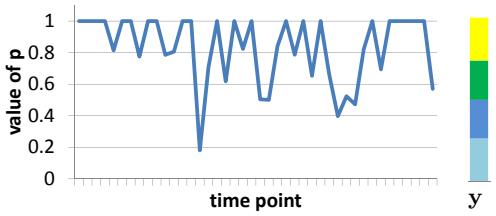

Figure 13: p vector and anomaly behavior y for hotel Punta Cana Princess (TripAdvisor).

Discoveries via prediction. Since our method exploits the temporal properties of the data, it allows us to predict future points in time. To show this potential of our algorithm for the scenario of anomaly detection, we deleted the rating distribution of the last time point from the restaurants from the yelp database. We then executed our algorithm on the remaining data, and we predicted the future rating distribution according to Eq. 5. Comparing the predicted ratings with the real (previously deleted) data, two products behaved noticeable among the others. These two products show huge difference to the predicted data, thus, very likely indicating anomalous behavior at the last point in time.

Figure 14 illustrates our observations. In each part of the diagram, the left distribution shows our prediction using the base behavior, while the right distribution presents the observed rating distribution. Clearly, both products show distributions which cannot be explained by the base behavior. The high increase in the number of low rating scores gives a strong indication for anomalous behavior.

Considering the first restaurant, the reviews at the last time point well explain the anomaly. Most of the reviewers comment ... *easily the worst bbq I've ever had...* or ...*I was so hoping to find a good barbecue restaurant near our house. Unfortunately, xxx BBQ just misses the mark...*, thus, indicating that the expectations of a previously high rated restaurant (cf. predicted rating distribution) have been missed. Even more noticeable are the reviews of the second restaurant, where most of the customers criticize the reduced quality using comments as ... *"What a come down! I've been to the xxx quite a few times and have always had a pretty good experience but not this last time ...*. This result gives high indication that the quality of the restaurant has *suddenly* dropped, e.g., due to a change of the chef.

With our method, we can spot these recently occurring anomalies, which can then be used to better inform the customers about the new characteristics of the product/service or to resolve the newly occurring problems from the salesman's perspective.

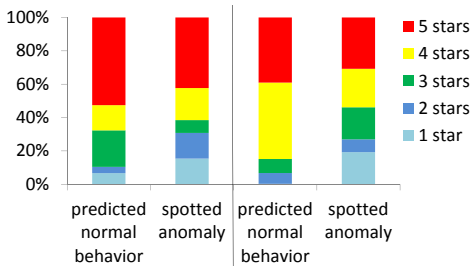

Figure 14: Spotting anomalies via prediction. Restaurants from the Yelp data.

6. CONCLUSION

We presented the RLA method for the temporal analysis of rating distributions. Our method can be used to detect the potentially evolving base behavior of users as well as to spot anomalous points in time where the rating distribution differs from this base behavior. Since our method exploits the temporal characteristics of the data, it further allows to predict the rating distributions of future points in time, which subsequently allows to detect newly occurring anomalies. The idea of our approach is to model the observed data as a mixture of a latent, autoregressive model representing the base behavior with a sparse anomaly signal. Based on this generative process, we derived an efficient algorithm which invokes a sequence of quadratic programs. In our experimental study, we demonstrated the strengths of our RLA method and we presented interesting discoveries found in rating data from Amazon, Yelp, and TripAdvisor.

As future work, we plan to extend our model to higher-order autoregressive processes and we want to investigate how to directly incorporate textual data in our method.

Acknowledgments.
Stephan Günnemann has been supported by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD). This material is based upon work supported by the National Science Foundation under Grant No. CNS-1314632. Research was also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Additional funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0088.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

7. REFERENCES

[1] C. C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[2] A. Agresti. *Analysis of ordinal categorical data*. Wiley, 2010.

[3] S. Baccianella, A. Esuli, and F. Sebastiani. Multi-facet rating of product reviews. In *Advances in Information Retrieval*, pages 461–472. Springer, 2009.

[4] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and

applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

[6] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[7] E. J. Candès, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[8] A. L. Chistov and D. Y. Grigor'ev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Mathematical Foundations of Computer Science 1984*, pages 17–31. Springer, 1984.

[9] C. Croux and K. Joossens. Robust estimation of the vector autoregressive model by a least trimmed squares procedure. In *COMPSTAT*, pages 489–501. Springer, 2008.

[10] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.

[11] M. Jamali, G. Haffari, and M. Ester. Modeling the temporal dynamics of social rating networks using bidirectional effects of social relations and rating patterns. In *WWW*, pages 527–536, 2011.

[12] M. Jamali, T. Huang, and M. Ester. A generalized stochastic block model for recommendation in social rating networks. In *RecSys*, pages 53–60, 2011.

[13] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[14] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *SDM*, pages 389–400, 2009.

[15] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[16] L. I. Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *IEEE Trans. Knowl. Data Eng.*, 25(5):1175–1180, 2013.

[17] K. Lerman. Dynamics of a collaborative rating system. In *WebKDD/SNA-KDD*, pages 77–96, 2007.

[18] R. B. Litterman. Forecasting with bayesian vector autoregressions. *Journal of Business & Economic Statistics*, 4(1):25–38, 1986.

[19] Y. Liu, X. Yu, A. An, and X. Huang. Riding the tide of sentiment change: sentiment analysis with evolving online reviews. *World Wide Web*, pages 1–20, 2013.

[20] H. Lütkepohl. *New introduction to multiple time series analysis*. Cambridge University Press, 2005.

[21] T. C. Mills. *Time series techniques for economists*. Cambridge University Press, 1991.

[22] S.-H. Min and I. Han. Detection of the customer time-variant pattern for improving recommender systems. *Expert Systems with Applications*, 28(2):189–199, 2005.

[23] S. Mukherjee, G. Basu, and S. Joshi. Incorporating author preference in sentiment rating prediction of reviews. In *WWW*, pages 47–48, 2013.

[24] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[25] J.-A. Ting, E. Theodorou, and S. Schaal. Learning an outlier-robust kalman filter. In *ECML*, pages 748–756, 2007.

[26] L. Xiong, X. Chen, and J. G. Schneider. Direct robust matrix factorizatoin for anomaly detection. In *ICDM*, pages 844–853, 2011.

APPENDIX

PROOF OF THEOREM 1. Let $\mathbf{p}$ and w be given. Rewriting the objective function $f(...)$, it becomes apparent that we only have to deal with (a) a constant independent of the variables $\mathbf{a}^{(t)}, \mathbf{b}, \mathbf{y}$, which, thus, does not affect their optimal values, and (b) terms $c \cdot u_i$, $c \cdot u_i^2$, and $c \cdot u_i \cdot u_j$, where u_i and u_j are one of the variables $a_d^{(t)}$, b_d, or y_d, and c is a constant depending on the choice which variables are considered. Carefully isolating these constants, leads to the vector $\mathbf{c}^\bullet \in \mathbb{R}^{(T+3)\cdot D}$ with

$$c_i^\bullet = \begin{cases} -2 \cdot x_d^{(t)} \cdot p_t & \text{if } D < i \leq (T+1) \cdot D \\ \sum_{t=1}^{T} 2 \cdot x_d^{(t)} \cdot (p_t - 1) & \text{if } (T+2) \cdot D < i \leq (T+3) \cdot D \\ 0 & \text{else} \end{cases}$$

where we used $t := \left\lceil \frac{i}{D} \right\rceil - 1$ and $d := 1 + [(i-1) \bmod D]$, and to the matrix $\mathbf{Q}^\bullet$ whose diagonal entries are

$$Q_{i,i}^\bullet = \begin{cases} 2 \cdot w^2 & \text{if } 1 \leq i \leq D \\ 2 \cdot p_t^2 + 2 + 2 \cdot w^2 & \text{if } D < i \leq T \cdot D \\ 2 \cdot p_t^2 + 2 & \text{if } T \cdot D < i \leq (T+1) \cdot D \\ 2 \cdot T \cdot (1-w)^2 & \text{if } (T+1) \cdot D < i \leq (T+2) \cdot D \\ \sum_{t=1}^{T} 2 \cdot D \cdot (1-p_t)^2 & \text{if } (T+2) \cdot D < i \leq (T+3) \cdot D \end{cases}$$

and off-diagonal entries $Q_{j,i}^\bullet = Q_{i,j}^\bullet$ (with $i < j$) and

$$Q_{i,j}^\bullet = \begin{cases} -2 \cdot w & \text{if } 1 \leq i \leq T \cdot D \wedge j = i + D \\ -2 \cdot w^2 + 2w & \text{if } 1 \leq i \leq D \wedge j = d + (T+1) \cdot D \\ -2 \cdot (1-w)^2 & \text{if } D < i \leq (T+1) \cdot D \wedge j = d + (T+1) \cdot D \\ 2 \cdot (p_t - p_t^2) & \text{if } D < i \leq (T+1) \cdot D \wedge j = d + (T+2) \cdot D \\ 0 & \text{else} \end{cases}$$

The matrix $\mathbf{Q}^\bullet$ is sparse because only $7 \cdot T \cdot D + 5 \cdot D$ elements are not zero. Since the set $\mathcal{C}_D$ used in problem P1, adds only linear constraints to the variables $\mathbf{a}^{(t)}, \mathbf{b}, \mathbf{y}$, it is easy to derive matrices $\mathbf{A}^\bullet$, $\mathbf{d}^\bullet$ describing these constraints. □

PROOF OF THEOREM 2. Rewriting the objective function shows that it only involves terms $c \cdot v_i$ and $c \cdot v_i^2$, where v_i is one of the variables p_t or w, and c is a constant depending on the choice which variable is considered. Isolating these constants, leads to the vector $\mathbf{c}^\circ$ with

$$c_i^\circ = \begin{cases} \sum_{d=1}^{D} -(a_d^{(i)} - y_d)(x_d^{(i)} - y_d) & \text{if } 1 \leq i \leq T \\ \sum_{t=1}^{T} \sum_{d=1}^{D} -(a_d^{(t)} - b_d)(a_d^{(t-1)} - b_d) & \text{if } i = T+1 \end{cases}$$

and to the diagonal matrix $\mathbf{Q}^\circ = diag(q_1^\circ, \ldots, q_{T+1}^\circ)$ with

$$q_i^\circ = \begin{cases} \sum_{d=1}^{D} 2 \cdot (a_d^{(i)} - y_d)^2 & \text{if } 1 \leq i \leq T \\ \sum_{t=1}^{T} \sum_{d=1}^{D} 2 \cdot (a_d^{(t-1)} - b_d)^2 & \text{if } i = T+1 \end{cases}$$

Obviously, $\mathbf{Q}^\circ$ is a sparse matrix. Since the L_0 norm used in the original problem definition has been replaced with the linear constraint of Equation 9, it is again easy to derive matrices $\mathbf{A}^\circ$, $\mathbf{d}^\circ$ describing these constraints. □

3.2 Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution

Peer-reviewed Conference Paper

Autors Stephan Günnemann, **Nikou Günnemann**, Christos Faloutsos

In *proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 841-850, 2014, [25].*

Abstract Rating data is ubiquitous on websites such as Amazon, Trip- Advisor, or Yelp. Since ratings are not static but given at various points in time, a temporal analysis of rating data provides deeper insights into the evolution of a product's quality. In this work, we tackle the following question: Given the time stamped rating data for a product or service, how can we detect the general rating behavior of users as well as time intervals where the ratings behave anomalous?

We propose a Bayesian model that represents the rating data as sequence of categorical mixture models. In contrast to our previous method, our new principle does not require any aggregation of the input but it operates on the original time stamped data. To capture the dynamic effects of the ratings, the categorical mixtures are temporally constrained: Anomalies can occur in specific time intervals only and the general rating behavior should evolve smoothly over time. Our method automatically determines the intervals where anomalies occur, and it captures the temporal effects of the general behavior by using a state space model on the natural parameters of the categorical distributions.

For learning our model, we propose an efficient algorithm combining principles from variational inference and dynamic programming. Overall, this ensures a runtime linear in the number of overserved time points. In our experimental study we show the effectiveness of our method and we present interesting discoveries on multiple real world datasets.

Contribution of thesis author: Algorithmic discussions and implementation, majority of experimental analysis as well as partly manuscript writing and revision.

Detecting Anomalies in Dynamic Rating Data: A Robust Probabilistic Model for Rating Evolution

Stephan Günnemann Nikou Günnemann Christos Faloutsos
Carnegie Mellon University, USA
{sguennem, nguennem, christos}@cs.cmu.edu

ABSTRACT

Rating data is ubiquitous on websites such as Amazon, TripAdvisor, or Yelp. Since ratings are not static but given at various points in time, a temporal analysis of rating data provides deeper insights into the evolution of a product's quality. In this work, we tackle the following question: Given the time stamped rating data for a product or service, how can we detect the general rating behavior of users as well as time intervals where the ratings behave anomalous?

We propose a Bayesian model that represents the rating data as sequence of categorical mixture models. In contrast to existing methods, our method does not require any aggregation of the input but it operates on the original time stamped data. To capture the dynamic effects of the ratings, the categorical mixtures are temporally constrained: Anomalies can occur in specific time intervals only and the general rating behavior should evolve smoothly over time. Our method automatically determines the intervals where anomalies occur, and it captures the temporal effects of the general behavior by using a state space model on the natural parameters of the categorical distributions. For learning our model, we propose an efficient algorithm combining principles from variational inference and dynamic programming. In our experimental study we show the effectiveness of our method and we present interesting discoveries on multiple real world datasets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications —*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

robust mining; anomaly detection; categorical mixtures

1. INTRODUCTION

Online rating data provides customers valuable information about products and services and supports their decision making process. Exploiting and presenting this data is a key feature of websites such as Amazon, Yelp, or Tripadvisor. Besides the usefulness of rating data for customers, also companies and manufactures can benefit from it by, e.g., using the data to detect functional weaknesses of their products or changes in the customers' satisfaction.

In this work, we propose a method for analyzing rating data that incorporates the data's temporal characteristics. Given the time stamped rating data for a product or service, we aim at detecting the general rating behavior of the users (called the base behavior) as well as time intervals in which these ratings deviate from the general population (anomalous behavior). The base behavior describes the general quality of a product or service accounting for temporal evolutions, e.g., resulting from decreasing quality due to technical progress of competing products. The anomalies, in contrast, deviate from the base behavior and might, e.g., occur due to spammers trying to push the success of a product or due to changes in the service quality.

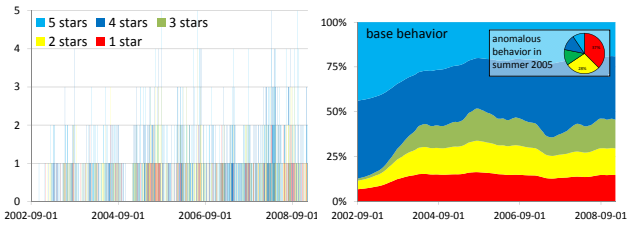

Fig. 1: Left: Time-stamped rating data analyzed by our method (here: a hotel from TripAdvisor). Right: Extracted base behavior. Upper corner: Anomalous behavior detected in summer 2005.

In Figure 1, we show a real world example for such an effect. The data we analyzed here is a hotel from the TripAdvisor database. On the left, we show a subset of the original time stamped data. The colors indicate the different star ratings, and the height of each bar the number of these ratings at the current time. Obviously, it is hard to analyze such data by hand. In particular, keep in mind that the ratings are not uniformly distributed over time.

On the right, we illustrate the detected *base behavior* of our method. As one can see, the base behavior nicely shows the general rating behavior of the users and evolves smoothly over time with primarily medium to high ratings. Additionally, our method has found anomalous behavior in the months of July and August 2005. As shown on the upper right, in these intervals, the fraction of low ratings (red and yellow) is highly increased compared to the base behavior (65% low ratings compared to around 30% in the base behavior). As we will show in Section 5, these anomalies occurred due to problems in the service of the hotel.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623721.

In general, our method detects time intervals in dynamic rating data which show anomalous behavior and – at the same time – it detects the base behavior if the data would *not* be polluted by anomalies. Besides using this principle to detect weaknesses in products and services, it can generally be used to filter out rating information which behave anomalous. Thereby, prospective customers might be provided with a cleaned history about the product; or, one might specifically highlight these time intervals to the users to provide the whole picture on a product (since otherwise these anomalous ratings are hidden in the larger set of normal behavior). As an additional benefit, we can exploit our method to predict the base behavior of future ratings. Accordingly, when new ratings arrive, we can estimate whether they match or deviate from the predicted behavior – thus, giving indication of new anomalies.

So far, there exists only a single method which analyzes temporal rating data under the presence of anomalies [7]. A potential drawback of [7], however, is the necessary aggregation/binning of the data. When using a coarse aggregation, the temporal effects of the data are not well captured (in the extreme, all data is a single bin). When using a fine aggregation, the analyzed distributions might degenerate (in the extreme, many bins are empty). In our model we avoid this problem by directly operating on the time stamped data which is modeled via a sequence of categorical mixture models. We explicitly keep into account that ratings might not uniformly arrive over time. Furthermore, the work [7] assumes that anomalies occur at individual points in time. Our work captures the effects of real data much better by accounting for multiple different types of anomalies appearing across several *time intervals*. Our contributions are:

- *Mining task:* We present a technique for the analysis of time stamped rating data. Our method detects the base behavior of users as well as time intervals where potential anomalies occurred. Additionally, our technique can be used to predict the rating behavior at future time points.
- *Theoretical soundness:* Our method is based on a sound Bayesian model that represents the rating data as a sequence of temporally constrained categorical mixture models. To capture the temporal effects of the base behavior we use a state space model on the natural parameters of the categorical distributions.
- *Algorithm design:* We propose an efficient algorithm for learning our model which combines principles from variational inference and dynamic programming.
- *Effectiveness:* We evaluate our method on different real world datasets and we show its effectiveness by presenting interesting findings.

2. BAYESIAN FRAMEWORK

In this section, we introduce our model for detecting the base rating behavior of users as well as time intervals in which anomalies have been occurred. Following convention, we do not distinguish between a random variable X and its realization $X = x$ if it is clear from the context. As an abbreviation, we denote sets of random variables with the index $*$, e.g. $z_*^{(t)}$ is the set $\{z_i^{(t)}\}$ with i in the corresponding domain, and z is an abbreviation for the set $z_*^{(*)}$. Vectors of (random) variables are written in bold font, e.g. $\boldsymbol{b}$, while the entries of the vectors are given in standard font, e.g. b_i. We write $t \in T$, as a shortcut for $t \in \{1, \ldots, T\}$.

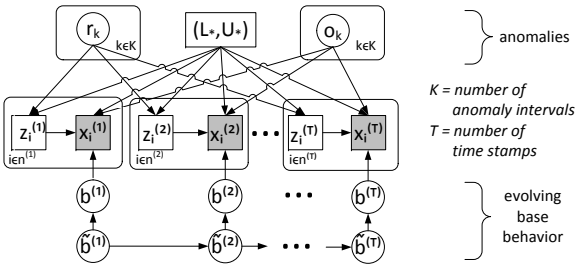

Fig. 2: Graphical model of our approach

Preliminaries. The data we consider is a time stamped collection of ratings. Let $x_i^{(t)}$ denote the i-th rating occurred at *time index* t, and $s^{(t)}$ the *time stamp* at time index t. At each time index we might observe multiple ratings (e.g. if time stamps are only measured/provided on a daily basis). We denote with $n^{(t)}$ the number of ratings at time index t. We assume the data to be ordered according to time, i.e. $s^{(t)}$ occurs after $s^{(t-1)}$. We denote with $\Delta^{(t_1,t_2)}$ the elapsed time between time stamp $s^{(t_1)}$ and $s^{(t_2)}$ and we set $\Delta^{(t)} := \Delta^{(t-1,t)}$. Note that for each t a different $\Delta^{(t)}$ might be observed since we do not require a fixed binning or aggregation of the rating data. We denote with T the number of time indices (i.e. the number of *distinct* time stamps) and we assume that users can choose ratings based on a rating scale with S different ratings (e.g. stars from 1 to S). As an abbreviation for later use, we define $n_s^{(t)} := |\{i \in \{1, \ldots, n^{(t)}\} \mid x_i^{(t)} = s\}|$ to be the number of ratings at time t which possess the evaluation $s \in S$.

Generative Process.

We model the rating data including potential anomalies via a probabilistic generative process. An overview of our generative process showing the used variables and their dependencies is illustrated by the graphical model in Figure 2. In the following we discuss the details of this process.

Given the observed rating data X, our aim is to extract the base behavior of the users and intervals in time where anomalies occur. Since the observed data might already be polluted by anomalies, we cannot directly use it to estimate the base behavior. Instead, we assume that the observed data is obtained by mixing the (unkown) base user behavior with the (unknown) anomaly behavior. Thus, both types of behavior are represented as *latent* variables which are not directly observed but inferred by our technique. Technically, at each point in time we have a categorical mixture model as illustrated in Figure 3. To incorporate the temporal properties of the data, we perform additional modeling:

Part 1: Mixing anomalies and base behavior. In contrast to the base behavior, which is present over the whole timespan, we assume that anomalies are rare events (otherwise, they would correspond to the majority of the data, making the term "anomaly" rather meaningless) and occur only in a specific time interval like, e.g., during a short attack of spam. Technically, instead of using an individual anomaly at each point in time, we restrict the number of anomalies to be small, i.e. smaller than a number K (we discuss later how to choose this parameter), and we temporally constrain the "influence" of each anomaly to a small time interval. For each anomaly, we define this interval by the random variables L_k and U_k, denoting the lower and upper bound of time indices at which the kth anomaly occurs.

In the following, we will use the function

$$k(t) = \begin{cases} k & \text{if } \exists x : x = k \wedge L_x \leq t \leq U_x \\ 0 & \text{else} \end{cases} \quad (1)$$

which maps the time index t to the corresponding anomaly (or to 0 if no anomaly exists at this point in time). Here, we require disjointness of the different intervals.

Note that the use of anomaly *intervals* is a huge advantage in contrast to [7], which models the anomalies at each time point individually. The potential of this extension is best shown for the case of very fine grained time stamps: In this case, we usually expect only a single rating per time stamp, i.e. $n^{(t)} = 1$. To capture anomalies at multiple consecutive points in time, the work of [7] has to use multiple anomalies, while in our work a single interval suffices.

At this point we want to mention the difference between outliers and anomalies [7]: While *outliers* are irregular behavior attributed to mostly random corruptions of the data (like, e.g., measurement errors), *anomalies* are irregular behavior that follow a specific pattern (like, e.g., time points with consistently low ratings due to a change in the product's quality). In our work, we consider anomalous behavior.

Giving the above information, the observed data at time t is modeled as a categorical mixture model defined by

$$x_i^{(t)} \sim \begin{cases} \text{Categorical}(\boldsymbol{\pi}^{(t)}) & \text{if } z_i^{(t)} = 0 \\ \text{Categorical}(\mathbf{o}_{k(t)}) & \text{if } z_i^{(t)} = 1 \end{cases} \quad (2)$$

$$z_i^{(t)} \sim \begin{cases} 0 & \text{if } k(t) = 0 \\ \text{Bernoulli}(r_{k(t)}) & \text{else} \end{cases} \quad (3)$$

Here, $z_i^{(t)}$ is the indicator variable showing which ratings are anomalies. $\boldsymbol{\pi}^{(t)} \in [0...1]^S$ is the vector describing the base behavior at time t, $\mathbf{o}_k \in [0...1]^S$ is the kth anomaly behavior, and r_k is the mixing proportion. The higher the value of r_k, the higher the proportion of anomalies within the corresponding interval. If no anomaly is present at time t, all variables $z_*^{(t)}$ are set to zero, corresponding of using only the base behavior at this point in time. Thus, the mixture model's components referring to the anomalies are constrained to specific intervals (cf. Figure 3).

For a Bayesian treatment, we equip the variables with corresponding prior distributions. We use

$$\mathbf{o}_k \sim Dir(\hat{\boldsymbol{\alpha}}) \qquad r_k \sim Beta(\hat{\alpha}, \hat{\beta})$$

due to the properties of conjugacy. The vector $\hat{\boldsymbol{\alpha}}$, for example, can be used to specify prior knowledge about potential anomalies (e.g. anomalies should represent primarily low ratings). In all of our experiments we use $\hat{\boldsymbol{\alpha}} = \mathbf{1}$ and $\hat{\alpha} = \hat{\beta} = 1$ corresponding to non-informative priors.

