## TABLE OF CONTENTS

Page

INTRODUCTION ..... 1

CHAPTER 1 RESEARCH DOMAIN ..... 5
1.1 Application Environment ..... 5
1.2 Problem Description ..... 8
1.2.1 Complexity of the Environment ..... 8
1.2.2 Complexity of the Data ..... 9
1.3 Research Question and Hypotheses ..... 10
1.3.1 Contributions ..... 11

CHAPTER 2 RELATED WORK ..... 13
2.1 Data collection, storage and processing ..... 13
2.2 Abnormal behavior detection ..... 16
2.2.1 Probabilistic Approaches ..... 18
2.2.2 Non-probabilistic approaches ..... 24
2.2.3 Trajectory based approaches ..... 26
2.2.4 Discussion ..... 29
2.3 LSA with Tensor analysis ..... 30
2.3.1 Introduction to tensors ..... 30
2.3.2 Tensor Decomposition ..... 31
2.3.3 Applications of Tensors in Data Mining so far ..... 32
2.3.4 Anomaly detection using tensor factorization ..... 34
2.4 Summary ..... 38

CHAPTER 3 LOG-LINEAR TENSOR FACTORIZATION ..... 39
3.1 General Overview ..... 39
3.1.1 Data representation ..... 40
3.1.2 Training ..... 43
3.1.3 Testing ..... 45
3.2 The proposed tensor factorization model ..... 45
3.2.1 Learning the empirical probability through factors ..... 46
3.2.2 Optimization ..... 47
3.2.3 Algorithm summary and pseudocode ..... 49
3.2.4 Computational complexity ..... 49
3.2.5 Abnormal Event Detection ..... 50

CHAPTER 4 EXPERIMENTAL SETUP ..... 53
4.1 Datasets ..... 54
4.1.1 Synthetically generated dataset ..... 54
4.1.2 Real-Life dataset ..... 55

4.1.2.1 Reality Mining ..... 56
4.1.2.2 Geo-Life Taxi Trajectories ..... 57
4.1.3 Anomalous Real-Life datasets ..... 60
4.1.3.1 Determining pairs to swap ..... 61
4.1.3.2 Swapping People ..... 62
4.1.3.3 Swapping Hours and Zones ..... 63
4.2 Performance Measures ..... 63
4.3 Models used for comparative analysis ..... 65
4.3.1 Non-negative Tensor Factorization ..... 65
4.3.2 Alternating Poisson Regression ..... 66
4.3.3 One-Class SVM ..... 66
4.3.4 Kernel Density Estimation (KDE) ..... 67

CHAPTER 5 EXPERIMENTS AND RESULTS ..... 69
5.1 Characteristics of the model ..... 69
5.1.1 Visualization of latent factors ..... 69
5.1.2 Convergence Analysis ..... 69
5.1.3 Parameter impact ..... 70
5.2 Parameter tuning ..... 71
5.3 Low-Rank Approximation ..... 74
5.4 Future Event Prediction ..... 80
5.5 Abnormal behavior Detection ..... 83
5.6 Discussions, Limitations and Recommendations ..... 88

CONCLUSION ..... 91

APPENDIX I ONLINE LOG-LINEAR TENSOR FACTORIZATION ..... 93

BIBLIOGRAPHY ..... 97

## LIST OF TABLES

Page

Table 4.1 Confusion Matrix ..... 64

Table 5.1 Future event prediction on Reality Mining Data ..... 83

Table 5.2 Future event prediction on Taxi Trajectory Data ..... 83

## LIST OF FIGURES

Page


Figure 1.1 Types of Badges ..... 6

Figure 1.2 Types of Receivers ..... 6

Figure 1.3 Data transferred from Middelware to Application Software ..... 7

Figure 1.4 Data transfer: Badge detection to Application Software ..... 8

Figure 2.1 Approaches for Abnormal Behavior Detection ..... 17

Figure 2.2 A third order tensor ..... 31

Figure 3.1 Overview of the proposed framework ..... 40

Figure 3.2 Data representation as en event tensor ..... 42

Figure 4.1 Visualization of Reality Mining Dataset ..... 57

Figure 4.2 Visualization of Taxi Trajectory Dataset ..... 58

Figure 4.3 Histogram of raw taxi trajectory data ..... 59

Figure 4.4 Log-polar coordinated taxi trajectory data ..... 60

Figure 4.5 Relative estimate of zones in Taxi Trajectory Dataset ..... 61

Figure 4.6 Area Under Curve ..... 65

Figure 5.1 Plot of latent factors values ..... 70

Figure 5.2 Convergence comparison ..... 71

Figure 5.3 Impact of Latent factor vector size on minimization of error ..... 72

Figure 5.4 Low Rank Approximated MAE on log-linear data ..... 76

Figure 5.5 Low Rank Approximated MAE on Poisson data ..... 77

Figure 5.6 Low Rank Approximated RMSE on log-linear data ..... 78

Figure 5.7 Low Rank Approximated RMSE on Poisson data ..... 79

Figure 5.8 Low Rank Approximated NLL on log-linear data ..... 81

Figure 5.9 Low Rank Approximated NLL on Poisson data..... 82

Figure 5.10 Comparative analysis after swapping events of two persons ..... 85

Figure 5.11 Comparative analysis after swapping zones of personal events ..... 86

Figure 5.12 Comparative analysis after swapping hours of personal events..... 87

## LIST OF ABREVIATIONS

| | |
|---|---|
| LLTF | Log Linear Tensor Factorization |
| LSA | Latent Semantic Analysis |
| MAE | Mean Absolute Error |
| RMSE | Root Mean Squared Error |
| NLL | Negative Log-Likelihood |
| ROC | Receiver Operating Characteristic |
| AUC | Area Under Curve |

## LISTE OF SYMBOLS AND UNITS OF MEASUREMENTS

| | |
|---|---|
| $a, i, ..$ | Scalars are denoted by lowercase letters, |
| $\mathbf{v}, \mathbf{z}, \mathbf{x}, ..$ | Vectors by boldface lowercase letters, |
| $A, M, ..$ | Matrices by capital letters, and |
| $\mathcal{X}$, $\mathcal{Z}$, $\mathcal{T}, ..$ | Higher order tensors by script letters. |
| $\langle\rangle$; E.g. $\langle A, B\rangle$ | denotes Inner product. |
| $\otimes$; E.g. $(A \otimes B)$ | is the Outer product. |
| $\circ$; E.g. $(\mathbf{a} \circ \mathbf{b})$ | is the Hadamard product. |
| $\odot$; E.g. $(A \odot B)$ | denotes the Khatri-Rao product. |

# INTRODUCTION

In recent years, many methods for surveillance have been developed such as Closed-circuit television (CCTV) cameras, satellite imagery, etc. Amongst those methods, Real Time Location Systems (RTLS), usually equipped with Radio Frequency Identification (RFID) technology, have proved to be quite convenient for indoor as well as outdoor monitoring. However, the objects or people being tracked are predominantly monitored manually, by humans. Any malicious behavior is inferred based on that monitoring, so, failure to detect such behavior may pose a threat to safety.

In this report, we show the efficiency of using some machine learning and data mining methods in the field of activity recognition to detect such malicious behaviors and, thus, hope to provide an efficient automated monitoring system for security.

RTLS with RFID uses radio waves to transmit information about the position and profile of the people or objects and, hence, is quite popular for real-time identification, location and tracking of resources [Malik (2009)]. As these systems produce large amounts of information in real-time, processing that incoming volume of data and analyzing it for security purposes becomes a very challenging task. Therefore, we present efficient processing and analyzing methods that help discover relationships between the entities and, thus, detect any specific or abnormal activities within the environment.

The RTLS makes logs of discrete *events* that capture the location information of an entity such as a person or an object, along with the time, day and other related elements, and stores them at fixed intervals. We believe that by analyzing the context of every event, we can infer the *normality* of it. For example, if we consider analyzing only two-dimensional information, where the dimensions are persons and locations, we may see a person going to a recurrent location such as this person's office. But, if we consider adding another dimension of time, we might discover events occurring at odd times, such as midnight which could be an anomalous

event if the person has a day-time job. However, seeing the person's status, as yet another dimension, we could have the category of that person, which could be a security guard on a night-shift. Thus, we can make informed decisions for taking corresponding actions regarding raising an alarm or not.

To be succinct, analyzing *contextual information* helps us understand an event within the given environment, better and hence, will help us perform efficient anomaly detection. Moreover, due to the large volume of the data, we need efficient methods to manage and analyze it.