For the lower and upper bounds we exploit the idea that anomalies should appear primarily in short time intervals. That is, we assume that the probability to observe an anomaly over a very long time frame is lower than the probability to observe anomalies over only a few time points. We capture this effect by drawing the lower and upper bounds according to an exponential distribution controlled by the duration $\Delta^{(L_k,U_k)}$ of the anomaly interval. Formally

$$p(L_*, U_*) \propto \begin{cases} \prod_{k=1}^{K} e^{-\lambda \cdot \Delta^{(L_k,U_k)}} & \text{if disjoint intervals} \\ 0 & \text{else} \end{cases} \quad (4)$$

Please note that this is the joint distribution over all intervals since we require their disjointness. The larger λ, the larger the bias to small anomaly intervals. Per default, if no prior knowledge is given, one should select $\lambda = 0$. In this case, any combination of L_*, U_* is equally likely, corresponding to a non-informative prior. Note also that $\lambda = 0$ is a *valid* prior since the domain of L_*, U_* is finite.

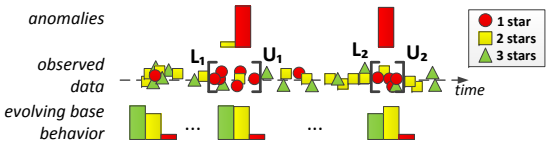

Fig. 3: Illustration of the generative process using temporally constrained categorical mixture models

Part 2: Smoothness of the base behavior. So far, we have not specified any distribution on the base behavior $\boldsymbol{\pi}^{(t)}$. The core idea is that the base behavior should evolve smoothly over time according to the general behavior of the users. That is, we want to incorporate the temporal properties of the data.

As pointed out in [6], the (mean) parameters of the categorical distribution and their corresponding Dirichlet hyperparameters are not amenable to sequential modeling. Therefore, we exploit a similar idea as proposed in [6, 17]: instead of operating on the (mean) parameters $\boldsymbol{\pi}^{(t)}$, we operate on the natural parameters (cf. exponential family [5]). For the categorical distribution, the natural parameters are simply given by the logs of the mean values, i.e. $b_s^{(t)} = log(\pi_s^{(t)}/\pi_S^{(t)})$. While the mean parameters are restricted to lie on the simplex, the natural parameters are unconstrained, leading to an elegant way of sequential modeling.

In the following, we only operate on the natural parameters $\mathbf{b}^{(t)}$. If the mean parameters are required (e.g. as in Equation 2), we can simply apply the transformation

$$\pi_s^{(t)} = \frac{exp(b_s^{(t)})}{\sum_{j \in S} exp(b_j^{(t)})} =: \pi(\mathbf{b}^{(t)})_s$$

Note that the term $b_S^{(t)}$ is always 0. Therefore, we can ignore it for the remainder of the discussion, thus, operating effectively on an $S-1$ dimensional space.

Given the natural parameters $\mathbf{b}^{(t)}$ at each time index $t \in T$, we model their smoothness by exploiting the idea of linear state space systems [5] in combination with Brownian motion [11, 17]. First, we assume an underlying state space $\widetilde{\mathbf{b}}^{(t)}$ which temporally evolves over time via

$$\widetilde{\mathbf{b}}^{(t)} \sim \mathcal{N}(\widetilde{\mathbf{b}}^{(t-1)}, \Delta^{(t)} \cdot \mathbf{Q}) \quad (5)$$
$$\widetilde{\mathbf{b}}^{(1)} \sim \mathcal{N}(\widetilde{\mathbf{b}}_0, \mathbf{Q}_0)$$

We call this space the "smoothed" base behavior. Intuitively, the state of the smoothed base behavior at time t corresponds to the old state plus a small deviation government by the noise covariance matrix $\Delta^{(t)} \cdot \mathbf{Q}$. The larger the time difference between two observed ratings, the larger the corresponding covariance. That is, we effectively allow a higher change in the base behavior if the elapsed time between two ratings is high. If time points are very close to each other, we allow only small changes in the base behavior. In the limit, this discrete-time Gaussian random walk corresponds to Brownian motion [11, 17].

This process captures naturally the effects of rating data. In the case of movies, for example, one might see many rat-

ings appearing in short time frames during the time the movie has been released to the theaters and again many ratings a few month later when the DVD has been released. Both time frames potentially describe different base behavior due to different audiences.

Given the smoothed base behavior, the actual base behavior is now obtained by the simple random process

$$\mathbf{b}^{(t)} \sim \mathcal{N}(\widetilde{\mathbf{b}}^{(t)}, \mathbf{R}) \tag{6}$$

which again allows a small deviation between the base behavior and its smoothed counterpart. Note that we do not directly impose the temporal evolution between the variables $\mathbf{b}^{(*)}$, but via the state space $\widetilde{\mathbf{b}}^{(*)}$. This additional layer is in particular beneficial if the number of ratings varies strongly between the time points. If we would not use $\widetilde{\mathbf{b}}^{(*)}$, a single time point with a huge amount of ratings could dominate most of the temporal behavior.

Finally, we add corresponding priors to the newly introduced parameters. By exploiting the fact of conjugacy it follows that $\mathbf{Q}$ is drawn from an Inverse-Wishart distribution, i.e. $\mathbf{Q} \sim \mathcal{W}^{-1}(\Psi_q^0, \nu_q^0)$. The parameters Ψ_q^0 and ν_q^0 can be used to control the smoothness of the base behavior by providing prior knowledge about the noise covariance. Similarly, $\mathbf{R}$ follows an Inverse-Wishart distribution and $(\widetilde{\mathbf{b}}_0, \mathbf{Q}_0)$ a Normal-Inverse-Wishart distribution.

Summary and Discussion.

Overall, our generative process captures the temporal properties of the data by modeling a smooth base behavior as well as accounting for anomalies which are constrained to occur at certain time intervals. We will show in Section 3 how we perform efficient (approximate) inference for this model.

Model Selection. So far, we assumed the number K of anomalies is given. If not apriori known, we can estimate it via model selection. We choose the Bayesian information criterion [5]. Any other criterion can be used as well.

As we will see in Sec. 3, we will integrate out all latent variables except of $\Theta = \{L_*, U_*, \boldsymbol{Q}, \boldsymbol{R}\}$. Thus, increasing the value of K by one, increases the number of free parameters in our model by about 2 (the lower and upper bound of the new anomaly interval). This is a slight overestimate since the intervals need to be disjoint and, thus, they are not completely independently free variables. Given this result, we can choose the K minimizing the BIC equation

$$BIC(K) = -2 \cdot \ln \mathcal{L}_K + (2 \cdot K + c) \cdot \ln(\sum_t n^{(t)})$$

Here, the constant c denotes the parameters of the model which do not increase when increasing K. Since the value of c does not affect the optimal choice for K, we can simply set it to 0. The term $\mathcal{L}_K$ denotes the likelihood of the data when using K anomaly intervals. We can approximate it with the technique shown in Section 3.

Prediction. Since the base behavior evolves via a linear state space system, we are able to predict the behavior at future points in time. Combining Eq. 5 and 6, it follows

$$\mathbf{b}^{(T+1)} \sim \mathcal{N}(\widetilde{\mathbf{b}}^{(T)}, \mathbf{R} + \Delta^{(T+1)} \cdot \mathbf{Q}) \tag{7}$$

Thus, given estimates for $\widetilde{\mathbf{b}}^{(T)}$, $\mathbf{R}$, and $\mathbf{Q}$ (cf. Section 3), comparing the observed ratings at time $T+1$ against the predicted base behavior can be used as an indicator whether new anomalies have been occurred.

3. ALGORITHM

While the previous section focused on the model's generative process, we now present our learning technique. That is, *given* a set of observations X we aim at inferring the values of the hidden variables which best describe the observed data. There are multiple ways to formulate this objective. In this work, we treat the variables $\Theta = \{L_*, U_*, \boldsymbol{Q}, \boldsymbol{R}\}$ as parameters and we are interested in finding their maximum a posteriori estimate Θ_{MAP} as well as the posterior distribution $p(V|X, \Theta_{MAP})$ of the latent variables $V = \{\mathbf{o}_*, r_*, z_*^{(*)}, \mathbf{b}^{(*)}, \widetilde{\mathbf{b}}^{(*)}\}$ (which can then, e.g., be used to pick specific realizations of the latent variables).

3.1 Variational EM

Since exact inference in our model is intractable, we compute an approximation using variational expectation-maximization [5]. The idea is to approximate $p(V|X, \Theta)$ by a tractable family of parametrized distributions $q(V|\Omega)$. The parameters Ω are the free variational parameters. These parameters are optimized such that the best approximation between q and p is obtained. This corresponds to the expectation step of the variational EM method. Technically, we minimize the Kullback-Leibler divergence between q and p by optimizing over Ω. Using Jensen's inequality, minimizing the KL divergence is equivalent to maximizing the following lower bound on the log marginal likelihood [5]:

$$\mathcal{L}(X; \Theta, \Omega) = \mathbb{E}_q[\ln p(X, V, \Theta)] + H(q) \tag{8}$$

where $\mathbb{E}_q[.]$ denotes the expectation w.r.t. the q distribution and H the entropy.

Given an approximation of $p(V|X, \Theta)$ via $q(V|\Omega)$, we then determine updated parameter values for Θ by again maximizing Equation 8. This corresponds to the maximization step of the EM method.[1]

In short, the general processing scheme of our method is to alternatingly maximize $\mathcal{L}(X; \Theta, \Omega)$ w.r.t. Ω and Θ. *As we will later see, we actually interweave both steps by simultaneously optimizing parts of* Θ *and* Ω.

3.1.1 Variational distribution

In contrast to the frequently used mean field approximation, which assumes a fully factorized distribution, we use

$$p(V \mid X, \Theta) \approx q(V|\Omega) := \prod_k q_1(\mathbf{o}_k) \cdot \prod_k q_2(r_k) \cdot \prod_t \prod_i q_3(z_i^{(t)}) \cdot \prod_t q_4(\mathbf{b}^{(t)}) \cdot q_5(\widetilde{\mathbf{b}}^{(1)}) \cdot \prod_{t>1} q_5(\widetilde{\mathbf{b}}^{(t)} \mid \widetilde{\mathbf{b}}^{(t-1)})$$

We retain the sequential structure of the smoothed base behavior in q_5. Indeed, as described later, we determine q_5 via a Kalman filter where it follows that $q_5(\widetilde{\mathbf{b}}^{(t)}|\widetilde{\mathbf{b}}^{(t-1)})$ is a Normal distribution given by $\mathcal{N}(\widetilde{\mathbf{b}}^{(t)} \mid \widetilde{\boldsymbol{\mu}}_{t|T}, \mathbf{P}_{t|T})$. For the remaining variational distributions we use

$$q_1(\mathbf{o}_k) = Dir(\mathbf{o}_k|\boldsymbol{\alpha}_k) \qquad q_3(z_i^{(t)}) = Bernoulli(z_i^{(t)}|\phi_{t,i})$$
$$q_2(r_k) = Beta(r_k|\alpha_k, \beta_k) \quad q_4(\mathbf{b}^{(t)}) = \mathcal{N}(\mathbf{b}^{(t)}|\boldsymbol{\mu}^{(t)}, v^{(t)} \cdot I)$$

where $\Omega = \{\boldsymbol{\alpha}_*, \alpha_*, \beta_*, \phi_{*,*}, \boldsymbol{\mu}^{(*)}, v^{(*)}\}$ are the variational parameters to be optimized.

[1] The only actual difference between these steps is that Θ represents a point estimate of the random variables, while Ω represents the hyperparameters of a full distribution.

Note that the distributions $q_3(z_i^{(t)})$ and $q_3(z_{i'}^{(t)})$ are identical when $x_i^{(t)} = x_{i'}^{(t)}$, i.e. when both ratings have the same value. Thus, in practice we do not need to keep track of $n^{(t)}$ many different distributions at time t but it is sufficient to record S many distributions; one for each possible evaluation. We denote with ϕ_t^s the variational parameter of the distribtion q_3 for all ratings showing evaluation s at time t.

3.1.2 Optimization Procedures

As described above, our goal is to update the values of Ω and Θ by maximizing (or more generally increasing) the value of Equation 8. One crucial requirement of our technique was to ensure the efficiency of our method. In the following, we want to highlight the most important results.

3.2 Optimizing the Lower/Upper Bounds

A first naive solution to update the lower/upper bounds of the anomaly intervals would be to test any possible combination. Obviously, this solution is not efficient and requires time $O(T^2)$ already for a single anomaly. We provide a principle which is *linear* in the number of time stamps.

We start with the case of a single anomaly and uniform gaps between all time stamps, i.e. it holds K=1 and $\Delta^{(t)}$=1 for all t.

3.2.1 Simultaneous Optimization

Equation 3 shows the dependency between L_1/U_1 and z. Intuitively, the bounds act as a switch on the distribution of z: if z is outside of the interval, it is the trivial 0 distribution; if it is inside, it is a Bernoulli. Accordingly, assuming the posterior distribution for z (or its approximation q_3) is *given*, an optimization of L_1/U_1 is rather meaningless since one trivially has to capture all time points where the distribution is not the constant 0. Therefore, we propose to *simultaneously* optimize L_1/U_1 and q_3 to maximize Equation 8.

Observation: If we know that a time point t fulfills $t \in [L_k, U_k]$, the optimal distribution of $q_3(z_i^{(t)})$ can be computed independent from all other points in time. The optimal distribution is obtained by setting its variational parameter $\phi_{t,i}/\phi_t^s$ to the value as derived in Sec. 3.3. In particular, this value is independent of the actual values of L_k and U_k (knowing that $t \in [L_k, U_k]$). Based on this result we can also compute the entropy

$$h_{t,s} := H(q_3(z_i^{(t)})) = -\phi_t^s \ln \phi_t^s - (1-\phi_t^s)\ln(1-\phi_t^s)$$

for all $z_i^{(t)}$ fulfilling $x_i^{(t)}$=s. If $t \notin [L_k, U_k]$, we have $q_3(z_i^{(t)} = 0) = 1$ and we define the entropy $H(q_3)$ to be zero.

Using these results and the derivations of the appendix, as well as removing all terms which are independent of L_1, U_1 and q_3, we can reformulate Equation 8 to:

$$\begin{aligned}
&\ln p(L_*, U_*) + \sum_{t\in T}\sum_{i\in n^{(t)}} \mathbb{E}_q[\ln p(z_i^{(t)}|...) + \ln p(x_i^{(t)}|...)] + H(q_3(z_i^{(t)})) \\
&= -\lambda\cdot\Delta^{(L_1,U_1)} + \sum_{t\notin[L_1,U_1]}\sum_{s=1}^{S} n_s^{(t)}\cdot\mathbb{E}_q\left[\ln\pi(\mathbf{b}^{(t)})_s\right] + \sum_{t\in[L_1,U_1]} \Big(\\
&\quad \sum_{s=1}^{S} [n_s^{(t)}\cdot h_{t,s} + n_s^{(t)}\cdot\phi_t^s\cdot(\mathbb{E}_q[\ln r_{k(t)}] + \mathbb{E}_q[\ln o_{k(t),s}]) \\
&\qquad + n_s^{(t)}\cdot[1-\phi_t^s]\cdot(\mathbb{E}_q[\ln(1-r_{k(t)})] + \mathbb{E}_q[\ln\pi(\mathbf{b}^{(t)})_s])]\Big) \\
&= \lambda + \sum_{t\in T}\sum_{s=1}^{S} n_s^{(t)}\cdot\mathbb{E}_q\left[\ln\pi(\mathbf{b}^{(t)})_s\right] + \sum_{t\in[L_1,U_1]} f_1(t) \qquad (9)
\end{aligned}$$

where we used the fact $\Delta^{(L_k,U_k)} = U_k - L_k$ and we defined

$$\begin{aligned}
f_k(t) := -\lambda + \sum_{s=1}^{S} n_s^{(t)}\cdot\phi_t^s\cdot[\mathbb{E}_q[\ln r_k] + \mathbb{E}_q[\ln o_{k,s}] - \mathbb{E}_q\left[\ln\pi(\mathbf{b}^{(t)})_s\right]] \\
+ n_s^{(t)}\cdot[1-\phi_t^s]\cdot\mathbb{E}_q[\ln(1-r_k)] + n_s^{(t)}\cdot h_{t,s}
\end{aligned}$$

Intuitively, the function $f_1(t)$ measures the "gain" in the log-likelihood when adding t to the anomaly interval.

The first two terms of Eq. 9 can be removed since they are constant w.r.t. the bounds and q_3 and thus do *not* affect the optimal solution. Accordingly, maximizing Eq. 8/9 w.r.t. L_1, U_1, and q_3 is equivalent to solving

$$(L_1^*, U_1^*) = \arg\max_{(L_1,U_1)} \sum_{t=L_1}^{U_1} f_1(t) \quad \text{with } 1 \le L_1 \le U_1 \le T$$

Since the function f_1 is independent of L_1/U_1, i.e. the terms $f_1(t)$ are constant within the current optimization step, we can record all values $f_1(t)$ in a finite array of length T. Thus, the above problem corresponds to the *Maximum Subarray Problem*. Using Kadane's algorithm [4], this problem can be solved in time $O(T)$.

3.2.2 Non-uniform gaps between timestamps

So far, we assumed $\Delta^{(t)} = 1$ for all $t \in T$. We now generalize the above result to handle varying values for $\Delta^{(t)}$.

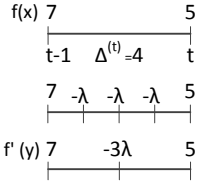

Fig. 4: Handling non-uniformity

W.l.o.g., due to finite precision in the measurement of time (e.g. UTC timestamps are usually measured in seconds), we can assume $\Delta^{(t)} \in \mathbb{N}^+$. Thus, a naive approach to handle the scenario of non-uniform gaps is to "blow up" the actual data by "artificial" time points where no ratings occur. After including the artificial time points, the Δ values are again equal to 1, and the previous technique can be applied. Figure 4 top and middle show this principle. Obviously, this principle is not suitable for huge time gaps and the new size of the array can be arbitrarily large.

Considering $f_1(t)$, it becomes apparent that its value evaluates to $-\lambda$ for each artificial time point. When searching for the subarray with maximal sum, these negative entries will never occur at the beginning/end of the anomaly interval [3]. If they would be at the beginning/end, one could easily shorten the interval to obtain a new one with higher sum. Thus, artificial time points are either completely contained in the interval or not included at all.

Using this result, we can safely "merge" all adjacent artificial time points to a single one with the function value $-\lambda\cdot u^{(t)}$, where $u^{(t)}$ is the number of artificial time points between time index t and $t-1$. Clearly, $u^{(t)} = \Delta^{(t)} - 1$ and the number of merged artificial time points is bounded by $T-1$. Overall, we can define a new array f' of size $2\cdot T - 1$ where

$$f_k'(y) = \begin{cases} f_k(\frac{y+1}{2}) & \text{if } y \text{ is odd} \\ -\lambda\cdot(\Delta^{(\frac{y}{2}+1)} - 1) & \text{if } y \text{ is even} \end{cases}$$

for $y \in [1, 2\cdot T - 1]$. And we now solve the problem

$$(a^*, b^*) = \arg\max_{(a,b)} \sum_{y=a}^{b} f_1'(y)$$

and set $(L_1^*, U_1^*) = (\frac{a^*+1}{2}, \frac{b^*+1}{2})$. Since the size of f' is bounded by $2\cdot T - 1$ the runtime complexity is $O(T)$.

3.2.3 Extension to multiple intervals

We now extend our result to multiple different anomalies/intervals. Using multiple anomalies affects the choice of the optimal q_3 distribution (cf. paragraph 'Observation' in Section 3.2.1). It is no longer sufficient to know whether $t \in [L_k, U_k]$ but we also have to know *which* k we consider.[2] Accordingly, for each anomaly k we have to use its individual function f_k/f'_k to measure the gain of adding a time point to the anomaly interval k. Overall, maximizing Eq. 8 using multiple intervals corresponds to solving

$$\arg\max_{(a_1,b_1),\ldots,(a_K,b_K)} \sum_{k\in K} \sum_{y=a_k}^{b_k} f'_k(y) \text{ with } a_k \le b_k < a_{k+1} \quad (10)$$

We solve the above problem by a dynamic programming technique. The necessary recursions are given by

$$g(1,1) = f'_1(1) \qquad g(1,t) = \max\{g(1,t-1) + f'_1(t), f'_1(t)\}$$
$$g(k,k) = m(k-1,k-1) + f'_k(k)$$
$$g(k,t) = \max\{g(k,t-1) + f'_k(t), m(k-1,t-1) + f'_k(t)\}$$
$$m(k,k) = g(k,k) \qquad m(k,t) = \max\{g(k,t), m(k,t-1)\}$$

Here, $m(k,t)$ (for $t \ge k$) denotes the maximal value of Eq. 10 when using k intervals and data up to point t. In contrast, $g(k,t)$ denotes the maximal value of Eq. 10 when the kth interval is forced to end at the t-th point in time (using k intervals and data up to t). Obviously, $g(k,t) \le m(k,t)$ holds. The value of $m(K,T)$ is the optimal value of Eq. 10.

We only provide a brief idea of these recursions: Assume we know the optimal intervals when using $k-1$ anomalies and data up to time point $t-1$. Let these intervals be denoted $(L_1,U_1),\ldots,(L_{k-1},U_{k-1})$. Additionally, assume the optimal intervals for k anomalies and data up to time point $t-1$ are given, denoted with $(L'_1,U'_1),\ldots,(L'_k,U'_k)$. Finally, assume the optimal intervals are given when the last interval is forced to end at time $t-1$ (we call these the g-intervals). Denote these with $(\hat{L}'_1,\hat{U}'_1),\ldots,(\hat{L}'_k,\hat{U}'_k)$, here $\hat{U}'_k = t-1$.

How will the solution for k intervals and data up to t look like? We can distinguish the following cases: (1) The time point t will be included in the optimal intervals. Obviously, since we are at the last point in time, it can only be represented by the kth interval. We can distinguish two subcases: (1a) The time point t is the beginning of the kth interval. In this case, the optimal intervals are $(L_1,U_1),\ldots,(L_{k-1},U_{k-1}),(t,t)$ and $m(k,t) = m(k-1,t-1) + f'_k(t)$. Since the last interval already ends at t, we also have $g(k,t) = m(k-1,t-1) + f'_k(t)$. (1b) The time point t is *not* the beginning of the kth interval. Thus, the optimal solution needs to be $(\hat{L}'_1,\hat{U}'_1),\ldots,(\hat{L}'_k,t)$ and we obtain $m(k,t) = g(k,t) = g(k,t-1) + f'_k(t)$.

(2) The time point t will *not* be included in the optimal intervals. In this case, since we want to find k intervals, the optimal solution is $(L'_1,U'_1),\ldots,(L'_k,U'_k)$ and $m(k,t) = m(k,t-1)$. For the g-intervals we have to distinguish two cases: (2a) The time point t is the beginning of the kth g-interval. In this case, the new g-intervals are $(L_1,U_1),\ldots,(L_{k-1},U_{k-1}),(t,t)$ and $g(k,t) = m(k-1,t-1) + f'_k(t)$. (Note that we use the optimal intervals from $m(k-1,t-1)$, not the g-intervals!) (2b) The time point t is *not* the beginning of the kth g-interval. Thus, leading to the solution $(\hat{L}'_1,\hat{U}'_1),\ldots,(\hat{L}'_k,t)$ with $g(k,t) = g(k,t-1) + f'_k(t)$.

[2]Technically, we could write $\phi^s_{t,k}$ to denote the optimal hyperparameter of q_3 assuming $t \in [L_k,U_k]$. We omitted k for brevity.