Therefore, we present our approach [1] Log-Linear Tensor Factorization (LLTF), which addresses the shortcomings of the existing approaches and integrates a convenient solution, to an extent. The proposed approach uses a multi-dimensional structure, called *tensor*. This structure is a generalization of a matrix, which is limited to two dimensions. Due to the flexibility of having multiple dimensions, we can incorporate as many contextual dimensions as deemed necessary to analyze the behavior in a given environment. The formulation of our method allows us to have a speedy training, easy implementation and real-time probability estimation. Having probability values instead of a clear class has two advantages: first, it gives us a score based anomaly detection system through which we can adjust the sensitivity with the means of a threshold. Secondly, it helps determine the variance from normal behavior to understand how *abnormal* the event is. The efficacy of our approach is presented via a series of experiments, which also demonstrate wide applications and possible extensions of our method.

This thesis is structured as follows: chapter 1 elaborates the application background highlighting the motivation and complexities of the problem, thus formally stating the research question, hypotheses and contributions, through this work. Chapter 2 gives a brief description of relevant concepts along with the review of related literature in this domain. The third chapter is

[1] This approach was presented at the 19th Pacific Asia Knowledge Discovery and Data Mining Conference (PAKDD, 2015).

concerned with the methodology of our proposed model. The datasets used, pre-processing steps taken, experimental setup strategies, performance measures and the methods used for comparing our performance are all explained in the fourth chapter. In chapter 5, we present the experiments and corresponding results highlighting the characteristics of our model as well as the out performance of our proposed approach compared to some state-of-the-art methods and discuss the observations. We note some limitations of our proposed approach and propose potential extensions to overcome those limitations, in chapter 6. Finally, we summarize and conclude this work.

# CHAPTER 1

# RESEARCH DOMAIN

In this chapter, an explanation of the intended application environment and the problem complexities arising from implementing an effective security surveillance system is provided. Following that, the research question and hypotheses pertaining to this thesis along with corresponding contributions are presented.

## 1.1 Application Environment

Large, open environments, such as airport terminals, hospitals, manufacturing plants, oil refineries, etc. are always filled with activities wherein people/employees and objects are continuously on the move. Monitoring activities is crucial for the security of valuable equipments, assets and employees.

Despite growing popularity of high-precision GPS, due to privacy concerns, tracking via mobile phone software are yet to be considered for wide-spread commercial applications. Instead, RTLS systems are relied upon for monitoring because they are simple to implement and cost-efficient. The underlying technology can vary between infrared, ultrasound, bluetooth, RFID, and more, depending on the requirements of the application environment. In any case, the general setup for monitoring involves four main components: badges, receivers, middleware and application software.

a. **Badges** are small active transmitter-like tags (as shown in figure 1.1) carried by an employee or attached to an equipment. Due to their active nature, they continuously transmit the associated entity's information such as id, authority level, role or any other category.

b. **Receivers** are devices (as shown in figure 1.2) that usually have a known position. They are used to locate the above-mentioned badges within the environment by sensing for badges in their range. Upon detecting a signal from a badge, the receivers forward the badge information, such as an associated id, to the middleware.


Figure 1.1 Types of Badges as developed by *Purelink Canada Inc.*
Taken From: www.purelink.ca/en/products/rtls-hardware.php

Figure 1.2 Receivers as developed by *Purelink Canada Inc.*
Taken From: www.purelink.ca/en/products/location-receiver.php

c. **Middleware** is the software that resides among the pure RTLS technology components such as the badges, the receivers, etc. Upon receiving the badge id detected by the receiver, the middleware processes that information to get the corresponding receiver's location and location information such as department id, etc. along with badge associated entity's information such as employee id, authority access levels, etc. As shown in figure 1.3, it fetches and forwards all this information, collectively, to the application software for analysis.

d. **Application Software** or the end-user application is the software (as shown in figure 1.3) that interacts with the RTLS middleware and does the work that the end-users are directly interested in such as tracking, security surveillance, etc.