Exploiting the fact $g(x,y) \le m(x,y)$ and that we want to maximize $m(x,y)$, leads to the recursion as defined above. It is easy to add data structures to the method which record the start/end positions of the optimal intervals. Solving the above recursions via dynamic programming, we obtain:

THEOREM 1. *The optimal values for L_*, U_* and the optimal distributions q^*_3 can be computed in time $O(K \cdot T)$.*

3.3 Optimization of q_1, q_2, q_3

Following the principle of [5], the optimal distribution for q_3 can be determined by

$$\ln q^*_3(z_i^{(t)}) = \mathbb{E}_{q \setminus z_i^{(t)}}[\ln p(X,V,\Theta)] + const$$

Here, the constant *const* absorbs all terms which are independent of $z_i^{(t)}$ and, thus, do not affected the optimal distribution of q_3. The term $\mathbb{E}_{q \setminus z_i^{(t)}}[.]$ denotes the expectation with respect to the distribution q taken overall all variables except of $z_i^{(t)}$. Assuming $k(t) = k \neq 0$, and using the results from the appendix, it follows that

$$\ln q^*_3(z_i^{(t)} = 1) = \mathbb{E}_q[\ln r_k] + \mathbb{E}_q[\ln o_{k,s}] =: x$$

$$\ln q^*_3(z_i^{(t)} = 0) = \mathbb{E}_q[\ln(1-r_k)] + \mathbb{E}_q[\ln \pi(\mathbf{b}^{(t)})_s] =: y$$

where $s = x_i^t$. Therefore, the optimal value of the variational parameter is $\phi_{t,i} = \phi^s_t = \frac{e^x}{e^x + e^y}$.

The same principle can be applied for the distributions q_1 and q_2, leading to

$$\alpha_{k,s} = (\hat{\boldsymbol{\alpha}})_s + \sum_{t=L_k}^{U_k} n_s^{(t)} \cdot \phi^s_t$$

$$\alpha_k = \hat{\alpha} + \sum_{t=L_k}^{U_k} \sum_{s=1}^{S} n_s^{(t)} \cdot \phi^s_t \qquad \beta_k = \hat{\beta} + \sum_{t=L_k}^{U_k} \sum_{s=1}^{S} n_s^{(t)} \cdot (1-\phi^s_t)$$

3.4 Remaining Optimizations

Optimizing the base behavior. The base behavior can be updated for each $\mathbf{b}^{(t)}$ independently. Removing all terms from Equation 8 which are independent of $\mathbf{b}^{(t)}$ leads to

$$\mathbb{E}_q[\sum_i \ln p(x_i^{(t)}|...)] + \mathbb{E}_q[\ln p(\mathbf{b}^{(t)}|\widetilde{\mathbf{b}}^{(t-1)}, \mathbf{R})] + H(q_4(\mathbf{b}^{(t)})) \quad (11)$$

The first term is given in the appendix, and $H(q_4(\mathbf{b}^{(t)})) = \frac{S-1}{2}\ln(2\pi e v^{(t)})$. For the second term we derive:

$$\mathbb{E}_q\left[\ln p(\mathbf{b}^{(t)} \mid ...)\right] = -\frac{1}{2} \cdot \mathbb{E}_q\left[(\mathbf{b}^{(t)} - \widetilde{\mathbf{b}}^{(t)})^T \cdot \mathbf{R}^{-1} \cdot (\mathbf{b}^{(t)} - \widetilde{\mathbf{b}}^{(t)})\right] + c_1$$

$$= -\frac{1}{2}\mathbb{E}_q\left[{\mathbf{b}^{(t)}}^T \cdot \mathbf{R}^{-1} \cdot \mathbf{b}^{(t)}\right] + \mathbb{E}_q\left[{\mathbf{b}^{(t)}}^T \cdot \mathbf{R}^{-1} \cdot \widetilde{\mathbf{b}}^{(t)}\right] + c_2$$

$$= -\frac{1}{2}\left[Tr(\mathbf{R}^{-1} \cdot v^{(t)}) + {\boldsymbol{\mu}^{(t)}}^T \cdot \mathbf{R}^{-1} \cdot \boldsymbol{\mu}^{(t)}\right] + \widetilde{\boldsymbol{\mu}}^T_{t|T} \cdot \mathbf{R}^{-1} \boldsymbol{\mu}^{(t)} + c_3$$

We absorbed all terms which are independent of $\mathbf{b}^{(t)}$ into the constants c_i. Overall, Eq. 11 can be written as a function of $\boldsymbol{\mu}^{(t)}$ and $v^{(t)}$, which we optimize using gradient ascent.

Optimizing the smoothed base behavior. Since our model corresponds to a linear system, we can use a Kalman filter/smoother to determine the distribution of q_5. We use the Rauch-Tung-Striebel smoother. Thus, the distribution of q_5 can be computed efficiently by a forward and backward pass, leading to an update with runtime complexity $O(T)$.

Since the outputs $\mathbf{b}^{(t)}$ of the dynamic system are not observations but distributions, we slightly adapt the Kalman

update/innovation equations. Following the standard calculus of Kalman filters, the predicted mean and covariance matrix for time t (given data up to time $t-1$) are given by $\widetilde{\boldsymbol{\mu}}_{t|t-1} = \widetilde{\boldsymbol{\mu}}_{t-1|t-1}$ and $\mathbf{P}_{t|t-1} = \mathbf{P}_{t-1|t-1} + \Delta^{(t)} \cdot \mathbf{Q}$. Given the measurement at time t, the measurement residual can be computed as $\mathbf{e}_t = \mathbb{E}_q[\mathbf{b}^{(t)}] - \widetilde{\boldsymbol{\mu}}_{t|t-1}$. Accordingly, the residual covariance is given by $\mathbf{S}_t = \mathbf{P}_{t|t-1} + \mathbf{R} + v^{(t)} \cdot I$. Note the increased variance due to the uncertainty of the base behavior. Letting the Kalman gain be defined by $\mathbf{K}_t = \mathbf{P}_{t|t-1} \cdot \mathbf{S}_t^{-1}$, we see that the Kalman gain is smaller for time points showing a high variance, i.e. high uncertainty, in the base behavior. These points affect the smoothed base behavior less strongly.

Continuing with the standard calculus, the updated mean and covariance are $\widetilde{\boldsymbol{\mu}}_{t|t} = \widetilde{\boldsymbol{\mu}}_{t|t-1} + \mathbf{K}_t \cdot \mathbf{e}_t$ and $\mathbf{P}_{t|t} = (I - \mathbf{K}_t) \cdot \mathbf{P}_{t|t-1} \cdot (I - \mathbf{K_t})^T + \mathbf{K}_t \cdot \mathbf{R} \cdot \mathbf{K}_t^T$. Here, we used the Joseph form of the covariance update equation since it holds for any value of $\mathbf{K}_t$. For the backward pass, the RTS smoother leads to $\widetilde{\boldsymbol{\mu}}_{t|T} = \widetilde{\boldsymbol{\mu}}_{t|t} + C_t \cdot (\widetilde{\boldsymbol{\mu}}_{t+1|T} - \widetilde{\boldsymbol{\mu}}_{t+1|t})$ and $\mathbf{P}_{t|T} = \mathbf{P}_{t|t} + C_t \cdot (\mathbf{P}_{t+1|T} - \mathbf{P}_{t+1|t}) \cdot C_t^T$, with $C_t = \mathbf{P}_{t|t} \cdot \mathbf{P}_{t+1|t}^{-1}$. Due to space limitations, we kindly refer to the rich literature on Kalman filter/smoother, for details about the derivations.

Optimization of Q and R. Updating $\mathbf{Q}$ and $\mathbf{R}$ follows from the properties of the conjugate prior. Note that Eq. 5 can also be written in the form $\widetilde{\mathbf{b}}^{(t)} - \widetilde{\mathbf{b}}^{(t-1)} \sim \mathcal{N}(0, \Delta^{(t)} \cdot \mathbf{Q})$. Also, by the definition of the normal distribution it holds that $\mathcal{N}(\mathbf{x}|0, \Delta \cdot \Sigma) = \Delta^{-d/2} \cdot \mathcal{N}(\mathbf{x}/\sqrt{\Delta}|0, \Sigma)$, where d is the dimensionality of the distribution. Since the terms $\Delta^{-d/2}$ are constant when optimizing the log-likelihood, they can be ignored. Thus, $\mathbf{Q}$ can be seen as the covariance matrix of a Normal distribution with *known* mean of zero. Correspondingly, we can use the Inverse-Wishart distribution as its (conjugate) prior. Following the results of conjugacy, the posterior distribution of $\mathbf{Q}$ is an Inverse-Wishart distribution $\mathcal{W}^{-1}(\Psi_q, \nu_q)$ with $\nu_q = \nu_q^0 + T - 1$ and scale matrix

$$\Psi_q = \Psi_q^0 + \sum_{t=2}^{T} \mathbb{E}_q \left[\frac{1}{\Delta^{(t)}} \left(\widetilde{\mathbf{b}}^{(t)} - \widetilde{\mathbf{b}}^{(t-1)} \right)^T \cdot \left(\widetilde{\mathbf{b}}^{(t)} - \widetilde{\mathbf{b}}^{(t-1)} \right) \right]$$

which can easily be computed by plugging in the known expectations. Given this distribution, the MAP estimate for $\mathbf{Q}$ can efficiently determined by selecting the mode of the Inverse-Wishart distribution, i.e. $\mathbf{Q}^* = \frac{1}{(S-1)+1+\nu_q} \cdot \Psi_q$. The same principle can be applied for $\mathbf{R}$.

3.5 Overall Processing and Complexity

Using the above optimizations and update equations, our method iteratively recomputes the values for Θ and Ω. If the change in Equation 8 is less than 0.1% we assume convergence and terminate. Based on the previous results, and assuming that $K, S << T$, each iteration is linear in the number of time stamps, i.e. we have a complexity of $O(T)$.

4. RELATED WORK

Spotting anomalies in rating data: So far, only [7] considers the temporal analysis of rating data incorporating potentially anomalous behavior. The work models the rating data as distributions over time. As mentioned in the introduction, it requires an aggregation/binning of the data and it cannot handle intervals of anomalies. We compare our technique against [7] in the experimental analysis.

Modeling of temporal continuous data: Similar to the work [7], traditional time series modeling methods such as vector autoregression [14, 13] or Kalman filter/smoother [5], analyze continuous data. They are not directly suited for our scenario of categorical data (or require a problematic binning). Furthermore, traditional approaches for time series modeling are sensitive to outliers. Thus, these models fail to find good approximations of the data corrupted by anomalies. Therefore, robust techniques to handle outliers have been proposed [16]. These methods are designed to handle *outliers* which are attributed to mostly independent, random corruptions of the data, while our work is designed to handle *anomalies* following a specific pattern.

Since in our work the Kalman filter operates on the (clean) base behavior, i.e. the anomalies have been 'removed' by the other mixture model components, the problem of anomalies is circumvented. We compare our technique against a Kalman filter in the experimental analysis.

Modeling of temporal documents: One might represent the ratings at a certain point in time as a document with the words corresponding to the ratings' evaluations. Modeling temporal document collections is handled by dynamic topic mining [6, 17, 2]. Applying these methods on the 'documents' generated via the ratings is questionable since each document most likely would contain only a single 'word'. Ignoring this issue, further problems for our scenario are: First, [6, 2] require a binning of the documents in fixed time slots. Second, [6, 17] require that topics exist over the whole lifetime. In our work, however, anomalies exist only in specific time intervals. While [2] allows topics to appear and disappear, they prefer smooth evolutions. In our case, however, anomalies abruptly appear/disappear in time. Also, all of these techniques are (of course) designed to detect multiple topics. In our scenario, however, we want to find a single base behavior which captures the general temporal evolution, enriched by a few number of anomalies.

Related applications: Multiple techniques have been proposed in the area of *outlier detection* [1]. While the majority of techniques tackles the case of independently distributed data, time-series outlier detection and outlier detection for streaming data are also an active field of research [1]. Both areas differ from our work since they are designed for continuous data. Also, most existing techniques consider outlier in the sense of independent, random errors in the data. *Change detection* techniques detect points in time where the state of the underlying system has changed [15]. A change might not generally indicate anomalous behavior. Indeed, even the base behavior might change over time.

Studying product ratings has been done in multiple research areas, all following different goals and objectives. *Recommender Systems* incorporate ratings and their temporal information [9, 10] to improve the prediction performance. *Opinion mining* aims at extracting the sentiment of users regarding specific products or features of a product [18]. *Modeling* of social rating networks, e.g., to compactly describe the underlying mechanism driving the network or to generate synthetic data, has been studied, e.g, in [12].

None of the existing methods is designed to detect anomalies and the underlying evolving base behavior in rating data.

5. EXPERIMENTAL ANALYSIS

We applied our method (called SpotRate due to its potential to spot anomalies in rating data) on *over six million* product ratings representing varies categories: an extract of the Amazon website [8] evaluating multiple different products, another subset of the Amazon website evaluating

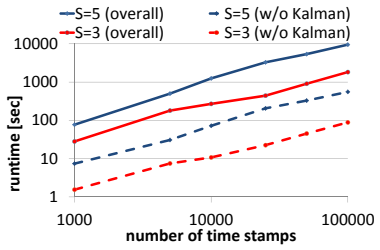

Fig. 5: Runtime vs. number of time stamps

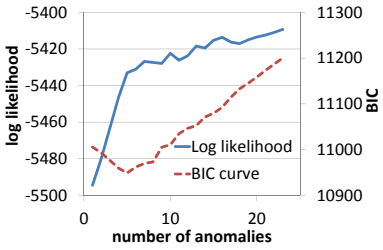

Fig. 6: Likelihood and BIC vs. K (synthetic data)

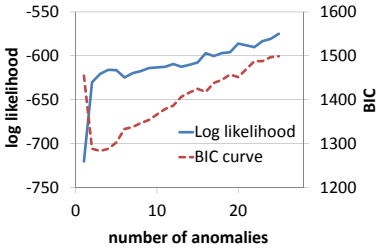

Fig. 7: Likelihood and BIC vs. K (real world data)

food products[3], ratings of restaurants in the area of Phoenix based on Yelp, and an extract of the TripAdvisor website[4] for hotel ratings. The data consists of tuples representing the ID of a product/service to be rated, the user who evaluated the product, the time stamp when the rating occurred, and a star rating in the range from 1 up to 5. Additionally, these datasets contain textual reviews, which we used to understand and describe the results of our method.

Besides these real world datasets we used synthetic data generated based on the presented process to analyze the scalability and robustness of our method.

5.1 Runtime Analysis

We briefly analyze the runtime of SpotRate. The runtime is primarily affected by the number of time stamps T and the rating scale S. The actual number of ratings does not affect the runtime (cf. Sec. 3.1.1). For the runtime analysis, we selected the product B00003TL7P from the Amazon dataset and we extended it to different length (from 1,000 to 100,000) by concatenation. Besides the original rating scale of $S = 5$, we used a rating scale of $S = 3$ by merging 1/2 and 4/5 ratings. All experiments were conducted on commodity hardware with 3 GHz CPU's and 4 GB main memory.

The results are shown in Fig. 5. Confirming our study of Sec. 3.5, the runtime increases linearly, showing the method's high scalability (note the slope of 1 in the log-log plot). The overall runtime for 100,000 time stamps (which would corresponds to 273 years when measured on a daily basis) is only about 158 minutes on commodity hardware. A brief study shows that the currently most rated products have around 20,000 (Amazon: Kindle Fire) or 8,000 (Yelp: Bottega Louie) ratings. Thus, even when considering the finest granularity, we highly exceed this number.

Additionally, we measured the runtime of our method when ignoring the time required for the Kalman smoother (dashed lines). As shown, the Kalman smoother contributs to around 90% of the absolute runtime. The remaining parts of our method are highly efficient.

We also studied the effect of the number of anomalies K on the runtime. According to Sec. 3.2.3, K linearly affects the runtime of the dynamic programming technique. Since the Kalman smoother (whose runtime is independent of K) accounts for most of the absolute runtime, we only observed a very small change of only a few seconds. Thus, overall, only T and S influence our method's practical applicability.

5.2 Effectiveness

In the following, we analyze the effectiveness of SpotRate considering different aspects. We start with the model selection principle. For this experiment, we generated synthetic data according to our model. We used 4000 ratings with 5 anomaly intervals. Figure 6 shows on the (first) y-axis the obtained log-likelihood of our method when varying the number K of potential anomalies. Obviously, the general trend shows that increasing K also increases the log-likelihood: more flexibility to describe the data is given. A very high increases is obtained until the value of 5, which corresponds to the true number of anomalies. After this point, the benefit of allowing further anomalies decreases.

This effect is well captured by the BIC score, which is shown on the second y-axis of the figure. The minimal BIC value is obtained for the value of 5. Thus, the model selection principle introduced before can be used as a good indicator how to select the number of anomalies.

The same behavior can be observed for real world data as, e.g., shown in Fig. 7. Here we plotted the log-likelihood and BIC score for a coconut-water sold on Amazon (cf. Sec. 5.4). Again, one sees a clear minimum of the BIC value, indicating that three anomaly intervals describe the data very well.

Next, we analyze our iterative optimization. In Fig. 8 we analyze how the log-likelihood increases when we increase the number of iterations until convergence. That is, on the x-axis we count how often the variables have been updated, while the y-axis shows the log-likelihood. We plotted the curves for different values of K, again for the product B00003TL7P. As expected, the first iterations lead to the highest improvement in the log-likelihood. Still, we see an improvement in the later iterations, showing the effectiveness of the optimization step. As also shown in the previous experiment, a higher value of K leads to a better likelihood. Additionally, for this product, we observed that a smaller number of intervals can lead to a lower number of required iterations. In general, however, the difference in the number of iterations was not as significant as shown for this product.

Finally, we analyze the effect of λ. Per default, a value of 0 can be selected to realize a non-informative prior. In Figure 9, we varied the value of λ between 1 and 0. We selected $K = 10$. As shown, for larger values of λ, shorter intervals are preferred. In particular, for $\lambda = 1$ the average interval length is close to the shortest possible length of 1. For $\lambda = 0$ larger intervals are captured. Note that $\lambda = 0$ does not mean that the whole set of time stamps is represented as an anomaly interval. Even in the case of $\lambda = 0$, we only report time intervals where the behavior is anomalous.

5.3 Comparison with related techniques

We compare SpotRate against the related technique RLA [7] and a Kalman smoother. Doing a fair comparison between these approaches is challenging since the data they analyze and goals they follow are different. In particular, the

[3] `http://snap.stanford.edu/data/`

[4] `http://sifaka.cs.uiuc.edu/~wang296/Data/`

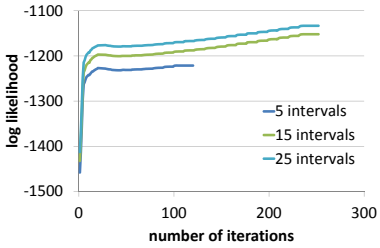

Fig. 8: Convergence analysis

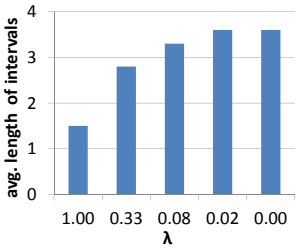

Fig. 9: Effect of λ

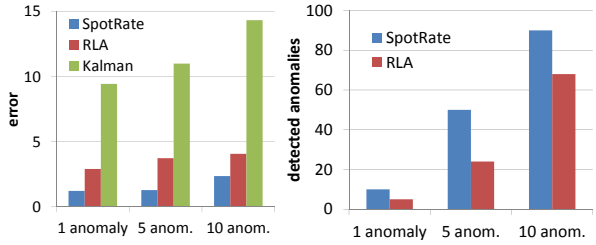

Fig. 10: Comparison with related methods

work [7] requires an aggregation of the data since it operates on the rating distributions. Thus, e.g., measuring the likelihood on the categorical data is questionable. Since RLA, however, is the only existing technique which also analyzes dynamic rating data, we try to study some effects.

For comparing the methods, we use two principles: In the first experiment, we compare the base behavior detected by the methods against the known base behavior for synthetically generated data. The base behavior is continuous in our model as well as in RLA, and for the Kalman smoother. Thus, it is fair to, e.g., measure the Frobenius norm between the ground truth base behavior and the detected ones. We generated data with 1000 time points and added a varying number of anomaly intervals to it, each covering 10 time points. We ensured that the anomaly intervals exactly match the aggregation required for RLA. Thus, this method gets a huge advantage since this assumption does not necessarily hold for real data. Fig. 10 (left) shows the results: Our method obtains the lowest error, it is able to detect the hidden base behavior. The Kalman smoother cannot handle anomalies and shows a high deviation to the ground truth.

In a second experiment, we evaluated whether the methods are able to detect the anomalous points in time (this is only possible for SpotRate and RLA). As shown in Figure 10 (right), our method almost perfectly detects all anomalous points in time. RLA in contrast is not able to spot all points, which also explains the previously observed higher error to the base behavior. Overall, our method outperforms the competing techniques in detecting the correct base behavior as well as spotting the anomalous points in time.

5.4 Discoveries

In the following, we will demonstrate the application of SpotRate by illustrating some of our interesting discoveries.

(1) We start with the example illustrated in Fig. 1. It represents a hotel in the Caribbean evaluated on TripAdvisor. While understanding the original time-stamped data is difficult, the extracted base behavior allows an easy understanding: clearly, the hotel is evaluated with mostly 4 and 5 stars. Our method found anomalous behavior in July and August 2005. In this time frame, the negative ratings highly increased. Analyzing the reviews at the detected time points, the reviewers criticized *"the restaurants with ridiculous reservation rules''* often showing overbooking and *"the nonfunctional air-conditions"*. These reviews indicate that in the given months the service of the hotel has dropped, potentially due to a highly increased number of guests. Our method was able to spot these anomalies, and it successfully smoothed out these points from the base behavior.

(2) Next, we show the result for a coconut-water sold on Amazon (http://www.amazon.com/dp/B000CNB4LE). Applying our method leads to the base behavior as shown in Figure 11. The three detected anomaly intervals appear at the end of 2010. As shown next to the figure, the detected anomalies are described by distributions $\mathbf{o}_k$ representing primarily low ratings. They clearly deviate to the base behavior. Inspecting the product's reviews during these times, most customers are not satisfied with the *"new plastic bottles"* the manufacturer has introduced, leading to a bad taste. Later time points do not show this anomalous behavior, indicating that the manufacturer has solved this problem (*"I can understand a lot of the initial bad reviews as I thought the new plastic bottle had a bad after taste. ... I can say that the taste is much improved ..."*).

(3) Next, we want to show the benefit of extracting an evolving base behavior. Figure 12 shows the base behavior of a baby bouncer (B00005QI1G) from the Amazon data. Looking at its evaluation, it is recognizable that the majority of reviewers evaluated this product with 5 stars. At the later time points, however, the number of low and medium ratings increases. Note that these intervals are not classified as anomalies but they represent the general evolution of the product. A closer look at the product's reviews at these time points explains that over time the customers were more and more unsatisfied by the product since it is *"nice to play but not long lasting"* and the *"battery simply does not last very long with the vibrating feature"*.

Discoveries via prediction. Finally, we want to show the potential of our method to detect anomalies via prediction. According to Eq. 7, we can predict the base behavior at future points in time. By comparing it against newly arriving ratings, anomalies can be spotted. We removed from all restaurants of the Yelp dataset the last 10 points in time. We applied our method on the remaining data. Figure 13 shows three restaurants whose predicted base behavior (left bar [a] of each diagram) highly deviates to the observed ratings (right bar [b]), thus, potentially indicating anomalies.

Inspecting the reviews of the first restaurant, we see comments like *"I've been eating at Stacy's for over a year so it pains me to kill them but the service [...] was pathetic. [...] I don't know if its a new employee or something going wrong but I'm probably not going back..."*. Thus, indicating that the service quality of the (previously very highly rated) restaurant has suddenly dropped.

For the second restaurant, we observed comments like *"All the prices have went up"* and *"The picture of the menu and prices is out dated"*, which again indicates a recent deviation to the previous behavior, potentially due to increased prices.

Finally, the reason for the abruptly appearing low ratings of the third restaurant seems to be caused by expanding/remodeling the old building. The old atmosphere of the restaurant seems not to be preserved and the larger capacity could not be handled by the service staff: *"the expanded building is nice [...] but something was lost. we didn't have*

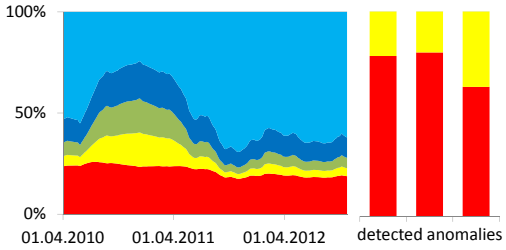

Fig. 11: Base behavior and anomalies (intervals spotted at the end of 2010).

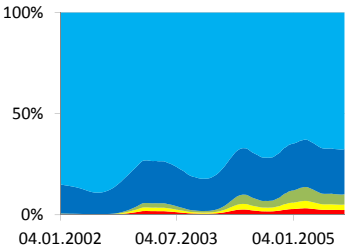

Fig. 12: Evolving base behavior (Amazon)

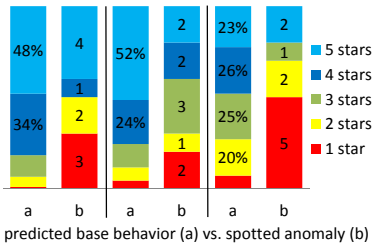

Fig. 13: Spotting anomalies via prediction (Yelp)

that hometown feel. we miss all the signatures and pictures." and *"I think its rad that you expanded, but if you cant handle the customer load then whats the point?"*.

Overall, by modeling the temporal evolution of the base behavior, our method is able to detect these newly occurring anomalies, which can then, e.g., be used to inform the corresponding companies.

6. CONCLUSION

We developed the method SpotRate for analyzing time stamped rating data. Our method detects the users' base behavior as well as time intervals representing anomalies. We proposed a sound Bayesian framework which represents the rating data via temporally constrained categorical mixture models. It accounts for the temporal evolution of the base behavior and enables us to predict the rating behavior for newly occurring ratings. We developed an efficient algorithm which exploits principles of variational inference and dynamic programming. Our experimental study has shown the potential of our method to spot anomalies and to use the base behavior for studying the evolution of a product.

Acknowledgments. Stephan Günnemann has been supported by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD). Research was also sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

7. REFERENCES

[1] C. C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[2] A. Ahmed and E. P. Xing. Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream. In *UAI*, pages 20–29, 2010.

[3] F. Bengtsson et al. Computing maximum-scoring segments in almost linear time. In *Computing and Combinatorics*, pages 255–264. Springer, 2006.

[4] J. Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

[6] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML*, pages 113–120, 2006.

[7] N. Günnemann, S. Günnemann, and C. Faloutsos. Robust multivariate autoregression for anomaly detection in dynamic product ratings. In *WWW*, pages 361–372, 2014.

[8] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

[9] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[10] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[11] G. F. Lawler. *Introduction to stochastic processes*. CRC Press, 2006.

[12] K. Lerman. Dynamics of a collaborative rating system. In *WebKDD/SNA-KDD*, pages 77–96, 2007.

[13] R. B. Litterman. Forecasting with bayesian vector autoregressions. *Journal of Business & Economic Statistics*, 4(1):25–38, 1986.

[14] H. Lütkepohl. *New introduction to multiple time series analysis*. Cambridge University Press, 2005.

[15] X. Song, M. Wu, C. M. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *KDD*, pages 667–676, 2007.

[16] J.-A. Ting, E. Theodorou, and S. Schaal. Learning an outlier-robust kalman filter. In *ECML*, pages 748–756, 2007.

[17] C. Wang, D. M. Blei, and D. Heckerman. Continuous time dynamic topic models. In *UAI*, pages 579–586, 2008.

[18] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *KDD*, pages 783–792, 2010.

APPENDIX

Let $[\![.]\!]$ denote the Iverson bracket, it holds: $\mathbb{E}_q[[\![z_i^{(t)}{=}1]\!]]{=}\phi_{t,i}$
$\mathbb{E}_q[[\![z_i^{(t)}{=}0]\!]]{=}1-\phi_{t,i}$ $\quad\mathbb{E}_q[\ln \mathbf{o}_{k,s}]{=}\psi(\alpha_{k,s})-\psi(\sum_j \alpha_{k,j})$
$\mathbb{E}_q[\ln r_k]{=}\psi(\alpha_k){-}\psi(\alpha_k{+}\beta_k)$ $\mathbb{E}_q[\ln(1{-}r_k)]{=}\psi(\beta_k){-}\psi(\alpha_k{+}\beta_k)$

Given the definition of the distribution p, it follows:

- For $k(t)=k\neq 0$ it holds:
  $\mathbb{E}_q[\ln p(z_i^{(t)}|\ldots)] = \mathbb{E}_q[\ln r_k^{[\![z_i^{(t)}=1]\!]}\cdot(1-r_k)^{[\![z_i^{(t)}=1]\!]}]$
  $= \mathbb{E}_q[[\![z_i^{(t)}=1]\!]]\cdot\mathbb{E}_q[\ln r_k] + \mathbb{E}_q[[\![z_i^{(t)}=0]\!]]\cdot\mathbb{E}_q[\ln(1-r_k)]$
- $\mathbb{E}_q[\ln p(x_i^{(t)}|\ldots)] = \mathbb{E}_q[[\![z_i^{(t)}=1]\!]]\cdot\mathbb{E}_q[\ln o_{k(t),s}]+$
  $\mathbb{E}_q[[\![z_i^{(t)}=0]\!]]\cdot\mathbb{E}_q[\ln\pi(\mathbf{b}^{(t)})_s]$ for $x_i^{(t)}=s$
- $\mathbb{E}_q[\sum_i \ln p(x_i^{(t)}|\ldots)] = \mathbb{E}_q\left[\sum_{s=1}^S n_s^{(t)}\cdot\ln p(x=s\mid\ldots)\right]$
  $= \sum_{s=1}^S n_s^{(t)}\cdot\phi_t^s\cdot\mathbb{E}_q[\ln o_{k(t),s}] + n_s^{(t)}\cdot[1-\phi_t^s]\cdot\mathbb{E}_q[\ln\pi(\mathbf{b}_t)_s]$
- $\mathbb{E}_q\left[\ln\pi(\mathbf{b}^{(t)})_s\right] = \mathbb{E}_q\left[\ln\frac{e^{b_s^{(t)}}}{1+\sum_{s=1}^{S-1}e^{b_s^{(t)}}}\right] = \mathbb{E}_q\left[\mathbf{b}_s^{(t)}\right] -$
  $\mathbb{E}_q\left[\ln(1+\sum_{s=1}^{S-1}e^{\mathbf{b}_i^{(t)}})\right] \geqslant \mu_s^{(t)} - \ln(\mathbb{E}_q\left[1+\sum_{s=1}^{S-1}e^{\mathbf{b_s}^{(t)}}\right])$
  $= \mu_s^{(t)} - \ln(1+\sum_{s=1}^{S-1}e^{\mu_s^{(t)}+\frac{v^{(t)}}{2}})$

3.3 Finding non-redundant multi-word events on Twitter

Peer-reviewed Conference Paper

Autors **Nikou Günnemann**, Jürgen Pfeffer

In *International Conference on Advances in Social Networks Analysis and Mining, pp. 520-525, 2015, [21].*

Abstract Twitter is a pervasive technology, with hundreds of millions of users serving as sensors that provide eyewitness accounts of events on the ground. In case of popular events, these sensors start to broadcast news by tweeting to their followers, and to the world. Within minutes these tweets can attract attention and also serve as a primary information source for traditional media. Given a huge set of tweets, the key questions are: (1) How can we detect informative events in general? (2) How can we distinguish relevant events from others?

In this paper we tackle these challenges with a statistical model for detecting events by spotting significant frequency deviations of the words' frequency over time. To detect unexpected events, we used the complementary cumulative distribution function as a lower bound for a significant threshold to detect the set of single words occurring suspiciously frequent. Since single word events do not necessarily capture all important information, our statistical process is complemented with an approach of finding events consisting of multiple co-occurring words. For this, we developed a formulation based on the principle of minimum dominating sets ensuring that events do not cover redundant information. The minimum domination set is finally approximated by an greedy algorithm, overall, leading to non-redundant events that provide the user with a succinct summary of the current events.

We used our model to analyze 24 million geotagged tweets that have been sent in the US from April 9 to April 22, 2013 - the time period of the Boston marathon bombing - and we show that our approach can create multi-word events that efficiently summarize real-world events. Besides the Boston bombing event, other interesting findings detected by our algorithm are presented in this paper.

Contribution of thesis author: Algorithmic design, development and implementation, experimental design and analysis, manuscript writing, revision and editing.

Finding Non-Redundant Multi-Word Events on Twitter

Nikou Günnemann
Carnegie Mellon University, USA
Email: nguennem@cs.cmu.edu

Jürgen Pfeffer
Carnegie Mellon University, USA
Email: jpfeffer@cs.cmu.edu

***Abstract*—Twitter is a pervasive technology, with hundreds of millions of users serving as sensors that provide eyewitness accounts of events on the ground. In case of popular events, these sensors start to broadcast news by tweeting to their followers, and to the world. Within minutes these tweets can attract attention and also serve as a primary information source for traditional media. Given a huge set of tweets, the key questions are: (1) How can we detect informative events in general? (2) How can we distinguish relevant events from others? In this paper we tackle these challenges with a statistical model for detecting events by spotting significant frequency deviations of the words' frequency over time. Besides single word events, our model also accounts for events composed of multiple co-occurring words, thus, providing much richer information. Our statistical process is complemented with an optimization algorithm to extract only non-redundant events, overall, providing the user with a succinct summary of the current events. We used our model to analyze 24 million geotagged tweets that have been sent in the US from April 9 to April 22, 2013 – the time period of the Boston marathon bombing – and we show that our approach can create multi-word events that efficiently summarize real-world events.**

I. Introduction

Twitter is a major player in the social media space, with 200 million users publishing 400 million tweets, each day [17]. Many Twitter users publish their tweets with their mobile devices, allowing them to update in real-time the events taking place around them, acting as "social sensors" for real events on the ground. These social sensors have provided real-time eyewitness accounts for major events in the past few years, including the Arab Spring and Occupy Wall Street protests, Hurricane Sandy, and numerous earthquakes around the globe. Tragic events are no exception, causing users to flock to Twitter both to seek information and to provide a firsthand account from their perspective. But also publishing news about a festival can be ranked among events.

Generally speaking, Twitter is the source of live notification related to people activities, observations, and thoughts. Exploiting and analyzing Twitter, as a massive amount of data source can help us by understanding the run of events around the world. Hence, the key research questions of our study by this paper are, "what are the major characteristics of relevant events as they pass through Twitter?" and "how can we detect relevant event occurrence by monitoring tweets"?

In our model, we do not limit to single word events in contrast to existing work for event detection [4] since it is questionable to construct a meaningful statement out of a single word. On this account, we belief that events described by arbitrary word length called *multi-word* events are much more expressive. E.g. the individual words 'boston' or 'bombing' are not specially predicative in which context they are used while 'boston bombing' provides a more thorough picture. Thus, our model provides the ability of detecting multi-word events.

As a major challenge, however, the detected multi-word events can (possibly) include redundant information: the same event can be expressed and represented in different ways. To avoid an excessive amount of events, we should select only those events which are most representative from the set of detected multi-word events. We formulate this task as an optimization problem where we minimize the number of events required to explain the whole population of events. Overall, the final result of events allows an easy analysis and interpretation for the end user. In summary, our contributions are:

- *Model*: We propose a novel event detecton model that accounts for multi-word events and that selects a small amount of non-redundant events, thus, allowing easy interpretation.
- *Algorithm*: By exploiting relations of our model to itemset mining and to the dominating set problem, we derive an efficient algorithmic solution of our problem.
- *Experiments*: We perform a case study on a large Twitter dataset with more than 24 million tweets, confirming the potential of our method to detect important events. We perform a temporal, and content-based analysis on the data to help providing an understanding of how a major events happens through the lens of Twitter.

II. Related Work

Event detection has been one of the most important research topics in social media analysis. Twitter is the most popular one since it reflects and reports real life information [12]. In this context, the authors of [15] provide an algorithm for classifying tweets based on features such as keywords in tweets or the number of words. Also Li et. al. [8] use spatial and temporal information for detecting crime and disaster related events. In their model they use text mining technique to extract keywords. Mendoza et. al. [9] use location-based properties of social networks to gain a signal from social media regarding crisis events. In [13] methods have been proposed for detecting bursts of individual words, particularly focusing on Flikr tags and spatial information. Finally, in Cataldi et. al. [5] the focus is on identifying emerging words which have been used extensively in a certain time interval. They identify terms that frequently co-occurred with the already detected keywords. Then they declare these together as emerging topics.

Applying these methods for detecting "events" is questionable, since (1) not every *single* keyword expresses the history of an event. E.g. the word "boston" can be referred to any event while "boston bombing" much clearer describes what the event is about. *In our model, we do not limit to single word events.* (2) Combining words that frequently co-occurred, do not necessarily refer to the same event. The interpretation of such topics as events could be even misleading. In our scenario, the co-occured multi-words are not combined accidentally, but they stem from the same tweet. It is important to mention here that *we do not use n-grams* [8], [2], since the main drawback of the n-gram model is that it suffers from data sparseness. In our method, we allow words to occur at various localities in the tweet. Thus, also long range dependencies as, e.g., mentioned in [10], can be captured. This is an important aspect for short text content like tweets. Various methods have been proposed for identifying different types of events in social media, such as news events [16] and natural disasters like earthquakes [15]. In contrast to this work, we do not focus on a certain type of events. In our paper, we try to extract all the possible events that are contained in our twitter data. This line of research is also know as open domain event detection [14]. While [14] relies on the principle of entity recognition, we detect multi-word events by identifying significant frequency deviations. Additionally, while some of the works assume that the keywords we are interested in are already *given* [15], we do not assume any prior information but automatically find the corresponding events.

Finally, none of the existing work considers the aspect of redundancy: different words (or in our case multi-words) might describe the same event. Thus, to avoid reporting a huge set of redundant patterns, we propose an optimization approach selecting a small set of events that provide a succinct summary.

III. Non-Redundant Multi-Word Event Detection

In this section, we introduce our model for detecting non-redundant multi-word events from Twitter data. We aim to detect and analyze the temporal occurrence of events. In this regard, we preprocess the tweets to facilitate the analysis we carry out. The preprocessing step extracts the set of distinct words from the raw data, it applies stop-word lists and performs stemming (cf. Section IV). We denote with $\mathcal{W} = \{w_1, \ldots, w_n\}$ the set of all words obtained after preprocessing.

The resulting data we consider is now represented by a database $\mathcal{T} = \{T_1, \ldots, T_k\}$ of tweets, where each tweet T_i is represented by a triplet $T_i = (u_i, W_i, t_i)$, with u_i being a user id, $W_i \subseteq \mathcal{W}$ being the sets of words used in the Tweet, and t_i the timestamp of the Tweet, measured, e.g., hourly. We denote with $O_w(t)$ the tweets that contain word w at time t, and with $N_w(t)$ the corresponding cardinality, i.e.

$$O_w(t) := \{T \in \mathcal{T} \mid T = (u, W, t) \wedge w \in W\} \quad (1)$$

$$N_w(t) := |O_w(t)| \quad (2)$$

Accordingly, we use $O_W(t)$ to denote the tweets at time t containing the *set* of words W, and $N_W(t)$ respectively. As an abbreviation, we write $k \in K$ for $k \in \{1, \ldots, K\}$.

A. Event Detection Model

We define an "event" as an appearance of an incident that attracts the attention of people to post messages about it within a short period of time. With a large corpus, it is hopeless to try to read each individual document to find the underlying events. In the following, we propose an automatic model for event detection.

Given the dataset $\mathcal{T}$, we assume that important events are characterized by a massive amount of tweets within a short period of time. Consequently, we are interested in determining words that people discussed above average. Formally, we classify events as those words shown significant deviation from the normal behavior.

1) Estimating the normal behavior: How to estimate the normal behavior of words? Clearly, considering the temporal behavior of the average usage of each individual word over the last time points helps us to analyze this behavior. That is, for each time t and word w, we consider the number of tweets $\{N_w(t-d)\}_{d \in D} =: X_{t,w}$ where $D \in \mathbb{N}_+$ specifies the time window (e.g. 24 hours).

Given the set of observations $X_{t,w}$, we assume the normal behavior follows a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with unknown μ and σ. This assumption is backed based on two reasons: First, the Gaussian distribution is the maximum entropy distribution among all distribution with a given mean and variance. That is, we are not biased to specific scenarios which we are not aware of, but only take into account the expected occurrence of tweets and their fluctuations. Second, the Poisson distribution, which is well suited for count data, converges to the Gaussian distribution, where, however, mean and variance are coupled. Using the Gaussian distribution, adds an extra degree of freedom since mean and variance can be chosen independently.

How can we estimate the values μ and σ of the normal behavior? Naively, this can be achieved by maximizing the data's likelihood. For the scenario of event detection and particularly short text messages as in Twitter, this approach, however, is highly un-robust. If the number of tweets including a certain word is small, estimates of variance can be noisy and often equal to zero. A zero variance can go against prior knowledge and results in underestimation of uncertainty in fixed coefficient estimates [6]. To bypass this problem, we refer to maximizing the posterior distribution, thus, representing a sound Bayesian approach. That is, we choose a prior distribution over the mean and variance, which can incorporate external domain knowledge and ensures well-defined solutions. Formally, we maximize

$$\max_{\mu,\sigma} p(\mu, \sigma | X_{t,w}) = \max_{(\mu,\sigma)} \{\mathcal{N}(X_{t,w} | \mu, \sigma) \cdot p'(\mu, \sigma)\}$$

By selecting the prior p' to be a Normal-inverse gamma distribution $NIG(\mu_0, \nu_0, \alpha_0, \beta_0)$, we realize conjugacy [3]. That is, we ensure that the posterior distribution $p(\mu, \sigma | w_1, \ldots, w_n)$ is also a Normal-inverse gamma distribution. Thus, the optimal parameters

$$\mu^*, \sigma^* = \arg\max_{\mu,\sigma} [p(\mu, \sigma | X_{t,w})]$$

can efficiently be obtained by computing the mode of the resulting (posterior) Normal-inverse gamma distribution [11], [3]. If no prior knowledge about the data is given, we can use an uninformative/least informative prior by selecting $\lambda_0, \alpha_0, \beta_0 \to 0$ and using μ_0 as the mean of data. This approach is used in our experiments. Please note that using

this (uninformative) prior is still advantageous compared to the maximum likelihood solution since we can always ensure that the final solution is well-defined.

2) Detecting significant events: ***Finding single word events.*** Given $\mu_{w,t}$ and $\sigma_{w,t}$ regarding a certain word w at time t, we now turn our attention to extracting time points at which the usage of w is increased suspiciously compared to the previous time points. For this purpose, we exploit the idea of significance testing: If the value $N_w(t)$ is unlikely to observe regarding the null hypothesis of the normal behavior, it is classified as an event. Thus, given the normal behavior modeled as $V \sim \mathcal{N}(\mu_{w,t}, \sigma_{w,t})$, we are interested in the probability $p(V \geq N_{w(t)})$. If the value is smaller than a significance threshold τ, an unexpected event occurred. The values of $p(V \geq N_{w(t)})$ can be computed by $\bar{\Phi}((N_{w(t)} - \mu_{w,t}) \cdot \sigma_{w,t}^{-1})$ using the complementary cumulative distribution function $\bar{\Phi}(x) = 1 - \Phi(x)$. Overall the set of single word events at time t is defined by

$$\mathcal{W}_{1,t} = \{w \in \mathcal{W} \mid \bar{\Phi}((N_w(t) - \mu_{w,t}) \cdot \sigma_{w,t}^{-1}) \leq \tau\} \quad (3)$$

Finding multi-word events. As discussed in the introduction, single word events do not necessarily capture all important information. The actual event might be better described by a set of words. To deal with this challenge, we extend our previous definitions to handle word combinations representing unusual events, called multi-word events.

Here, an important aspect is that we do not only require the word combination to be unexpected, we also expect the parts of a multi-word to be an event. That is, each subset of the word set represents an event. This principle ensures to avoid words which are only included due to other words but which itself are not interesting. Formally we define the set of multi-words covering k words at time t as:

$$\mathcal{W}_{k,t} = \{W \subseteq \mathcal{W} \mid |W| = k \wedge \forall U \subseteq W : \bar{\Phi}((N_U(t) - \mu_{U,t}) \cdot \sigma_{U,t}^{-1}) \leq \tau\} \quad (4)$$

Take note that since we analyze Twitter data where each tweet consists of less than 140 characters, values for k are usually small.

It is worth mentioning that our method does not corresponds to a simple n-gram analysis, where pairs of consecutive words are considered. For short text data, such an approach would be too limited. In our case, multi-word events stem from arbitrary localities in the tweets. Also, unlike to existing work, the multi-word events are not simply constructed based on a post-processing step (e.g. using co-occurrence of single words). Multi-word events correspond to statistically significant deviations fulfilled by their contributing members.

3) Detecting non-redundant events: As a result of the previous step we have a set of candidate events for different levels of k. We denote with $\mathcal{W}_t$ the union of all events: $\mathcal{W}_t := \bigcup_{k \geq 1} \mathcal{W}_{k,t}$

The disadvantage becomes quickly apparent: By introducing multi-word events, the final result may contain redundant information. Various events may represent almost the same information. A large result set can quickly overwhelm the user and does not help in understanding the data. As a solution, we extract a subset of only non-redundant events, providing a succinct representation of the whole set $\mathcal{W}_t$.

Our goal is to select the smallest amount of events such that all other events can be explained. Here, we argue that two events very likely refer to the same event, if they use similar words or are contained in similar tweets. Thus, we define the similarity $\phi : \mathcal{W}_t \times \mathcal{W}_t \to \mathbb{R}$ between pairs of events as

$$\phi(W_1, W_2) := \max\{\frac{|W_1 \cap W_2|}{|W_1 \cup W_2|}, \frac{|O_{W_1}(t) \cap O_{W_2}(t)|}{|O_{W_1}(t) \cup O_{W_2}(t)|}\}$$

If the similarity is high, one of the events can be removed as redundant. Thus, overall, given ϕ our goal is to select a small number of events such that all other events are within a certain λ-neighborhood. Formally, the final set of non-redundant events is given by the following constrained optimization problem

$$\mathcal{W}_t^* := \arg\min_{\mathcal{W}' \subseteq \mathcal{W}_t} |\mathcal{W}'| \text{ subject to } \forall e \in \mathcal{W}_t : [e \in \mathcal{W}' \vee [\exists e' \in \mathcal{W}' : \phi(e, e') \geq \lambda]] \quad (5)$$

Overall, as a result of our model, the user gets only a small number of events covering all important information and being informative due to the multi-words.

B. Algorithmic Aspects

In the following, we discuss algorithmic aspects of our model, i.e. how to efficiently solve the above proposed technique. We focus on the two crucial parts: detecting the multi-word events and finding the non-redundant events.

1) Multi-word event detection: Naively solving Eq. 4 is far too expensive. The number of potential multi-words grows exponentially in the number of overall words. Furthermore, for each event, we have to test any of its subsets for interestingness, again being exponential in the size.

To solve this challenge, we exploit a principle well-known in the area of itemset mining: the apriori algorithm [1]. For this purpose, first note that Equation 4 can equivalently be written in a recursive fashion as

$$\mathcal{W}_{k,t} = \{W \subseteq \mathcal{W} \mid |W| = k \wedge \bar{\Phi}((N_W(t) - \mu_{W,t}) \cdot \sigma_{W,t}^{-1}) \leq \tau \wedge \forall x \in W : W \backslash \{x\} \in \mathcal{W}_{k-1,t}\} \quad (6)$$

Due to the recursive definition, if for each $x \in W$ it holds $W \backslash \{x\} \in \mathcal{W}_{k-1,t}$, then any subsets $U \subset W$ passes the significance test. The case $U = W$ is explicitly captured in Eq. 6. Thus, Eq. 4 and Eq. 6 are equivalent.