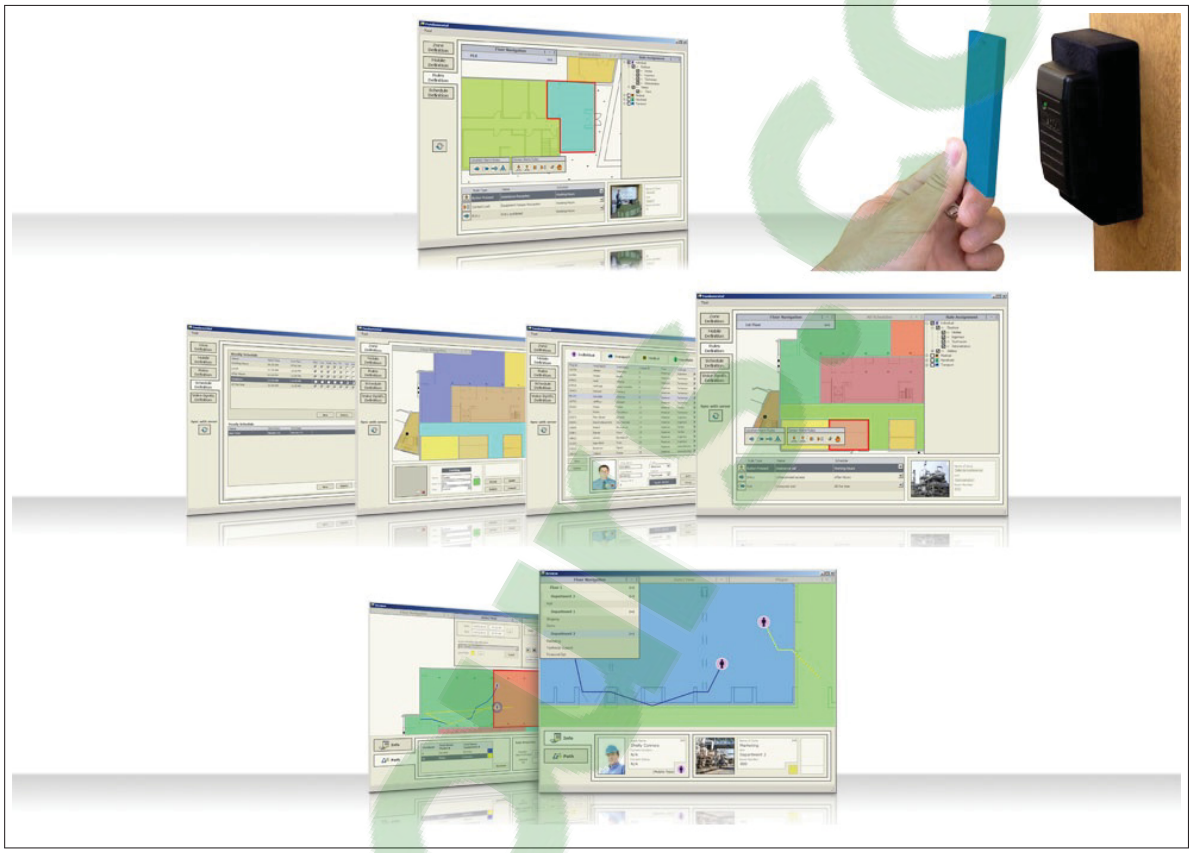


Figure 1.3 Data transferred from Middleware to Application Software for end-users as designed by *Purelink Canada Inc.*
Taken From: www.purelink.ca/en/products/tracking-software.php

To summarize, in any environment, active tags that are associated with the person or object being tracked, transmit information continually, which is received by the nearest location sensor or receiver. A receiver, placed at a known location, can hence receive information from multiple tags. This leads to the receiver forwarding the tags' information to the middleware which retrieves additional information associated to the tags detected by the receiver, as well as that particular receiver. The middleware then forwards this bulk of information to the application software, which is the interface for the end-user who takes actions upon analysis.

Hence, in this work, we focus on the application software. We assume the data received by the application software and formulate our approach to analyze this data.