Using Eq. 6 it becomes apparent that candidates for multi-word events of size k can be constructed by extending events of size $k-1$. Only if a multi-word can not be constructed in this way, it can't fulfill Eq. 6. More precisely, we can exploit the following enhanced processing scheme [1]: Without loss of generality, assume that the words in each event are in lexicographical order. Given $\mathcal{W}_{k-1,t}$, to generate candidates $\mathcal{C}_k$ for $\mathcal{W}_{k,t}$, we first merge pairs of events whose first $k-2$ words are in common, leading to new events W of size k. Since we assumed the words are ordered lexicographically, we do not have to check for a duplicate generation of W due to word permutations. Given W, we check whether each of its subsets of size $k-1$ is contained in $\mathcal{W}_{k-1,t}$. If this is true,

Algorithm 1 Multi-Word Event Detection

$\mathcal{W}_t \leftarrow \emptyset$ // final result
compute $\mathcal{W}_{1,t}$ // Eq. 3
$k \leftarrow 1$
while $\mathcal{W}_{k,t} \neq \emptyset$ **do**
    generated candidates $\mathcal{C}_{k+1}$ based on $\mathcal{W}_{k,t}$
    $\mathcal{W}_{k+1,t} \leftarrow \{W \in \mathcal{C}_{k+1} \mid \bar{\Phi}((N_W(t) - \mu_{W,t}) \cdot \sigma_{W,t}^{-1}) \leq \tau\}$
    $\mathcal{W}_t \leftarrow \mathcal{W}_t \cup \mathcal{W}_{k+1,t}$
    $k \leftarrow k + 1$

W can potentially be an event and it is, thus, added to the candidate set $\mathcal{C}_k$ for further investigation. Thus, overall, only for a small number of multi-words $\mathcal{C}_k$ we eventually have to compute the significance test. Each set W passing the test will be added to $\mathcal{W}_{k,t}$.

Algorithm 1 shows the overall processing scheme: We first determine the single word events (k=1) according to Eq. 3. The algorithm then continues iteratively by extending events of size k. Candidate multi-word events of size $k+1$ are constructed based on the previous explanation, exploiting the fact that words are ordered. Then, for each candidate the significance test is performed and successful events are recorded. The algorithm continues until no further events can be detected or a specific maximal length has been obtained. Overall, the apriori principles allow us to efficiently detect multi-word events.

2) Non-redundant event detection: Next, we tackle the challenge of finding the non-redundant events. Solving Eq. 5 exactly is a challenging task since it is NP-hard.

Corollary 1: The result of Eq. 5 corresponds to a minimum dominating set on the graph $G = (V, E)$ with $V = \mathcal{W}_t$ and $E = \{(e, e') \in \mathcal{W}_t \times \mathcal{W}_t \mid \phi(e, e') \geq \lambda\}$

Proof: It is easy to see that two events explain each other when there exists an edge in G, and vice versa. A subset S of V is a minimum dominating set in G if (i) each node not in S has at least one neighbor in S and (ii) there is no smaller set S' that fulfills the first property. Property (i) is equivalent to the constraint that all events need to be explained, while property (ii) represents the minimization of the number of events. ■

Finding the minimum dominating set is infeasible since it is an NP-complete problem – and so is solving Eq. 5. However, due to the relation to the dominating set problem we are able to exploit well-established approximation algorithms. In particular, by performing a greedy approximation as shown in Algorithm 2, we obtain an optimal approximation ratio of $((1 - o(1)) \cdot \ln \Delta)$, where Δ denotes the maximal number of events another event explains. That is, the obtained solution is at most $((1-o(1)) \cdot \ln \Delta)$ times larger than the optimal solution.

Algorithm 2 allows us to find a non-redundant set very efficiently: To find a small event set $\mathcal{W}_t^*$, each event needs to explain as many unexplained events as possible. Let Ψ denotes the number of events which have not been explained so far, in each iteration of the algorithm we pick the event W explaining most of them. Formally, we denote with

$$\Lambda(W, \Psi) = \{W' \in \Psi \mid \phi(W, W') \geq \lambda\}| \qquad (7)$$

the (so far unexplained) events which can be explained by the event W. Picking the best W and updating the sets $\mathcal{W}_t^*$ and Ψ, the algorithm continues until all events are explained as required by Eq. 5.

Overall, combining the idea of the apriori algorithm with fast approximations of the dominating set allows us to detect the set of non-redundant events very efficiently.

Algorithm 2 Non-Redundant Event Detection

$\mathcal{W}_t^* \leftarrow \emptyset$ // final result
$\Psi \leftarrow \mathcal{W}_t$ // not explained events; all at the beginning
while $\Psi \neq \emptyset$ **do**
    $W = \arg\max_{W \in \mathcal{W}_t} |\Lambda(W, \Psi)|$ // best explaining event
    $\mathcal{W}_t^* \leftarrow \mathcal{W}_t^* \cup \{W\}$
    $\Psi \leftarrow \Psi - \Lambda(W, \Psi)$ // update not explained events

IV. Experimental Analysis

In this section, we empirically analyze our method and discuss some of its findings.

a) Datasets: We applied our method to over 24 million tweets which were collected from April 9th to April 21st, 2013 with a geographic boundary box around the Continental United States: (-124.7625, -66.9326) to (24.5210, 49.3845).

The collected Tweets consist of user ids uid, Tweets T and timepoints t at which the tweets were sent. Additionally, we use the given geographic location $l = (latitude, longitude)$ to estimate states s from where the tweets originate.

We preprocess each tweet in the dataset to facilitate the analysis that we carry out in this work. The preprocessing steps are as follows: First, we perform stopword removal using a stopword list we adapted for Twitter data that contains Twitter specific words such as "RT", "lol", etc. In addition to stopwords removal, we delete special characters and rare words (words that occur fewer than 30 times in the entire dataset). Second, we applied stemming to reduce inflected words to their word stem using the Porter stemming algorithm [7].

b) Discoveries: In the following, we demonstrate the application of our method by illustrating some of our interesting discoveries. Since our data collection methodology yields a rich collection of geographic information, where each tweet is geotagged, we can use this information to study the location patterns of U.S. Twitter users. For that reason the resulting data are grouped by state and sorted by time hourly. Since it is not feasible to enumerate all the real-life events happened in the collected time interval in all USA, we only focus on topics with more contents than small talk about celebrities.

Our first example in Figure 1 shows a detected multi-word event which is clearly associated with the American comedian called "Jonathan Winter" who died on April 11th, 2013. The figure depicts the frequency of the multi-word event over time

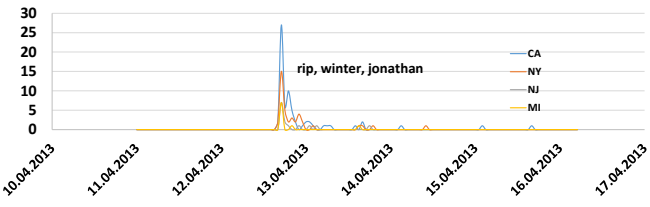

Fig. 1. Frequency of tweets to the detected event: *Johnathan Winter died*.

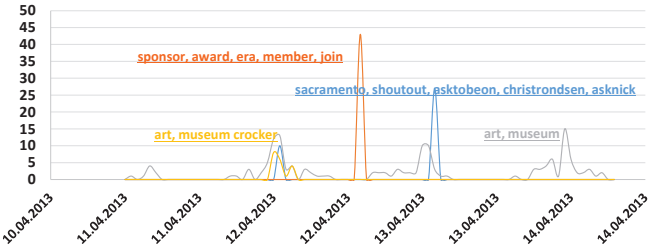

Fig. 2. Frequency of tweets for multi-word events referring to an art award at the Crocker Art Museum in Sacramento, California.

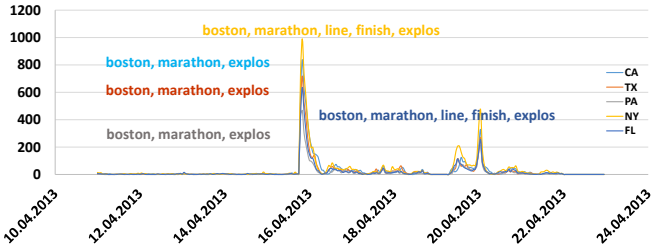

Fig. 4. Frequency of tweets for events referring to the Boston Marathon Bombing for different states of the United States.

for different states. Interesting is the strong abrupt ascent of the words' frequencies a day after his death specially in California (CA). While the event has also been detected in other states, its topicality last longer in CA. This effect could be based on the reason that the comedian lived and died in CA. This first example shows the strong benefit of using multi-word events: based on the combination of "rip", "winter", and "jonathan" an easy understanding of the result is possible. Using only single word events would not capture this correlation.

In our next example in Figure 2, we show multiple events detected by our algorithm and their frequency over time. While the gray line represents a relatively constant usage of the words "art" and "museum" over time, the peaks in yellow, blue and orange lines indicate abrupt and specific events that address an art award at the Crocker Art Museum in Sacramento, California at the 12th of April. Noticeable is again the enhanced interpretation by using multi-words. Furthermore, by focusing on only non-redundant events, the "art,museum" pattern is replaced by more 'specific' instances of the event. In contrast to our previous example, these events about the art celebration arouse only interest of a local population in CA.

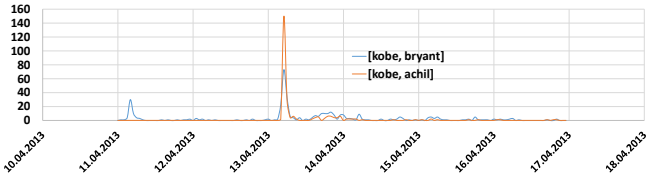

Fig. 3. Event referring to the Achilles tendon rupture of Kobe Bryant.

Another interesting discovery illustrated in Figure 3 shows tweets from California about Kobe Bryant, an American basketball Player whose Achilles tendon was torn on 13th of April 2013. Comparing the usage of the multi-words "Kobe Bryant" (blue line) and "Kobe Achil" (orange line) over time shows that some fluctuation in the blue line continue after the main event, while the orange line converges to zero much earlier. This example again shows how the idea of significant testing successfully recognize at which time the usage of certain words is increased suspiciously compared to the previous time points.

c) Lifespan of events: All events described in the previous part have a very short lifespan, which could be considered as a feature for measuring the event's significance. In the following we deal with the question of how long the Twittersphere maintains interest in a major event?

Figure 4 shows an event detected by our algorithm that emerges for a long time period and in multiple states (for illustration purposes we plot 6 states only). Besides, also the number of tweets approaching it is very high. The detected event illustrates the tragedy of the Boston Marathon Bombing on 15th of April 2013. In this tragedy, two bombs exploded near the finish line of the Boston Marathon on Boylston Street. Three days later an MIT police officer was shot in an exchange of fire with the suspects; one suspect was killed, the other was wounded. In the following hours a massive manhunt was conducted in Watertown, a city next to Cambridge, MA, resulting in capturing the secondsuspect and ending the series of events related to the bombing.

As shown in the figure, in three regular intervals, we see a pattern of heavily-peaked spikes and a period of very fast, massive adoption followed by a rapid decrease in activity. While the first spike is the strongest, it returns to normality within 15 hours. The second and third spike are connected. Between the three main spikes, there are continuous fluctuations which refer to the time between the exploded bombs and the capture of suspect. These three intervals referring to the same event are captured by our method as the figure clearly illustrates. The associated multi-word events again lead to an easy interpretation. Interestingly, in some states the event is very precisely described adding the terms "finish,line" to the general event "boston,marathon,explos".

A comparison between the frequency of tweets from different states in Figure 4 shows that the frequency is highly correlated to the states' population. Notable is the exception of "CA" as the state with the highest population followed by "NY"; for the detected event, however, NY shows a stronger signal than CA which might be caused by the fact that the Bombing tragedy took place on the east coast.

Another event with a long lifespan detected by our method is associated with the multi-word "job, tweetmyjob". Its temporal pattern is shown in Figure 5. The event happened between 4:00 PM and 5:00 PM in every state in the US, repeatedly every day. A more precise look at the result points out that the tweets are related to a spam of job advertisements.

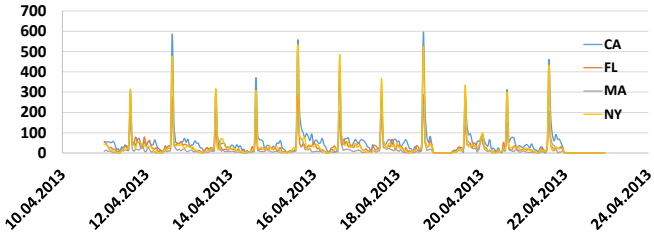

Fig. 5. Multiple occurrences of an event referring to spam of job advertisements. The event is described by the words "job" and "tweetmyjob".

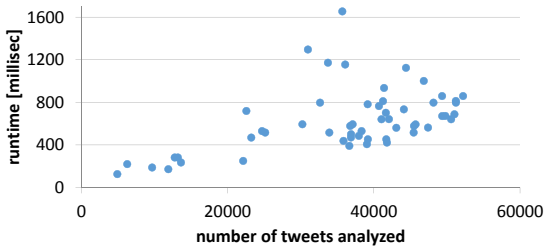

Fig. 6. Runtime analysis of our method

d) Analysis of Non-Redundancy: In the following we focus on the results of the non-redundant multi-word event detection and we compare these with the results where redundancy has *not* been removed. Table I lists all (redundant and non-redundant) multi-word events that took place on 19th of April 2013, tweeted between 10:00 AM and 11:00 AM. The reported events are about the suspects from the Boston Bombing tragedy. As explained in Section III our algorithm determines first the single word events continued by iteratively extending events of higher size until no further events can be detected or a maximal length has been reached. In bold font we highlight the *non-redundant* events extracted by our optimization principle. It is obvious that the boldface printed result provides a clear and compact summary of the long listed results. Since the not selected events cover similar words and/or tweets, this smaller subset still covers the core information of the event. Overall, the redundancy removal ensures a quick and easy interpretation of the detected events.

e) Runtime: Last, we analyze the runtime of our algorithm. The runtime is primarily effected by the number of tweets. To analyze this effect, we consider the runtime of our algorithm for tweets from California. All experiments were conducted on commodity hardware with 3 GHz CPU's and 4 GB main memory. The result is depicted in Figure 6.

The y-axis shows the runtime of our Algorithm for analyzing tweets of one hour of data. The x-axis shows the number of these tweets. The maximum runtime is only around 1653 ms, showing its high efficiency. Notably, the number of Tweets sent at this time is not the highest. Inspecting the result more carefully, it becomes apparent that at this time point the number of detected multi-word events is very high. Thus, besides the number of tweets also the number of events affects the overall runtime.

V. Discussion

An advantage of our method is its general applicability to arbitrary sets $\mathcal{W}$, i.e. it is agnostic to the used dictionary. While in our current experimental study, we instantiated $\mathcal{W}$ to be the set of preprocessed words only, it is straightforward to include further information. For example, $\mathcal{W}$ might be enriched by specifically distinguishing between usual words and Twitter hashtags. This way, the users' highlighting of terms using hashtags can be incorporated into the multi-words events. Note that our technique is still more expressive than using hashtags alone since the multi-word events allow, for example, to combine hashtags with usual words; thus, providing a more detailed description. In general, the set $\mathcal{W}$ can be extended to capture different (event) meta-data including, for example, location information or extracted entity information.

length of multi-word event	detected events; non-redundant events in bold font
k=1	[russia], [chechnya], [fridai], [boston], [marathon], [suspect], [bomber], [safe], [slept], [manhunt], [watertown], [massachusett], [tgif], [bomb], [polic], [loos]
k=2	**[suspect, chechnya]**, **[boston, marathon]**, **[boston, watertown]**, [boston, suspect], [boston, bomb], **[boston, safe]**, [marathon, suspect], [bomb, suspect]
k=3	**[boston, bomb, suspect]**

TABLE I. Our optimization principle successfully reduces a large set of events to a more succinct overview (bold font).

VI. Conclusion

We have introduced a technique for finding non-redundant multi-word events in Twitter. Our statistical model spots events by identifying significant frequency deviations. To provide rich information, events are composed based on multiple co-occurring words present at various localities of the tweet. Simultaneously, we ensure interpretative results by providing a succinct summary of all events based on an optimization method that confines the result to only non-redundant events. We analyzed our method on a large set of twitter messages and we highlighted interesting findings detected by our technique.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[2] J. Benhardus and J. Kalita. Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139, 2013.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[4] C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In *WWW*, pages 675–684. ACM, 2011.

[5] M. Cataldi, L. D. Caro, and C. Schifanella. Personalized emerging topic detection based on a term aging model. *ACM TIST*, 5(1):7, 2013.

[6] Y. Chung, S. Rabe-Hesketh, V. Dorie, A. Gelman, and J. Liu. A nondegenerate penalized likelihood estimator for variance parameters in multilevel models. *Psychometrika*, 78(4):685–709, 2013.

[7] K. S. Jones. *Readings in information retrieval*. Morgan Kaufmann, 1997.

[8] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *CIKM*, pages 155–164. ACM, 2012.

[9] M. Mendoza, B. Poblete, and C. Castillo. Twitter under crisis: Can we trust what we rt? In *Workshop on Social Media Analytics*, pages 71–79. ACM, 2010.

[10] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *ACM SIGIR*, pages 472–479, 2005.

[11] K. P. Murphy. Conjugate bayesian analysis of the gaussian distribution. Technical report, University of British Columbia, 2007.

[12] M. Naaman, J. Boase, and C.-H. Lai. Is it really about me?: message content in social awareness streams. In *ACM conference on Computer supported cooperative work*, pages 189–192. ACM, 2010.

[13] T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *SIGIR*, pages 103–110, 2007.

[14] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from twitter. In *SIGKDD*, pages 1104–1112, 2012.

[15] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860. ACM, 2010.

[16] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *SIGSPATIAL/GIS*, pages 42–51. ACM, 2009.

[17] H. Tsukayama. Twitter turns 7: Users send over 400 million tweets per day, March 2013.

3.4 Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model

Peer-reviewed Conference Paper

Autors **Nikou Günnemann**, Jürgen Pfeffer

In *Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 210-222, 2017, [22].*

Abstract Connectivity and automation are evermore part of today's cars. To provide automation, many gauges are integrated in cars to collect physical readings. In the automobile industry, the gathered multiple datasets can be used to predict whether a car repair is needed soon. This information gives drivers and retailers helpful information to take action early. However, prediction in real use cases shows new challenges: misclassified instances have not equal but different costs. For example, incurred costs for not predicting a necessarily needed tire change are usually higher than predicting a tire change even though the car could still drive thousands of kilometers.

To tackle this problem, we adapt the standard logistic loss function to a new example-dependent cost sensitive prediction model in four different ways. While the first to functions focus on increasing the average loss according to the assigned costs, the last two functions focus on increasing the fraction between the false and correct classified loss. Overall, our model allows different costs of misclassified instances, thus, enabling to give, e.g., higher cost to rarely occurring events such as anomalies or errors. And the finally learned classifier leads to a decision boundary aiming to obtain overall less misclassification cost.

The experiments show that our method consistently outperforms the state-of-the-art in example-dependent cost-sensitive logistic regression on various datasets. For example, applying our methods to vehicle data from a large European car manufacturer, we show cost savings of about 10%.

Contribution of thesis author: Algorithmic design, development and implementation, experimental design and analysis, manuscript writing, revision and editing.

Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model

Nikou Günnemann and Jürgen Pfeffer

Technical University of Munich
{Nikou.Guennemann, Juergen.Pfeffer}@tum.de

Abstract. Connectivity and automation are evermore part of today's cars. To provide automation, many gauges are integrated in cars to collect physical readings. In the automobile industry, the gathered multiple datasets can be used to predict whether a car repair is needed soon. This information gives drivers and retailers helpful information to take action early. However, prediction in real use cases shows new challenges: misclassified instances have not equal but different costs. For example, incurred costs for not predicting a necessarily needed tire change are usually higher than predicting a tire change even though the car could still drive thousands of kilometers. To tackle this problem, we introduce a new *example-dependent cost sensitive prediction model* extending the well-established idea of logistic regression. Our model allows different costs of misclassified instances and obtains prediction results leading to overall less cost. Our method consistently outperforms the state-of-the-art in example-dependent cost-sensitive logistic regression on various datasets. Applying our methods to vehicle data from a large European car manufacturer, we show cost savings of about 10%.

1 Introduction

Automation has become of prime importance to improve the quality of our life. An example from the vehicle industry, where predictive maintenance [18] looms large, is to predict whether the tires of a car need to be changed soon. Goals are (i) providing customers services with less latency for tire change, and (ii) forecasting tire delivery in correct number for all customers with a tire change need.

Given historical data, such potential 'malfunctions' (required tire change) can be predicted based on binary classification algorithms like logistic regression, support vector machines, ARIMA models or neural networks etc. [4–6, 2]. Such approaches, however, often do not meet the real world use cases since intuitively they try to minimize the so called zero-one loss with the assumption that all misclassified instances have equal cost. Meaning correct classifications lead to a cost of zero and misclassification gets a cost of one [7].

In many applications, however, the costs for misclassified instances might vary significantly from one instance to the other. Predicting tire change is ranked among these applications. The cost associated with, e.g., an incorrect early tire

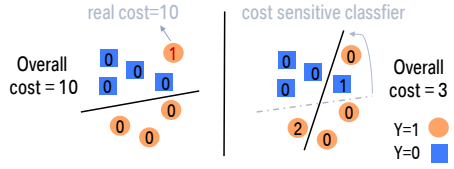

Fig. 1. Left: result of a cost-insensitive classifier; right: result of a cost-sensitive classifier with smaller overall cost

	True Pos. ($y_i = 1$)	True Neg. ($y_i = 0$)
Predicted Pos. ($\hat{y}_i = 1$)	0	${c_i}^{FP}$
Predicted Neg. ($\hat{y}_i = 0$)	${c_i}^{FN}$	0

Fig. 2. Cost matrix for cost-sensitive classification

change prediction is smaller than the expected cost for not predicting an imminent tire change at all, which could even cause customer dissatisfaction. Also, instances where the tire change has been predicted too early might show different costs. In some cases the tires might actually be used for further 20,000 km compared to only 1,000 km. Hence, in practical applications different misclassified instances can cause different costs. Since standard binary classifiers are not suited for such scenarios, *example-dependent cost-sensitive classification* [8] has been introduced, considering the different costs of instances during learning.

Technically, to distinguish between different misclassified instances, a predefined cost value can be assigned to each instance in the dataset. Figure 2 shows the cost for an instance according to their actual class vs. their predicted class. When the instance is a false positive, we have cost of ${c_i}^{FP}$, if it is a false negative, the cost is ${c_i}^{FN}$. If the instance is correctly classified (i.e. true positive or true negative), we assign a cost of zero. Note that each instance i might get different cost (indicated by the index i). Having assigned costs to each instance i, a possible way to estimate the overall misclassification cost is

$$Cost = \sum_{i}^{m} y_i \cdot (1 - \hat{y}_i) \cdot {c_i}^{FN} + (1 - y_i) \cdot \hat{y}_i \cdot {c_i}^{FP} \tag{1}$$

where $y_i \in \{0, 1\}$ is the observed and $\hat{y}_i \in \{0, 1\}$ the predicted label of instance i, with m as the number of overall instances [4]. Accordingly, instead of considering each instance equally, our goal is to train a classifier that takes the overall misclassification cost into account. The benefit of such an approach is shown in Figure 1. On the left, the cost-insensitive classifier misclassified only one instance but its associated cost is very high (10). On the right, the potential result of a cost-sensitive classifier is shown. Although now two instances are classified false, they have only very low cost of 1 and 2; the previously misclassified instance with cost of 10 is classified correctly. Thus, the overall misclassification cost is only 3. Based on this motivation, in this work, we focus on the well established classification model of logistic regression – and we extend it by the principle of example-dependent cost sensitive learning. The contributions of this paper are:

– We propose an enhanced binary classification model that includes the individual costs of instances while fitting a model to the training set – thus, costs are not simply used in a post-processing step but during training.

– We propose four different variants of our model each extending the sound principle of logistic regression but considering different properties.
– We perform experiments on multiple real-world datasets including data from a leading European car manufacturer showing that our methods successfully lower misclassification costs.

2 Background

In a binary classification problem, input vectors $X = \{x_1, ..., x_m\}$ with $x_i \in \mathbb{R}^d$, and class labels $Y = \{y_1, ..., y_m\}$ with $y_i \in \{0, 1\}$ are given. Here, x_i is the i-th instance described by d features, and $y_i = 1$ represents the class of instances with a certain issue (tire change required) and $y_i = 0$ the opposite. In our scenario, each instance i is associate with a certain predefined cost c_i.[1] The higher the cost, the worse is a potential misclassification of this instance. Our goal is to fit a model on observed data X to predict Y denoted by $\hat{Y}$ at best. More precisely, our aim is to find a model that leads to small overall misclassification cost.

2.1 Logistic Regression and Important Properties

Logistic regression treats the binary classification problem from a probabilistic perspective. Given the instance x, the probability of the occurrence of an issue (i.e. $y = 1$) is denoted by $p(y = 1 \mid x)$, and $p(y = 0 \mid x) = 1 - p(y = 1 \mid x)$ respectively for $y = 0$. Here, $p(y = 1 \mid x)$ is defined as the sigmoid function, known as logit:

$$p(y = 1 \mid x) = f(g(x, \beta)) = \frac{1}{1 + e^{-g(x,\beta)}} \tag{2}$$

where $0 \leq f(g(x, \beta)) \leq 1$ and $g(x, \beta) = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot x_j$ is a linear expression of Eq. 2 including the explanatory features and the regression coefficients β. Considering the sigmoid equation, the question is how to estimate β in $g(x, \beta)$ to make $f(g(x, \beta)) = \hat{y}$ close to y? To formalize this, and assuming that the m samples in the data are independent, we can write $p(Y|X; \beta)$ as a product, leading to the following overall Likelihood function:

$$L(Y, X, \beta) = \prod_{i=1}^{m} f(g(x_i, \beta))^{y_i} \cdot (1 - f(g(x_i, \beta)))^{1-y_i} \tag{3}$$

The β in logistic regression can be obtained by maximizing Eq. 3, i.e. it corresponds to the maximum likelihood estimate. Obviously, instead of maximizing $L(Y, X, \beta)$, we can equivalently minimize the negative log likelihood given by

$$l(Y, X, \beta) = \sum_{i=1}^{m} y_i \cdot (-\log f(g(x_i, \beta))) + (1 - y_i) \cdot (-\log(1 - f(g(x_i, \beta)))) \tag{4}$$

[1] Note that we do not have to explicitly distinguish between ${c_i}^{FP}$ and ${c_i}^{FN}$. If $y_i = 0$, then ${c_i}^{FP} = c_i$, if $y_i = 1$, then ${c_i}^{FN} = c_i$. For a single instance, ${c_i}^{FP}$ and ${c_i}^{FN}$ can never occur together.

which is the logistic loss function. Fig. 3 shows the logistic loss function for $y_i = 1$ (e.g. tire change)[2]: $y_i \cdot (-\log f(g(x_i, \beta)))$. Clearly, if $f(g(x_i, \beta)) = 1$ the prediction is correct and we have zero loss. For $f \to 0$, in contrast, the loss will increase. Thus, minimizing the loss means lowering the prediction error.

Loss of Correctly Classified Instances: In logistic regression the return values of the sigmoid function are between 0 and 1. Therefore, we have no deterministic decision which samples are classified correctly and which are classified wrong. To turn the predicted probabilities into binary responses, a threshold is used. Based on the probabilistic view and as default in literature, we choose 0.5 as threshold. That is, if $f(g(x_i, \beta)) \geq 0.5$, the predicted class is 1, otherwise 0. The resulting observation is that *even for correctly classified instances the logistic loss is not zero*. This becomes obvious in Fig. 3: e.g. an instance with $f(g(x_i, \beta)) = 0.9$ has a loss of 0.05 even if it is correctly classified.

Assuming the correctly classified instances get a probability $f(g(x_i, \beta))$ uniformly random between 0.5 and 1, then the average loss of a correctly classified instance is proportional to $T_{log} := \int_{0.5}^{1} y_i \cdot (-\log f(g(x_i, \beta))) \, \mathrm{d}f \approx 0.15$.[3] Here, T_{log} can also be illustrated as the area under the 'logistic loss'-curve as shown in Fig. 3. Likewise, the incorrectly classified instances get an average loss proportional to $F_{log} := \int_{0}^{0.5} y_i \cdot (-\log f(g(x_i, \beta))) \, \mathrm{d}f \approx 0.85$ where F_{log} represents the area from 0 to 0.5. Also note that $T_{log} + F_{log} = 1$. That is, the average loss assigned to an instance (independent if correctly or incorrectly classified) is 1. Obviously $F_{log} > T_{log}$, which means that a correct prediction actually leads to smaller loss. However, the two loss terms F_{log} and T_{log} are constant and identical for each instance. That is, the standard logistic loss function does not distinguish between different losses caused by different instances with different costs.

3 Example Dependent Cost-Sensitive Logistic Regression

The above discussion leads to the core motivation of our paper: How can we adapt the logistic loss function in a sound way, so that different samples having different costs are treated differently? How can we define a loss function to make sure that instances with higher costs are more likely to be predicted correctly?

General Framework To answer these questions, we adapt the standard logistic loss function to a cost sensitive one in four different ways. The general framework we explore in these versions is to minimize the loss function $l(Y, X, \beta)$ defined as

$$\sum_{i=1}^{m} a_i \cdot y_i \cdot (-\log f(g(x_i, \beta))^{b_i}) + a_i \cdot (1 - y_i) \cdot (-\log(1 - f(g(x_i, \beta)))^{b_i}) \quad (5)$$

where a_i and b_i depend on c_i. That is, $a_i = a(c_i)$ and $b_i = b(c_i)$ based on functions $a : \mathbb{R}^+ \to \mathbb{R}^+$ and $b : \mathbb{R}^+ \to \mathbb{R}^+$. As shown next, given different

[2] The case $y_i = 0$ is equivalent; only mirrored. W.l.o.g. we consider in the following only $y_i = 1$.

[3] More precise, the average loss for correctly classified instances would be $2 \cdot T_{log}$.

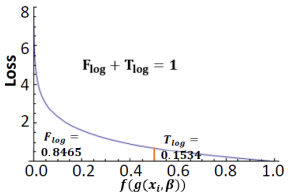

Fig. 3. Loss function of standard Logistic Regression for $y = 1$.

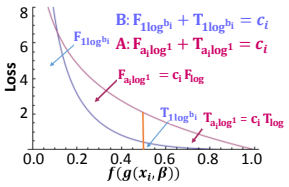

Fig. 4. Loss function for A & B. The loss ratio for B is smaller.

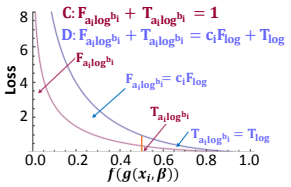

Fig. 5. Loss function for variant C & D. Both variants control the loss ratio.

choices of a and b, we realize different properties. For convenience, let us already introduce the following notation: $F_{a_i log^{b_i}} := \int_0^{0.5} a_i \cdot (y_i \cdot (-\log f(g(x_i,\beta)))^{b_i})\ \mathrm{d}f$ represents the average loss for misclassified instances and $T_{a_i log^{b_i}} := \int_{0.5}^{1} a_i \cdot (y_i \cdot (-\log f(g(x_i,\beta)))^{b_i})\ \mathrm{d}f$ is the average loss for correctly classified instances.

Variant A: Weighting the logistic loss function The first, simplest way is to weight the logistic loss function depending on the cost value c_i by setting the area under the curve – representing the average loss – equal to the cost.

$$\int_0^1 a_i \cdot (y_i \cdot (-\log f(g(x_i,\beta))))\ \mathrm{d}f \stackrel{!}{=} c_i \tag{6}$$

This way, instances with a higher cost will get a higher average loss value. Since in standard logistic regression the area is 1, it obviously holds that the weight factor a_i needs to be equal to c_i. The variable b_i is equal to 1. The purple curve in Figure 4 shows the plot for $c_i = 3$. Clearly, by weighting the loss function, we oversample the instances proportional to their costs, i.e. an instance with cost 2 is basically considered twice. But this solution has one drawback: By weighting the loss, not only the misclassification loss $F_{a_i log^1}$ but also the 'correct' loss $T_{a_i log^1}$ will be higher. *Correctly classified instances are penalized by this version, too.* In particular, the ratio between $F_{a_i log^1}$ and $T_{a_i log^1}$ does *not* change. Thus, for every instance a misclassification has always $\frac{F_{log}}{T_{log}} \approx 5.5$ higher loss than a correct classification. Thus, this solution might not well represent the intuition that the cost of *misclassification* will be higher.

Variant B: Logistic loss function to the power of *b* To avoid penalizing correctly classified instances, we exchange the weighting in Eq. 6 by an exponentiation of the logistic loss function to the power of b. That is, we increase the average loss from 1 to c_i by using the term b_i and keeping $a_i = 1$:

$$\int_0^1 y_i \cdot (-\log f(g(x_i,\beta)))^{b_i}\ \mathrm{d}f \stackrel{!}{=} c_i \tag{7}$$

Since Eq. 7 is equal to $\Gamma(b_i + 1)$, the solution for b_i given a specific c_i is equal to

$$b_i = \Gamma^{-1}(c_i) - 1 \tag{8}$$

Γ^{-1} is the inverse of the Gamma function Γ which can be computed numerically.

Fig. 4 shows the corresponding loss function by the blue curve with $c_i = 3$. While the loss area $T_{1log^{b_i}}$ is pressed downwards, the loss area of instances in $F_{1log^{b_i}}$ wins on more importance since instances with high costs are more important to be classified correct. Thus, not only the average loss increases for these instances but also the ratio between $F_{1log^{b_i}}$ and $T_{1log^{b_i}}$. A potential drawback is that the ratio $F_{1log^{b_i}}/T_{1log^{b_i}}$ is not controlled explicitly.

Variant C: Controlling the ratio - I We aim to control the ratio between the loss area of $F_{a_i log^{b_i}}$ and $T_{a_i log^{b_i}}$. That is, for an instance with cost c_i we want to ensure $\frac{F_{a_i log^{b_i}}}{T_{a_i log^{b_i}}} \stackrel{!}{=} \frac{F_{log}}{T_{log}} \cdot c_i$. The ratio between the loss of false and correct classification is c_i times higher than for an instance with cost 1. Simultaneously, the average loss of the instances should be independent of c_i. The motivation is that in average each instance is equally important, but for some of them the *misclassification* should be penalized stronger. That is, the area under the curve has to be equal to 1, meaning $F_{a_i log^{b_i}} + T_{a_i log^{b_i}} \stackrel{!}{=} 1$. This constraint implies that

$$a_i = \frac{1}{\Gamma(b_i + 1)} \tag{9}$$

The value of $b_i > 0$ can be computed numerically by solving (see Appendix)

$$\frac{\Gamma(b_i + 1)}{\Gamma(b_i + 1, 0.6931)} = 1 + \frac{T_{log}}{c_i \cdot F_{log}} \tag{10}$$

where $\Gamma(s, r)$ is the incomplete gamma function. The effect of this variant is shown in Fig. 5 as variant C, again for $c_i = 3$. Here, we have $F \approx 0.94$ and $T \approx 0.057$. Thus, the ratio is c_i times higher than in the standard case. Still the average loss is identical (i.e. equal to 1).

Variant D: Controlling the ratio - II In version C, we kept the average loss at 1 but increased the ratio between false and correct classification; thus, the area $T_{a_i log^{b_i}}$ needs to decrease. Accordingly, instances with a high costs will not only have higher *misclassification loss* but also lower *correct classification loss* compared to instances with low costs – which again, might not be intended since the costs for correct classification is constant. Therefore, we introduce our last version which (i) directly controls the ratio, and (ii) ensures that the correct classification loss stays constant. This idea can be transformed to $\frac{F_{a_i log^{b_i}}}{T_{a_i log^{b_i}}} \stackrel{!}{=} \frac{F_{log}}{T_{log}} \cdot c_i$ and $T_{a_i log^{b_i}} \stackrel{!}{=} T_{log}$. Solving this, we obtain for b_i the identical solution as in variant C; only the weighting a_i changes to

$$a_i = \frac{T_{log}}{\Gamma(b_i + 1) - \Gamma(b_i + 1, 0.6931)} \tag{11}$$

Indeed, what we observe is that for this variant we have $F_{a_i log^{b_i}} = c_i \cdot F_{log}$. Thus, we only increase the average loss for the misclassified instances by a factor of c_i.

variant	a_i	b_i	avg. loss	ratio F/T	T
LR	1	1	1	constant	T_{log}
A	c_i	1	c_i	constant	$c_i \cdot T_{log}$
B	1	Eq. 8	c_i	adaptive	adaptive
C	Eq. 9	Eq. 10	1	$\propto c_i$	adaptive
D	Eq. 11	Eq. 10	$T_{log} + c_i \cdot F_{log}$	$\propto c_i$	T_{log}

Fig. 6. Proposed variants and their properties

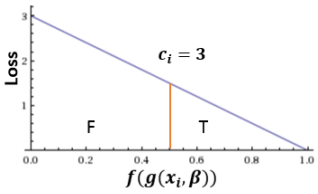

Fig. 7. Loss function used in [8].

This effect is shown in Figure 5 as variant D. As one can also observe, the area $F_{a_i log^{b_i}}$ in variant D is equal to the area $F_{a_i log^{b_i}}$ of variant A; in both variants the area increases by a factor of c_i compared to standard logistic regression. While in variant A, however, also the area $T_{a_i log^{b_i}}$ increases, it stays constant in variant D. Thus variant D better captures the increased costs for *misclassification*.

Summary & Algorithmic Solution Figure 6 summarizes our different variants. While variants A and B focus on increasing the average loss according to the costs, variants C and D focus on increasing the fraction between false and correct classification loss.

Our final goal is to find the parameter β that minimizes the loss function in Eq. 5: $\beta^* = \arg\min_\beta l(Y, X, \beta)$. For this purpose we exploit a gradient descent search. Starting from a random solution, we iteratively follow the steepest descent direction: $\beta^{t+1} \leftarrow \beta^t - \alpha \nabla l(\beta)$ where α is the learning rate.

4 Related Work

Various research papers are published with focus on cost sensitivity [1, 9, 8, 10–14]. Often, the main objective is predicting potential customers with financial obligation based on their existing financial experience. While [15, 12–14] use constant costs for misclassified instances, the authors in [10] propose a bayes minimum risk classifier including the financial costs of credit card fraud detection in order to have a cost sensitive detection system. Another interesting approach is introduced in [16], by presenting a taxonomy of cost-sensitive decision tree algorithm using the class-dependent cost. An extension of [16] with focus on example-dependent cost for decision trees is published by [17].

The only method similar to ours is [8], which proposes an example-dependent cost sensitive logistic regression. Here the loss function of logistic regression is changed to a cost sensitive one by integrating the cost as a factor into its calculation. A drawback of [8] is that the loss function is no longer a logarithmic function but linear. That is, for the case that correct classification has 0 cost, [8] uses $\frac{1}{m}\sum_i^m y_i(1 - f(g(x_i, \beta)))c_i + (1 - y_i)f(g(x_i, \beta))c_i$. Thus, the loss decreases linearly: starting from c_i to 0 (see Fig. 7). Using a linear loss function causes weak differentiation between false and correctly classified instances. The two areas marked by F and T in Figure 7 show this problem. As we will see in our experimental analysis, this principle will often perform worse than our technique.

5 Experimental Analysis

In this section we compare our four variants A-D with standard logistic regression LR and the competition model proposed in [8]. For this purpose, we test our designed models on the basis of three different datasets: (i) a vehicle dataset from a large European car manufacturer for predicting tire change service, (ii) the dataset breast cancer[4] to predict whether a patient is affected by breast cancer or not, and (iii) data from the 2011 Kaggle competition Give Me Some Credit[5] to predict whether a customer will experience financial distress in the next two years.

Our main goal is to achieve low overall misclassification cost (see. Eq. 1). Thus, a technique is successful if it obtains the lowest overall cost. As an evaluation measure we compute the savings of our techniques w.r.t. logistic regression $savings = \frac{Cost_{LR} - Cost_x}{Cost_{LR}}$ where $Cost_{LR}$ is the obtained misclassification cost (Eq. 1) based on the result of logistic regression and $Cost_x$ the cost based on the result of the technique x. In each scenario we used $\frac{2}{3}$ of the data for training our models and $\frac{1}{3}$ to evaluate them.

5.1 Tire Change Service

The vehicle dataset is a binary classification dataset containing 1,800 instances, each with 40 features. The features are indirectly influenced by tire wear and which, thus, indicate a resultant tire change. An example of such features could be acceleration.[6] Important to mention is that features resulting through, e.g. a sensor which directly measures the tire tread to asses a tire wear are not considered here. The target variable is whether a vehicle needs a tire change: yes=1 or no=0. Instances requiring a tire change account to ∼15% amount of the whole data. Each instance is assigned with a cost; the higher the cost value, the more urgently a tire change is needed. The degree of urgency was determined by the domain experts.

Figure 8 shows the results. Here, the threshold to cast the predicted probabilities to binary responses is set to 0.5. Generally all of our four versions obtain lower overall misclassification cost than traditional logistic regression. But the best savings are achieved by B, C, and D with a win of around 10%. Applying the competing variant from [8] shows even higher overall cost than our variants. As discussed, this is caused by the used non-discriminative loss function in their logistic regression model.

While a threshold of 0.5 is from a probabilistic view the correct one, it might, however, not lead to smallest misclassification cost. Thus, in Figure 9 we report the results for each method when individually using the threshold that leads to lowermost misclassification cost. These 'optimal' thresholds are given in Table

[4] https://goo.gl/U2Uwz2
[5] http://www.kaggle.com/c/GiveMeSomeCredit/
[6] Due to nondisclosure agreements we unfortunately can not provide more details on the dataset. The two other datasets studied in this work are publicly available.

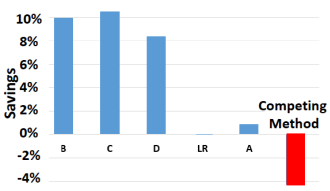

Fig. 8. Savings of the methods when threshold is 0.5. Our techniques significantly outperform logistic regression.

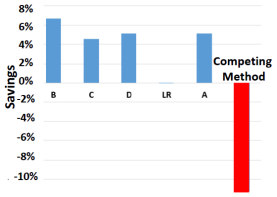

Fig. 9. Savings of the methods with variable threshold with minimal misclassification cost. Our techniques perform better

1. Note that in practice such a tuning is not possible since we a-priori do not know the true class of an instance.

The results show that even in a scenario of complete knowledge our techniques significantly outperform logistic regression. Example dependent cost improves the ability to make less failure than the standard process. Also, the results from Table 1 are very close to $t = 0.5$. Thus, applying our models for $t = 0.5$ without tuning the threshold in practice would still obtain very good performance.

LR	A	B	C	D	Competing
0.4542	0.4509	0.4950	0.5060	0.4775	0.5001

Table 1. Variable thresholds leading to minimal costs in vehicle datasets

5.2 Breast Cancer

The Wisconsin breast cancer dataset is a binary classification dataset, e.g., available in scikit-learn. The total number of instances is 569, each with 30 attributes. 212 ($\sim 37.2\%$) of records are malignant denoted by 1 and 357 ($\sim 62.8\%$) records are benign presented by 0. Since the instances are not presented by different costs we randomly assigned each instance a cost between 1 to 5 to guaranty the fairness. To obtain reliable results, we generated $n = 10$ such datasets.

Table 2 shows the number of correctly classified instances and false classified ones (on avg. on the test data). Also, the average corresponding misclassification cost of the false classified instances are presented by the column *cost*. The left part of the table presents results for $t = 0.5$, the right part shows the results for the 'optimal'/tuned thresholds.

For $t = 0.5$, versions B and D return the lowest overall cost with a small number of false classified malignant instances. Surprisingly, not only the cost is lower in our variants, but also the classification accuracy increases. The same behavior can be seen for the variable threshold. In comparison to the competing variants LR and [8] our results are much better.

Figure 10 and 11 show the relative savings of the techniques w.r.t. logistic regression. Since we applied the algorithms 10 times based on different randomly assigned costs, different savings are observed. In Figure 10, the bars show the mean savings for $t = 0.5$ achieved by each algorithm; the black lines represent

Method, t	Incorrect	Correct	avg. cost	Method, t	Incorrect	Correct	avg. cost
LR, 0.5	≈ 17	≈ 171	52.61	0.39	≈ 14	≈ 174	38.07
A, 0.5	≈ 13	≈ 175	40.29	0.52	≈ 11	≈ 177	36.23
B, 0.5	≈ **12**	≈ **176**	**36.96**	0.4857	≈ **11**	≈ **177**	**33.37**
C, 0.5	≈ 14	≈ 174	43.72	0.496	≈ 13	≈ 175	38.61
D, 0.5	≈ **12**	≈ **176**	**37.21**	0.496	≈ **11**	≈ **177**	**34.02**
Competing, 0.5	≈ 87	≈ 101	282.8	0.499	≈ 38	≈ 150	115.49

Table 2. Wisconsin breast cancer dataset. Left: threshold $t = 0.5$; right: variable thresholds with overall minimum misclassification cost.

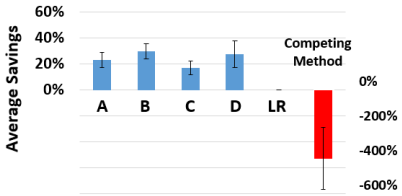

Fig. 10. Average savings when threshold t=0.5. Average over 10 runs.

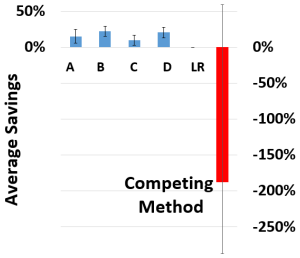

Fig. 11. Average savings when selecting variable t with minimal misclassification cost (see. Table 2). Average over 10 runs.

the standard deviation over the 10 runs. In average, D as well as B save at most whereas the model in [8] cause even more loss than standard LR. *As shown by the black lines, these results are significant.* While all our 4 versions return very similar good results, [8] shows bad performance and strong fluctuation. A similar behavior can be considered in Figure 11 for variable thresholds.

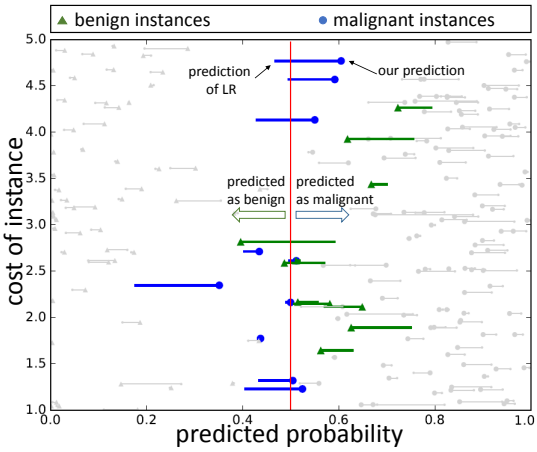

Fig. 12. Comparison between LR and D. The predicted probabilities of D (big end of each line) reflect better the true labels. Triangles should be on the left; circles on right.

Finally, Figure 12 shows why our techniques (here variant D) perform better than LR, that is we show the difference in their prediction. Each line in Figure 12 represents one instance from the dataset. The start of each line indicates the predicted probability based on LR, while the end (shown by a circle/triangle) the probability assigned by our model. In an optimal classification, all triangles (green) should be on the left of the threshold line ($t = 0.5$); all circles (blue) on the right. We *grayed out* all instances which are correctly classified by both

techniques; thus, the *colored lines* show the interesting differences between LR and model D. As we can see, our method pushes more instances to the correct side of the line, i.e. classifies them correctly. Specifically, we also observe these changes for instances with high costs, thus, leading to overall lower misclassification cost.

5.3 Credit datasets

The Kaggle Credit dataset is a bi-class dataset containing 112,915 credit borrowers as instances. Each instance has 10 features with a proportion of 6.74% positive examples. Some features are, e.g., monthly income or monthly debt. Our goal is to perform credit score prediction using our different versions of logistic regression. For assigning instances different costs, we took the same costs proportion as in [8]. Table 3 shows the corresponding results. In contrast to the first two experiments, the best models for Kaggle Credit data, are A and [8]. For $t = 0.5$, indeed version A has the best accuracy of around 93.0% but it saves slightly less than [8]. In contrast, using the optimal threshold, version A wins more on savings than [8] but the number of false classified instances is higher. This means that in model A, primarily instances with low costs are classified incorrectly. The model D performs good w.r.t. the savings and shows in both cases a low number of false classified instances.

Method, t	Incorrect	Correct	Cost	Savings	t	Incorrect	Correct	Cost	Savings
LR, 0.5	2470	34792	8260.35	-	0.39	2573	34689	8056.65	-
A, 0.5	**2444**	34818	7972.32	**3%**	0.38	**2586**	34676	7732.88	**4%**
B, 0.5	2474	34788	8248.05	0.1%	0.36	2483	34779	8056.65	-2%
C, 0.5	2471	34791	8261.57	-0.01%	0.40	2547	34715	8225.40	-2%
D, 0.5	2466	34796	8152.35	1%	0.33	2480	34782	7893.33	2%
Comp., 0.5	**2695**	34567	7823.53	**5%**	0.503	**2497**	34765	7766.05	**3%**

Table 3. Results on credit dataset. Left: threshold $t = 0.5$; right: variable t with overall minimum misclassification cost.

In summary, considering all datasets together, model D has consistently ranked among the best competing methods. Based on our model description (Sec. 3) it also very naturally captures example-dependent cost.

6 Conclusion

In this paper we have presented four different extensions of logistic regression to a cost sensitive one, each using a different loss functions having different properties. We evaluated the impact of each model based on two different public datasets as well as on a vehicle dataset to predict tire change. Our results confirm that a cost sensitive model not only classifies instances with higher importance better but can also improve the accuracy of classical logistic regression. For our use case on tire change service, we obtained significant savings of 10%.

References

1. Zadrozny et. al. "Cost-sensitive learning by cost-proportionate example weighting." ICDM, (2003): 435-442.
2. Günnemann et. al. "Robust multivariate autoregression for anomaly detection in dynamic product ratings." WWW, (2014): 361-372.
3. Weiss, G. M. "Mining with rarity: a unifying framework." ACM Sigkdd Explorations Newsletter, (2004): 7-19.
4. Murphy, K. P. Machine learning. "A probabilistic perspective." MIT press, (2012).
5. Quinlan, J. R. "Induction of decision trees." Machine learning (1986): 81-106.
6. Haykin, S. "A comprehensive foundation." Neural Networks, (2004): 41.
7. Weiss, G. M. "Learning with rare cases and small disjuncts." ICML, (1995): 558-565.
8. Bahnsen et. al. "Example-dependent cost-sensitive logistic regression for credit scoring." ICMLA, (2014): 263-269.
9. Anderson, R. "The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation." Oxford University Press, (2007).
10. Bahnsen et. al. "Cost sensitive credit card fraud detection using bayes minimum risk." ICMLA, (2013): 333-338.
11. Bahnsen et. al. "Improving credit card fraud detection with calibrated probabilities." SIAM, (2014): 677-685.
12. Alejo et. al. "Making accurate credit risk predictions with cost-sensitive mlp neural networks." Management Intelligent Systems. Springer International Publishing, (2013): 1-8.
13. Beling et. al. "Optimal scoring cutoff policies and efficient frontiers." Journal of the Operational Research Society 56.9, (2005): 1016-1029.
14. Oliver et. al. "Optimal score cutoffs and pricing in regulatory capital in retail credit portfolios." University of Southampton, (2009).
15. Verbraken et. al. "Development and application of consumer credit scoring models using profit-based classification measures." European Journal of Operational Research 238.2, (2014): 505-513.
16. Lomax et. al. "A survey of cost-sensitive decision tree induction algorithms." CSUR, 45.2, (2013): 16.
17. Bahnsen et. al. "Ensemble of Example-Dependent Cost-Sensitive Decision Trees." arXiv preprint arXiv:1505.04637, (2015).
18. Mobley, R. K. "An introduction to predictive maintenance." Butterworth-Heinemann, (2002).

7 Appendix

$$\frac{F_{a_i log^{b_i}}}{F_{a_i log^{b_i}}} = \frac{a_i \Gamma(b_i+1, 0.6931)}{a_i(\Gamma(b_i+1) - \Gamma(b_i+1, 0.6931))} \stackrel{!}{=} c_i \cdot \frac{F_{log}}{T_{log}}$$

$$\Leftrightarrow \Gamma(b_i+1, 0.6931) \stackrel{!}{=} c_i \cdot \frac{F_{log}}{T_{log}} \cdot \Gamma(b_i+1) - c_i \cdot \frac{F_{log}}{T_{log}} \Gamma(b_i+1, 0.6931)$$

$$\Leftrightarrow \frac{\Gamma(b_i+1)}{\Gamma(b_i+1, 0.6931)} \stackrel{!}{=} \frac{1 + c_i \cdot \frac{F_{log}}{T_{log}}}{c_i \cdot \frac{F_{log}}{T_{log}}} = 1 + \frac{T_{log}}{c_i \cdot F_{log}}$$

3.5 Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals

Peer-reviewed Conference Paper

Autors **Nikou Günnemann**, Jürgen Pfeffer

In *International Workshop on Learning with Imbalanced Domains: Theory and Applications at ECML-PKDD, pp. 92-102, 2017, [23].*

Abstract This paper addresses for the first time the problem of engines' damage prediction using huge amounts of imbalanced data from "structure borne noise" signals related to the internal engine excitation. The achieved results from classical classification methods such as cost sensitive random forest, cost sensitive logistic regression and kernel based method such as SVM showed uninspiring performance. Thus, analytical approaches from the field of deep learning/neural networks are used. Specifically, to consider the temporal evolution of frequencies in the measured signals, models covering the basic idea of moving average (*convolution*) are regarded.

In this context, we propose the usage of a convolutional neural networks on our temporal input signals, subsequently combined with additional static features. Using informative mini batches during training we take the imbalance of the data into account, thus enabling to better capture the rarely occuring malfunctions of engines. The experimental results indicate good performance in detecting the minority class on our large real-world use case.

Contribution of thesis author: Algorithmic design, development and implementation, experimental design and analysis, manuscript writing, revision and editing.

Proceedings of Machine Learning Research 74:92–102, 2017 Full Paper - LIDTA 2017

Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals

Nikou Günnemann NIKOU.GUENNEMANN@TUM.DE
Jürgen Pfeffer JUERGEN.PFEFFER@TUM.DE
Technical University of Munich, Germany

Editors: Luís Torgo, Bartosz Krawczyk, Paula Branco and Nuno Moniz.

Abstract

This paper addresses for the first time the problem of engines' damage prediction using huge amounts of imbalanced data from "structure borne noise" signals related to the internal engine excitation. We propose the usage of a convolutional neural network on our temporal input signals, subsequently combined with additional static features. Using informative mini batches during training we take the imbalance of the data into account. The experimental results indicate good performance in detecting the minority class on our large real-world use case.

Keywords: Prediction, Predictive Maintenance, Imbalanced Data, Deep Learning.

1. Introduction

Combustion engines are becoming increasingly complex to meet a variety of standards. To guaranty a smooth assembly process and functional engines, an efficient cost effective cold test is performed at the end of each production line. During the cold test, the engine is passively driven by an electric motor. In doing so, several measurements including structure bornes – mechanical vibrations at lower frequencies as 50 **Hz** or 100 **Hz** – are measured for determining the properties of the materials of the engines (Cremer and Heckl, 2013). Based on these structure-borne measurements, the goal of our work is to predict the result of the endurance run for the tested engine: is the engine fully functional or defective?

The collected measurements, however, implicate various challenges, making the prediction task very hard: (1) The available signals collected from different process steps refer to different views. While the structure-borne measurements are collected during the cold test, other views such as the engine information in general or environment features are additionally considered. Such 'multi-view' data are of different types followed by different statistical distributions and different kinds of uncertainties (Sun, 2013). (2) Furthermore, the high dimensionality of the collected signals pose a huge challenge: each signal itself represents a time series. It is worth point out that the lengths of these time series differ between signals. Thus, it can not simply be treated as a multivariate time series. (3) Most importantly, the data suffers from a highly imbalanced distribution of class labels. That is, the class of defective engines is highly under-represented compared to the class of non-defect ones which makes learning quite a difficult task.

Motivated by the effectiveness of deep learning techniques in recent years, we study in this work – using data from a real world use case – whether such architectures also enable to predict defective engines from the collected imbalanced data.

2. Related Work

Deep learning has achieved extremely high performance in many tasks in the field of image or video recognition (Schmidhuber, 2015; Karpathy et al., 2014) as well as natural language processing (Mikolov et al., 2013). Likewise, researchers have investigated deep neural network architectures and algorithms to evaluate their performance in time series tasks – particularly, real world applications with focus on multivariate time series classification have become increasingly important, such as in health care, human activity recognition, or renewable energies (Doucoure et al., 2016; Yang et al., 2015; Zheng et al., 2014; Borovykh et al., 2017).

Since neural networks are capable of extracting non-linear information from data they are a natural fit for time series which rarely represent only linear effects. While on a first sight, temporal/sequence neural network models such as RNNs/LSTMs seem to be a natural choice for temporal data classification, the early work of Kim (2014) has shown that simple Convolutional Neural Networks (CNNs) are equally or even better suited for such tasks. With this idea in mind, (Cui et al., 2016; Borovykh et al., 2017) have shown that CNNs perform impressively not only in classical use cases like face verification and audio classification but also in tasks like time series classification.

The main reason for the success of CNN in this context is explained by its ability to learn from local information, being correlated to further features by sliding the convolutional filter along time axis. Denoting the convolution as the main advantage of this model, in this paper we focus on the questions: (a) Having a real-world multi-view dataset with high dimensionality, is a CNN architecture capable of predicting engines affected by malfunctions with a high Area Under the Curve (AUC)? (b) Can CNNs perform well even when the data is massively imbalanced?

To answer these questions, we investigate a CNN deep learning architecture (Section 4) to predict defective engines using data from a real-world use case (Section 3). We investigate different architectures of the CNN, including different number of layers, using different pooling functions and activation functions in our experimental study – as well as different properties of the data and its effect on the result (Section 5).

3. Data Description

To gather the structure borne noise of each engine, a vibrometer is used to measure different mechanical surface vibrations in time domain. Besides, the influence of the engine vibration on the air pressure is measured. Using fourier transformation, the measured signals are converted in sequential frequency domain. Figure 1 shows a snippet of 3 measured signals for *one* engine. While the x-axis shows different sequence of frequency spectra, the y-axis presents the measured values pointing to e.g. a vibration signal or crankshaft motions etc.

Formally, the resulting data is a set of matrices $S_i \in \mathbb{R}^{N \times L_i}$, where each $i \in M$ represents *one* signal measured for all instances. Here, N denotes the number of instances/engines and

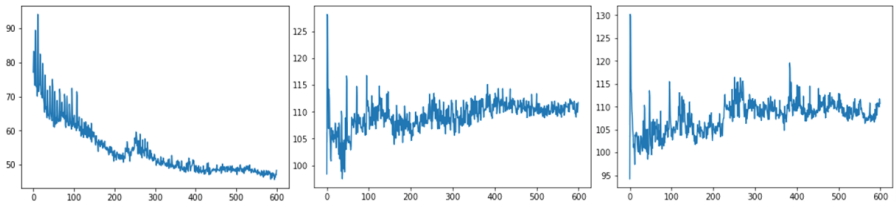

Figure 1: Visualization of three different signals for one engine.

L_i the length of frequency spectra regarding signal i. Accordingly, the jth row of S_i refers to the frequency spectrum collected for engine $j \in N$ regarding signal i. For simplicity we denote these row vectors with x_j^i. Note again that for each signal i the length of these vectors L_i might differ.

Furthermore, for each engine we have a set of non-sequential (environmental) variables, which are indicated by the matrix $E \in \mathbb{R}^{N \times L_E}$ – with e_j denoting the jth row vector. And each instance $j \in N$ is assigned to a label y_j with $y_j \in \{0, 1\}$. Here, $y_j = 1$ represents class of instances with a certain issue and $y_j = 0$ the opposite. We denote with N_1 all instances having label 1, and N_0 all instances with label 0.

Overall, for our use case we have $\approx 93,000$ engines. For each engine, 6 different sequential based signals complemented with 7 environment features are considered. The ratio of defected instances compared to non-defective ones is around 1:32.

It is worth noting that the signals of functional and defect instances cannot be distinguished easily, making the prediction very difficult and not possible by humans. It is also noticeable that the illustrated signals include strong fluctuations with several local minima and maxima. By using CNNs on S_i, our goal is to exploit this information to perform accurate predictions.

4. Convolutional Neural Network on Engine Vibration Signals

In the following we will describe the structure of our deep CNN used to classify an engine as a malfunction one or not.

4.1. Deep CNN Architecture

Figure 2 shows the general structure of the deep feedforward neural network used in our proposed model.

(A) First, to facilitate the training of CNN layers and simultaneously to improve the efficiency, straight after signal processing, we first down-sample our signals by applying a (1x5) Max-pooling. In doing so, we progressively reduce the spacial size of the amount of features and computation complexity while keeping the location sensitivity in the model since the size of pooling is small.

(B) Second, we use multiple CNN/pooling layers as non-linear learnable feature extractors on the temporal signals. More precise, we combine the following three functions whose variations we investigate in our experimental analysis:

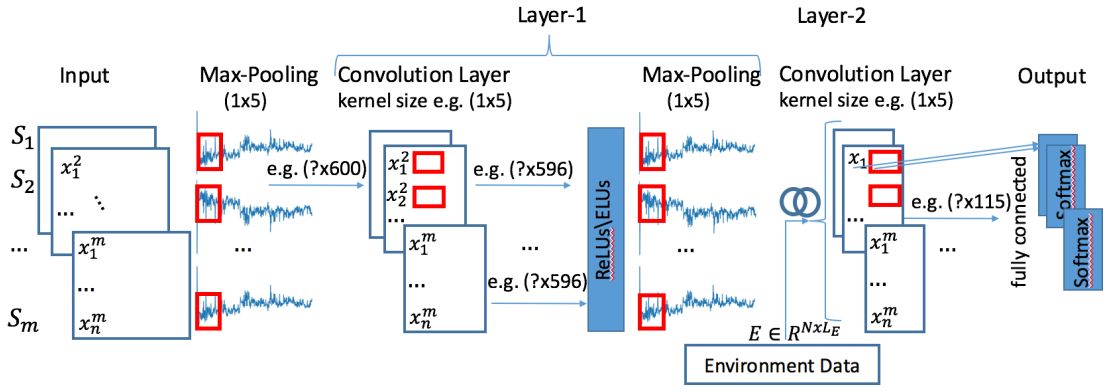

Figure 2: An overview of the two layered CNN-architecture: as input it obtains the engine signals which are concatenated with environment data on a later layer

(i) Convolutional layer: It convolves learnable kernel filters across the input data S_i to compute a dot product between the entries of the filter and the input. For each signal (and layer) we learn different kernel weights, thus, accounting for the multi-view nature of the data. As a result, the network learns different filters which aim to detect specific type of features at certain positions. To extract maximum local temporal information from the whole data, it is beneficial to restrict the size m_l of the filters f_i^l, where l is the corresponding layer and i the ith signal. Note that in contrast to, e.g., images our filters are one-dimensional (i.e. vectors).

(ii) Activation layer: The results generated by the convolutional layer is given as an input to the *activation function* to capture some of the non-linearities from the input data. In our model we use the exponential linear unit *(ELU)* activation function (Clevert et al., 2015). While generally the most used activation function is the rectified linear units (ReLUs) Nair and Hinton (2010), it turned out that ELUs are more powerful in our scenario (see experimental study). By mapping negative values to zero, ReLU are helpful in leading to less dense solutions. However, as a disadvantage the phenomena of *dead* ReLU can be observed – e.g. in the case when having a large negative bias term from the previous layer.

One attempt to address this issue is the Exponential Linear Units (ELUs), defined as:

$$f(x) = \begin{cases} x_i^{l-1}, & \text{if } x_i^{l-1} > 0 \\ \alpha(\exp(x_i) - 1), & \text{if } x_i^{l-1} \leq 0 \end{cases} \tag{1}$$

In contrast to ReLUs, ELU can deal with negative values of feature maps from convolutional layers (Clevert et al., 2015). This is realized by α with $\alpha > 0$. The negative values of feature maps push the mean activations closer to zero which enable faster learning for bringing the gradient closer to the natural gradient. When the input gets smaller, ELUs saturate to a negative value. This decreases the variation and the information propagate to the next layer. The performance of both functions if analyzed in Section 5.

(iii) Pooling Layer: Finally, we down-sample the resolution of the input from the previous layer using either a) the average value in each neighborhood within the selected

pooling window or b) the maximum value. The intuition is, that the exact location of features is not important as its rough location relative to other features Zheng et al. (2014). Note that the last convolutional layer is not followed by a pooling. We analyzed the effects of different pooling window and convolution sizes in our experimental study.

(C) Last, after performing potentially multiple CNN/pooling layers (operating on each temporal signals) we combine the output with the static environmental data by feeding them jointly into a fully connected neuron – followed by a softmax activation function to generate the final classification decision.

More precise, let $\hat{x}_j^i \in \mathbb{R}^{k_i}$ be the output of the last CNN layer (for engine j and signal i) and e_j the environmental features of engine j, the predicted probability of being a malfunction engine is given by

$$\tilde{y}_j = \frac{1}{1 + exp(-w^T \cdot \delta_j + b)} \quad \text{where} \quad \delta_j = [\hat{x}_j^1, \hat{x}_j^2, \ldots, \hat{x}_j^M, e_j]$$

Here $w \in \mathbb{R}^{L_E + \sum_{i=1}^{M} k_i}$ and $b \in \mathbb{R}$ are learnable weights. Since we consider here a two-class problem, the probability of being a correctly functional engine is accordingly $1 - \tilde{y}_j$.

4.2. Loss function, class imbalance, and batch learning

To learn the parameters of the neural network, we have to define a corresponding loss function – operating on the predicted class probabilities. Naively, in a (binary) classification task one could employ the cross entropy loss:

$$L_1(\{\tilde{y}_i\}_{i \in N}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \cdot log(\tilde{y}_i) + (1 - y_i) log(1 - \tilde{y}_i)] \tag{2}$$

where y_i denotes the true label and $\tilde{y}_i$ the predicted probability of the instance i, respectively. By minimizing the loss – e.g. using gradient descent –, our goal is to lower the prediction error with the hope to keep the AUC on the test set as high as possible.

Class Imbalanced Problem However, the engine dataset analyzed in this paper is hit by the class imbalance problem (Japkowicz and Stephen, 2002), where number of positive instances (i.e. the number of malfunctioned engines in our case), is clearly less than the total number of non-damage engines. When the number of instances from one class arises the other one, the learning process from the training data is adversely effected. Due to the importance of this issue, many existing research works have addressed this problem (Bermejo et al., 2011; Mazurowski et al., 2008). In general, these approaches can be divided into three main classes (Galar et al., 2012): ***(a) Algorithm level approaches*** try to modify the learning algorithm to bias the learning with focus on the even class as the minority one. Required is here a prior know how why the learning behaviour of the algorithms fails when e.g. the class of damaged engines is in minority. ***(b) Data level approaches*** rebalance the dataset by resampling the even class and the non even class in e.g. an equal distribution (Stefanowski and Wilk, 2008). ***(c) Cost sensitive learning*** is a combination of the first two versionsa) and b). While the focus is to assign costs to different instances to e.g. prioritize them in the minority class, the classifier is biased toward the even class

by minimizing e.g. the result of the loss function used in classification (Maldonado et al., 2014; Günnemann and Pfeffer, 2017).

In this paper, we tackled the class imbalance problem by using a combination of data level and cost sensitive approaches. More precise, we used stochastic learning (mini batches) to not only speed up the learning process but also to handle imbalance data.

First, we aim to minimize the following loss function

$$L_2(\{\tilde{y}_i\}_{i\in N}) = -\frac{1}{N}\sum_{i=1}^{N}[\alpha \cdot y_i \cdot log(\tilde{y}_i) + (1-y_i)log(1-\tilde{y}_i)] \tag{3}$$

where $\alpha = \frac{|N_0|}{|N_1|}$ is the ratio of class 0 to 1 (in our data: $\alpha \approx 32$). That is, class 1 is α times less frequent than class 0. Intuitively, using Eq. 3, the less frequent class gets higher influence.

Second, we minimize the loss function using stochastic gradient descent using *informative* mini batches. Instead of randomly selecting instances for the minibatch, each minibatch B contains *all* instances N_1 and a random subset of instances from N_0 of the same carnality. Thus, the batch B is 'biased' towards the minority class (compared to the original data distribution). This is specifically helpful if the minority class is very small in the overall data since the likelihood to pick these instances in a minibatch would be small.

Using the informative batches B, the stochastic gradient descent step is now computed based on the loss L_1 (not L_2): $L_1(\{\tilde{y}_i\}_{i\in B})$. The class imbalance factor α is already absorbed by the minibatch. Thus, overall we not only deal with class imbalance problem but also speed up the learning process by training our CNN on a stochastic sample.

5. Experimental Results and Discussions

As mentioned in Section 3, the performance of proposed CNN network was tested on a structure born noise dataset consisting of multiple signals, each in turn, representing 92890 frequency spectra vectors. In our case, each vector presents an engine mapped to a binary label. Using the proposed CNN architecture on our dataset, our goal is to classify the two groups of damaged respectively no damaged engines from each other with a high AUC. We will now analyze our model in more details. All experiments were performed using a train-test split of 80:20 and learning was done using the ADAM optimizer until convergence. All numbers indicate the results on the test set. The model was implemented in TensorFlow and executed on a K80 GPU disposing of 6 cores and 56GB Memory.

5.1. Analysis of parameters

We start by assessing the performance of our model when varying different parameters of the architecture and learning method.

Window sizes and number of layers. Table 1 summarizes different AUCs produced by using different sizes of convolutional filters or pooling windows used in our model – as well as the number of layers used in the architecture (see Fig 2). As described, a convolution layer is followed by a pooling layer unless the last layer of the model. For a fair comparison, we used the same activation function (ELUs), pooling function (Max-function), and learning rate 0.0001 for all architectures.

nr.	layer 1		layer 2		layer 3		auc
	conv	pool	conv	pool	conv	pool	
1	(1,3)	(1,3)	(1,3)	-	-	-	0.889
2	(1,5)	(1,5)	(1,5)	-	-	-	0.913
3	(1,3)	(1,5)	(1,3)	-	-	-	0.90
4	(1,3)	(1,10)	(1,3)	-	-	-	0.90
5	(1,5)	(1,10)	(1,5)	-	-	-	0.91
6	(1,3)	(1,3)	(1,3)	(1,3)	(1,3)	-	0.87
7	(1,5)	(1,5)	(1,5)	(1,5)	(1,5)	-	0.89

Table 1: Impact of size of filters and poolings on AUC.

For the convolution layer, we consider the two filters of size (1x3) and (1x5) since any bigger size would cause loss of information in the temporal domain. In addition to this, we analyzed the influence of max-pooling windows of size (1x3), (1x5) and (1x10) on AUC values. Considering the results of the 2 layers architectures, it is obvious that using pooling and filter with size of (1x5) leads to the best result followed by a pooling size of (1x10), where AUC value drops slightly.

Considering the results of the 3 layers' architecture, a better result is not achieved. One reason might be the larger number of parameters, which makes the learning approach more difficult and also might lead to overfitting. Therefore, in the following experiments, we focus on the two layer architecture with a window size of (1x5) for pooling and filtering.

Learning rate and convergence. We also evaluated the AUC of our model in dependency of the learning rate parameter. From the results in Figure 3, we can see that by setting the value of the learning rate denoted by lr equal to 0.01, the loss value fluctuates unstably.

However, by decreasing the learning rate parameter, a continuously decreasing loss can be obtained. The AUC values for several selected learning rates are shown in Figure 4, with 0.001 achieving the highest AUC value (and being efficient at the same time) and thus also used in the following.

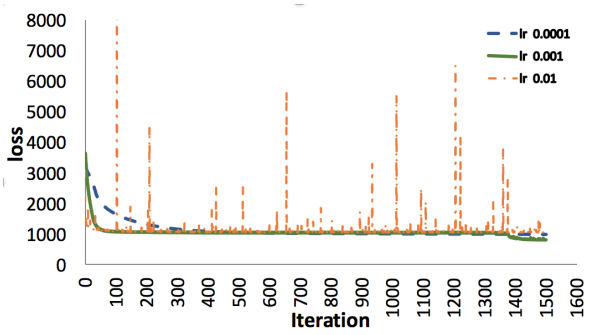

Figure 3: Evaluation of Learning rate versus convergence

learning rate	auc
0.0001	0.929
0.001	0.913
0.01	0.899

Figure 4: Analyzing prediction performance in dependency of learning rate

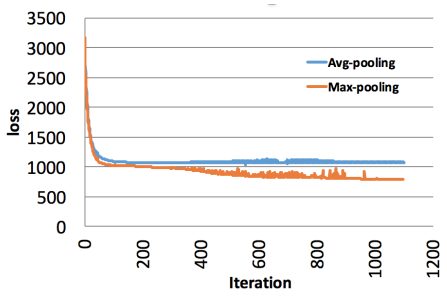

Figure 5: Evaluation of pooling functions

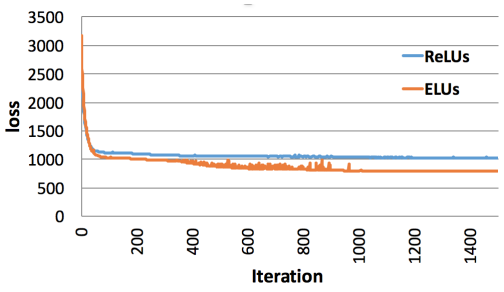

Figure 6: Evaluation of activation functions

Pooling and activation functions. In Figure 5 and 6, we explore the impact of the two pooling functions max-pooling vs. average-pooling on AUC values as well as the activation functions ELUs vs. ReLUs. In Figure 5, it is obvious that by using max-pooling function a lower loss has been achieved. The AUC results in Table 2 confirm that this lower loss is reflected in a better AUC values. Important to mention is that the used activation function in both cases was the ELU. Exploring the impact of ReLU on AUC value versus ELU is shown in Figure 6 as well as in Table 2 under the activation row. Again, here we can conclude that ELU combined with max-pooling returns the best result.

	function	loss	auc
activation	ELUs	785.71	0.929
	ReLUs	1024	0.84
pooling	Max	785.71	0.929
	Avg	1065.7	0.895

Table 2: Evaluation of activation and pooling functions.

Overall, our experimental analysis on our real-word dataset has shown that a two layered CNN architecture including an ELU activation function combined with Max pooling leads to the best AUC results.

5.2. Impact of data properties on the confusion matrix

In industry, it is essential for prediction tasks to reach a high precision and recall on the *minority* class – since our main goal is to avoid malfunctions and high costs. Having imbalanced data, such as in our use case, makes the realization of this goal difficult since often false positives (i.e. functional engines will be predicted as defective) are reported.

As discussed in Section 4.2, we use informative mini batches to handle the imbalanced data, thus, realizing the loss presented in Eq. 3. In the following we compare the classification performance when considering or ignoring the aspect of imbalance (effectively setting $\alpha = 1$ in Eq. 3).

Figure 7 (left) shows the results by considering the corresponding recall-precision curves. While the x axis shows the recall value, presenting the proportion of true positive in the set of truly damaged instances, the y axis refers to precision. The red line shows our principle while the blue line the result when ignoring the imbalance. As clearly shown, our informative minibatches lead to a significantly higher precision even for larger recall values.

Considering the red line, we can observe that for high recall values of around 0.75-0.80, we clearly outperform the blue line – considering the imbalance improves the results. In contrast to the original class label distribution of 1:32 we can reach a level of around 1:1

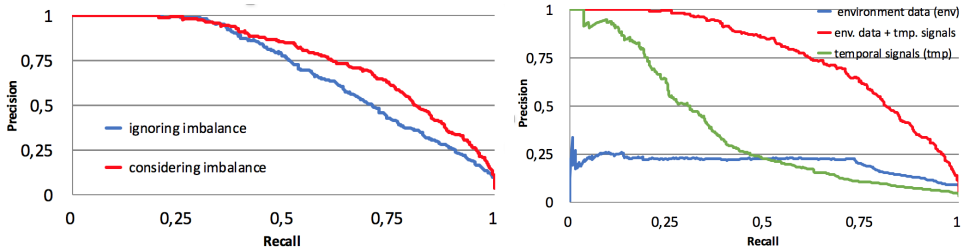

Figure 7: Comparing classification performance on precision-recall curves: left) Prediction when considering class imbalance (red line) clearly outperforms prediction when ignoring class imbalance (blue line). right) Influence of different data sources on prediction performance.

(precison > 0.5) even for high recall values – a significant improvement. *It is also worth noting that our method does not simply lead to a trivial solution where all instances are predicted as being in the majority class* – a common issue for imbalanced data.

Multi-view data. Another important property of our dataset is that the data originates from different sources. In general, we have the two main data sources, one related to the cold test results (temporal data) and the other one related to the (static) environmental data including features such as engines' model or number of cylinders. To examine the relevance of each source on prediction performance, we analyzed each source individually. Figure 7 (right) shows the corresponding recall-precision curve for each dataset.

Considering the lines for the environmental data only, it is clear that the classifier only slightly discriminates the classes. In case of the temporal data, only for small recall values high precision can be reached; followed by a quick drop. However, as it becomes apparent both sources cover different combination: by combined them (our approach; red line) we clearly outperform each individual source. Our proposed architecture realizes this combination in an effective way.

6. Conclusion

In this paper, state of the art convolutional neural networks are used to deal with predicting defective engines from a real-world use case. The data contains multi-view information and is highly imbalanced. To predict the defective engines, we used a two layer CNN-architecture operating on the temporal signal including different attempts on different activation and pooling functions to reach the best results. To deal with the imbalance problem, we used a combination of datalevel and cost sensitive approaches. The experimental results showed that this approach leads to better recall and precision results.

While this paper analyzed for the first time the highly challenging scenario of engine classification based on structure-borne noise measurements there are multiple directions for future work. First, we aim to analyze ensemble methods to further improve the precision and to handle the class imbalance. Additionally, it is of interest to combine our CNN architecture with different sequential models as in e.g. Günnemann et al. (2014a,b) with the goal to capture further dependencies.

References

Pablo Bermejo, Jose A Gámez, and Jose M Puerta. Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. *Expert Systems with Applications*, 38(3):2072–2080, 2011.

Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Lothar Cremer and Manfred Heckl. *Structure-borne sound: structural vibrations and sound radiation at audio frequencies*. Springer Science & Business Media, 2013.

Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

Boubacar Doucoure, Kodjo Agbossou, and Alben Cardenas. Time series prediction using artificial wavelet neural network and multi-resolution analysis: Application to wind speed data. *Renewable Energy*, 92:202–211, 2016.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

Nikou Günnemann and Jürgen Pfeffer. Cost matters: A new example-dependent cost-sensitive logistic regression model. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 210–222. Springer, 2017.

Nikou Günnemann, Stephan Günnemann, and Christos Faloutsos. Robust multivariate autoregression for anomaly detection in dynamic product ratings. In *Proceedings of the 23rd international conference on World wide web*, pages 361–372. ACM, 2014a.

Stephan Günnemann, Nikou Günnemann, and Christos Faloutsos. Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 841–850. ACM, 2014b.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1746–1751, 2014.

Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information Sciences*, 286:228–246, 2014.

Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2): 427–436, 2008.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751, 2013.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61: 85–117, 2015.

Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. *Lecture Notes in Computer Science*, 5182:283–292, 2008.

Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001, 2015.

Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

4 Discussion

In this section, the published papers are further discussed. First, the achieved results are summarized, then limitations along with potential future extensions are discussed.

4.1 Summary and Future Works

In this publication based dissertation, techniques for rare event detection in time series data from the two core topics of unsupervised generative and supervised discriminative models have been developed. The relevance and usability of each method has been shown based on their application on real datasets. All listed papers in this thesis, are original published peer reviewed papers which are published and presented at international conferences. In total, this dissertation contains five published papers: the first three [25, 20, 21] discuss new generative methods to detect rare local events within time series; the last two papers [23, 22] reflect the potential discriminative methods on labeled data from the automobile industry to distinguish rare anomalous events from normal ones, while taking into account the highly imbalanced distribution of class labels.

The core contribution of this thesis with respect to such rare event detection on time series data lies in the modeling, implementation, and evaluation of efficient algorithms that are able to solve these tasks.

Focusing on the temporal analysis of rating data about a product's quality in Amazon, Tripadviser or Yelp, the two novel methods of *RLA* [20] and *SpotRate* [25] are developed. Assuming that temporal data are polluted with anomalies, the two methodologies deal with challenges of i) how to automatically detect the base behaviour of users regarding a product's evaluation over time, and ii) how to detect intervals where the rating distribution differs from this base behavior, e.g., due to attacks or spontaneous changes in the product's quality. In this context, intervals including anomalies are considered as time points whose values deviate from the rest of the measured data points. Such intervals are categorized as local anomalous behavior, in this thesis.

Using generative approaches, we modeled the observed data as a mixture of a latent, autoregressive model representing the base behavior with sparse anomaly signals. Particularly, the developed algorithms exploit principles of variational inference and dynamic programming. The potential of the developed methods were shown based on experimental studies to spot anomalies and to use the base behaviour for studying the evolution of products.

In addition, generative approaches were developed by analyzing different kind of temporal data from social media such as twitter. The contribution of this work dealt with the two question of i) how to detect informative events in general and ii) how to distinguish

relevant events from others. The mentioned challenges are tackled with a statistical model for detecting events by spotting significant frequency deviations of the words' frequency over time. Besides single word events, this model also accounts for events composed of multiple co-occurring words, thus, providing much richer information. The used statistical process is complemented with an optimization algorithm to extract only non-redundant events, overall, providing the user with a brief summary of the current events. An additional advantage of the designed method is its general applicability to arbitrary sets of words, i.e. it is agnostic to the used dictionary.

Apart from generative models, in the second part of this thesis, supervised discriminative models are used on labeled temporal data with focus on distinguishing malfunction events (errors) from normal one while taking into account the highly imbalanced distribution of class labels. In this thesis, the appearance of anomalies as instances in the dataset is categorized as global anomalous behavior. In this context, the focus is on predicting malfunction events as accurately as possible, since in industrial use cases, misclassifying such instances could cause high costs. To tackle this problem, in [22], a new example-dependent cost sensitive prediction model is introduced extending the well-established idea of logistic regression. The advantage of this model is to enable different costs of misclassified instances and obtains prediction results leading to overall less cost. This method consistently outperforms the state-of-the-art in example-dependent cost-sensitive logistic regression on various datasets. This is confirmed in [22]. Complementary to this, in [23], the principle of convolutional neural networks is used to deal with predicting defective engines from a real-world use case. The data contains multi-view information and is highly imbalanced. To predict the defective engines, a designed two layer CNN-architecture is used, operating on the temporal signal including different attempts on different activation and pooling functions to reach the best results. To deal with the imbalance problem, a combination of data level and cost sensitive approaches are used leading to better recall and precision results.

Overall, this thesis showed that depending on the type of time series data, different approaches from the area of machine learning can be used to recognize anomalous events over time. Nevertheless, the results show that future research can be done to extend the usability of the models. For instance, the works [25] and [20] currently investigate every product individually. A joint analysis of multiple products might show further anomalies and unusual patterns which are only visible by considering cross-product correlations. Future works are also needed to investigate how to directly incorporate textual data from the reviews into RLA and Spotrate and how such additional information can influence the accuracy and interpretability of the models. The work [21], in contrast, would highly benefit from a live streaming implementation. While the model itself permits such an execution, the detection of non-redundant events potentially needs further investigation for speeding up its performance.

Focusing on the supervised models presented in Section 2.3, new hybrid approaches as presented in [40, 48] are a potential direction to investigate. Transferring the usability of such methods to our use cases could provide the ability of extracting more dependencies from the observed data to classify global anomalies better than yet. The idea in these papers is the combination of the existing RNN models with the CNN models with the

hope of learning not only from local variations and their influences on the corresponding labels but also to learn from long term dependencies. Most of the existing hybrid methods are shown on data from the images and language domain. Investigating the performance of such methods on temporal data as in [23], could be a highly inspiring new case study.

Finally, another important aspect is the robustness of the designed models. While RLA and Spotrate ensure robustness, other methods such as CNNs point out to have weakness in this context, as many other machine learning methods as well. Having methods with a guaranteed robustness is of importance, specially for industrial use cases. Thus, future works are needed to improve the reliability of results for scenarios where the test data does not adhere to the patterns of the training data.

Bibliography

[1] Abdel-Hamid, O. ; Mohamed, A.-r. ; Jiang, H. ; Penn, G. : Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* IEEE, 2012, S. 4277–4280

[2] Akaike, H. : Fitting autoregressive models for prediction. In: *Annals of the institute of Statistical Mathematics* 21 (1969), Nr. 1, S. 243–247

[3] Akaike, H. : A Bayesian analysis of the minimum AIC procedure. In: *Annals of the Institute of Statistical mathematics* 30 (1978), Nr. 1, S. 9–14

[4] Akaike, H. : Likelihood of a model and information criteria. In: *Journal of econometrics* 16 (1981), Nr. 1, S. 3–14

[5] Bar-Joseph, Z. ; Gerber, G. ; Gifford, D. K. ; Jaakkola, T. S. ; Simon, I. : A new approach to analyzing gene expression time series data. In: *Proceedings of the sixth annual international conference on Computational biology* ACM, 2002, S. 39–48

[6] Bishop, C. M.: *Pattern recognition and machine learning*. springer, 2006

[7] Blei, D. M. ; Lafferty, J. D.: Dynamic topic models. In: *Proceedings of the 23rd international conference on Machine learning* ACM, 2006, S. 113–120

[8] Bouchard, G. ; Triggs, B. : The tradeoff between generative and discriminative classifiers. In: *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, 2004, S. 721–728

[9] Box, G. E. ; Jenkins, G. M. ; Reinsel, G. C. ; Ljung, G. M.: *Time series analysis: forecasting and control*. John Wiley & Sons, 2015

[10] Breiman, L. : Random forests. In: *Machine learning* 45 (2001), Nr. 1, S. 5–32

[11] Burnham, K. P. ; Anderson, D. R.: Kullback-Leibler information as a basis for strong inference in ecological studies. In: *Wildlife research* 28 (2001), Nr. 2, S. 111–119

[12] Carpenter, G. A. ; Grossberg, S. ; Reynolds, J. H.: ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. In: *Neural networks* 4 (1991), Nr. 5, S. 565–588

[13] Clevert, D.-A. ; Unterthiner, T. ; Hochreiter, S. : Fast and accurate deep network learning by exponential linear units (elus). In: *arXiv preprint arXiv:1511.07289* (2015)

[14] Cortes, C. ; Vapnik, V. : Support-vector networks. In: *Machine learning* 20 (1995), Nr. 3, S. 273–297

[15] Derntl, M. ; Günnemann, N. ; Tillmann, A. ; Klamma, R. ; Jarke, M. : Building and exploring dynamic topic models on the web. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* ACM, 2014, S. 2012–2014

[16] Esposito, F. ; Formisano, E. ; Seifritz, E. ; Goebel, R. ; Morrone, R. ; Tedeschi, G. ; Di Salle, F. : Spatial independent component analysis of functional MRI time-series: To what extent do results depend on the algorithm used? In: *Human brain mapping* 16 (2002), Nr. 3, S. 146–157

[17] Färber, I. ; Günnemann, S. ; Kriegel, H.-P. ; Kröger, P. ; Müller, E. ; Schubert, E. ; Seidl, T. ; Zimek, A. : On using class-labels in evaluation of clusterings. In: *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*, 2010, S. 1

[18] Goodfellow, I. ; Bengio, Y. ; Courville, A. : *Deep Learning*. MIT Press, 2016. – `http://www.deeplearningbook.org`

[19] Griinwald, P. : Introducing the minimum description length principle. In: *Advances in minimum description length: Theory and applications* (2005), S. 3

[20] Günnemann, N. ; Günnemann, S. ; Faloutsos, C. : Robust multivariate autoregression for anomaly detection in dynamic product ratings. In: *Proceedings of the 23rd international conference on World wide web* ACM, 2014, S. 361–372

[21] Günnemann, N. ; Pfeffer, J. : Finding non-redundant multi-word events on twitter. In: *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on* IEEE, 2015, S. 520–525

[22] Günnemann, N. ; Pfeffer, J. : Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining* Springer, 2017, S. 210–222

[23] Günnemann, N. ; Pfeffer, J. : Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals. In: *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, 2017, S. 92–102

[24] Günnemann, S. : Subspace clustering for complex data. (2012)

[25] Günnemann, S. ; Günnemann, N. ; Faloutsos, C. : Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* ACM, 2014, S. 841–850

[26] Han, J. ; Pei, J. ; Kamber, M. : *Data mining: concepts and techniques*. Elsevier, 2011

[27] Haykin, S. S. ; Haykin, S. S. ; Haykin, S. S. ; Haykin, S. S.: *Neural networks and learning machines*. Bd. 3. Pearson Upper Saddle River, NJ, USA:, 2009

[28] Hecht-Nielsen, R. u. a.: Theory of the backpropagation neural network. In: *Neural Networks* 1 (1988), Nr. Supplement-1, S. 445–448

[29] Hinich, M. J.: Testing for Gaussianity and linearity of a stationary time series. In: *Journal of time series analysis* 3 (1982), Nr. 3, S. 169–176

[30] Hochreiter, S. : The vanishing gradient problem during learning recurrent neural nets and problem solutions. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (1998), Nr. 02, S. 107–116

[31] Hochreiter, S. ; Schmidhuber, J. : Long short-term memory. In: *Neural computation* 9 (1997), Nr. 8, S. 1735–1780

[32] Hooi, B. ; Shah, N. ; Beutel, A. ; Günnemann, S. ; Akoglu, L. ; Kumar, M. ; Makhija, D. ; Faloutsos, C. : Birdnest: Bayesian inference for ratings-fraud detection. In: *Proceedings of the 2016 SIAM International Conference on Data Mining* SIAM, 2016, S. 495–503

[33] Hosmer Jr, D. W. ; Lemeshow, S. ; Sturdivant, R. X.: *Applied logistic regression*. Bd. 398. John Wiley & Sons, 2013

[34] Hosseini, H. G. ; Luo, D. ; Reynolds, K. J.: The comparison of different feed forward neural network architectures for ECG signal diagnosis. In: *Medical engineering & physics* 28 (2006), Nr. 4, S. 372–378

[35] Huang, N. E. ; Wu, M.-L. ; Qu, W. ; Long, S. R. ; Shen, S. S.: Applications of Hilbert–Huang transform to non-stationary financial time series analysis. In: *Applied stochastic models in business and industry* 19 (2003), Nr. 3, S. 245–268

[36] Hubel, D. H. ; Wiesel, T. N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. In: *The Journal of physiology* 160 (1962), Nr. 1, S. 106–154

[37] Kalpakis, K. ; Gada, D. ; Puttagunta, V. : Distance measures for effective clustering of ARIMA time-series. In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on* IEEE, 2001, S. 273–280

[38] Kantz, H. ; Schreiber, T. : *Nonlinear time series analysis*. Bd. 7. Cambridge university press, 2004

[39] Kapur, J. N. ; Kesavan, H. K.: Entropy optimization principles and their applications. In: *Entropy and Energy Dissipation in Water Resources*. Springer, 1992, S. 3–20

[40] Kim, Y. ; Jernite, Y. ; Sontag, D. ; Rush, A. M.: Character-Aware Neural Language Models. In: *AAAI*, 2016, S. 2741–2749

[41] Kohavi, R. u. a.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Ijcai* Bd. 14 Stanford, CA, 1995, S. 1137–1145

[42] Kranen, P. : Anytime algorithms for stream data mining. (2011)

[43] LeCun, Y. ; Bengio, Y. u. a.: Convolutional networks for images, speech, and time series. In: *The handbook of brain theory and neural networks* 3361 (1995), Nr. 10, S. 1995

[44] LeCun, Y. ; Bengio, Y. ; Hinton, G. : Deep learning. In: *nature* 521 (2015), Nr. 7553, S. 436

[45] Lee, Y. ; Oh, S.-H. ; Kim, M. W.: An analysis of premature saturation in back propagation learning. In: *Neural networks* 6 (1993), Nr. 5, S. 719–728

[46] Lester, J. ; Choudhury, T. ; Kern, N. ; Borriello, G. ; Hannaford, B. : A hybrid discriminative/generative approach for modeling human activities. (2005)

[47] Liaw, A. ; Wiener, M. u. a.: Classification and regression by randomForest. In: *R news* 2 (2002), Nr. 3, S. 18–22

[48] Lin, T. ; Guo, T. ; Aberer, K. : Hybrid neural networks for learning the trend in time series. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, S. 2273–2279

[49] Lo, S.-C. B. ; Chan, H.-P. ; Lin, J.-S. ; Li, H. ; Freedman, M. T. ; Mun, S. K.: Artificial convolution neural network for medical image pattern recognition. In: *Neural networks* 8 (1995), Nr. 7, S. 1201–1214

[50] Menze, B. H. ; Van Leemput, K. ; Lashkari, D. ; Weber, M.-A. ; Ayache, N. ; Golland, P. : A generative model for brain tumor segmentation in multi-modal images. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention* Springer, 2010, S. 151–159

[51] Murphy, K. P.: *Machine learning: a probabilistic perspective*. 2012

[52] Nair, V. ; Hinton, G. E.: Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, S. 807–814

[53] Nason, G. P.: Stationary and non-stationary time series. In: *Statistics in Volcanology. Special Publications of IAVCEI* 1 (2006), S. 000–000

[54] Ng, A. Y. ; Jordan, M. I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: *Advances in neural information processing systems*, 2002, S. 841–848

[55] Ngufor, C. ; Wojtusiak, J. : Extreme logistic regression. In: *Advances in Data Analysis and Classification* 10 (2016), Nr. 1, S. 27–52

[56] Pan, W. : Akaike's information criterion in generalized estimating equations. In: *Biometrics* 57 (2001), Nr. 1, S. 120–125

[57] Raftery, A. E.: Bayesian model selection in social research. In: *Sociological methodology* (1995), S. 111–163

[58] Sadowski, P. : Notes on backpropagation. In: *homepage: https://www. ics. uci. edu/˜ pjsadows/notes. pdf (online)* (2016)

[59] Scholkopf, B. ; Smola, A. J.: *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001

[60] Shariff, A. ; Murphy, R. F. ; Rohde, G. K.: A generative model of microtubule distributions, and indirect estimation of its parameters from fluorescence microscopy images. In: *Cytometry Part A* 77 (2010), Nr. 5, S. 457–466

[61] Shibata, R. : Selection of the order of an autoregressive model by Akaike's information criterion. In: *Biometrika* 63 (1976), Nr. 1, S. 117–126

[62] Velicer, W. F. ; Fava, J. L.: Time series analysis. In: *Handbook of psychology* (2003)

[63] Wang, X. ; McCallum, A. : Topics over time: a non-Markov continuous-time model of topical trends. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* ACM, 2006, S. 424–433

[64] Whittle, P. : Estimation and information in stationary time series. In: *Arkiv för matematik* 2 (1953), Nr. 5, S. 423–434

[65] Yule, G. U.: On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 226 (1927), S. 267–298

[66] Zheng, Y. ; Liu, Q. ; Chen, E. ; Ge, Y. ; Zhao, J. L.: Time series classification using multi-channels deep convolutional neural networks. In: *International Conference on Web-Age Information Management* Springer, 2014, S. 298–310