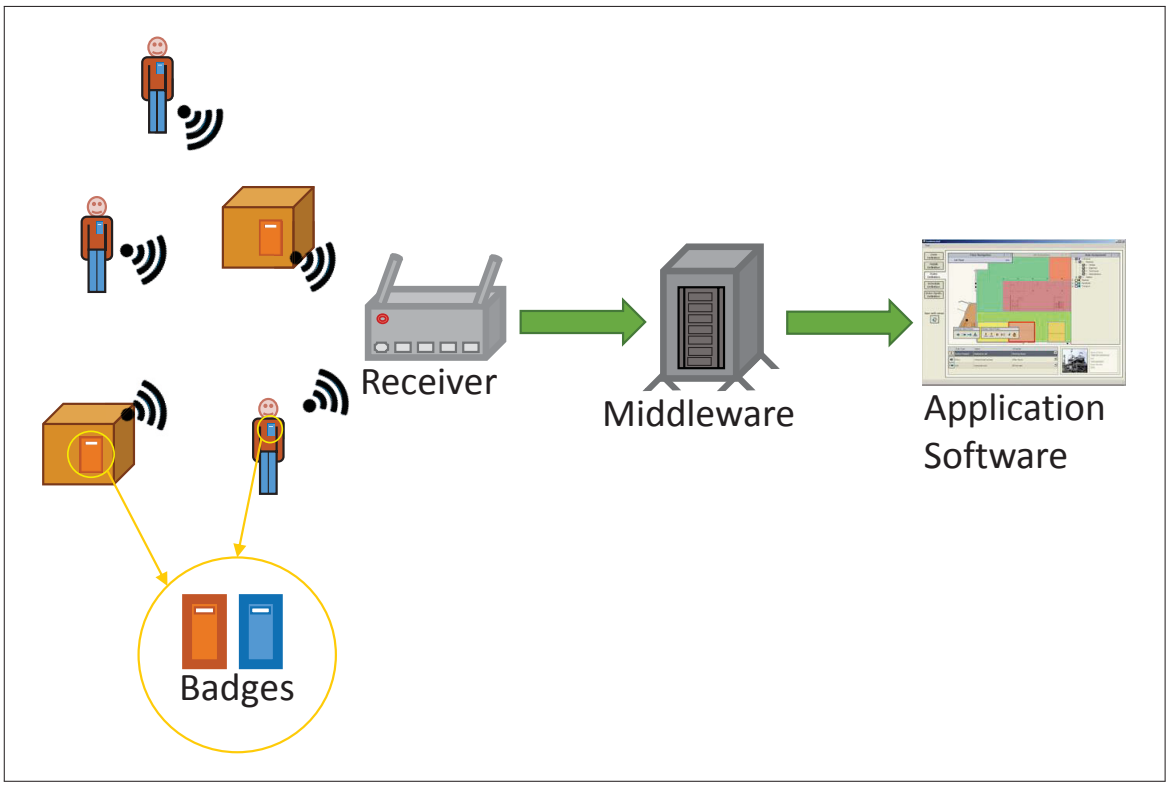



Figure 1.4 Example of badges being detected by a receiver with the information forwarded to middleware and then, the application software

## 1.2 Problem Description

RTLS systems are quite complex, due to which, the convenience and economical aspects reduce. We broadly categorize these complexities into two: environmental and data.

### 1.2.1 Complexity of the Environment

With our application in a large and an open and dynamic environment, such as an airport or a hospital, our project surpasses the conventional activity recognition application in a closed and static environment. This is the toughest challenge that is attempted to be overcome through this work.

The environment defines the conditions that affect the performance of a system. In a closed and static environment, it is easy to track a person or object and determine the behavior. For instance, in a hospital, if medical equipment is being taken out of the respective department, it can be immediately realized. Likewise, in a nursing home, if an elderly is not seen following his or her daily routine, an alert can be made.

However, if the environment becomes open, tracking and predicting the activities based on the movements of people and objects become difficult. This is because it is very unlikely that the patterns constituting an activity would be repeated in the exact same manner. Moreover, if the environment is dynamic, new patterns are very frequent due to which, many statistical models and analytical methods fail to form a behavioral template.

Hence, it is essential that we focus on discovering high-level and repeatable relations and patterns that provide more useful information. The method proposed in this work enables the discovery of patterns through the use of tensor factorization, which captures the latent information from the complex environment for analysis.

### 1.2.2 Complexity of the Data

The data received by the application software, is high-dimensional and quite noisy. Preprocessing techniques such as smoothing or normalizing, applied by the middleware, are limited to hardware, for example, triangulation of receivers.

The data, as received by the application software thus consists of a lot of information corresponding to a single tag. For example, if an employee is being tracked, information such as that employees ID, department, access rights, past-record, current location (e.g., co-ordinates), current location information (e.g., cafeteria or lab), etc. will be analyzed to determine prospect of that employee violating any rules.

If information from all such tags is to be analyzed in the incoming realtime scenario, methods that scale to a high, complex space and computational level are required. Moreover, for the

analysis itself, these methods need to have a high performance despite the noise and raw nature of the data through effective pre-processing techniques.

The existing RTLS based on RFID technologies are not designed to address the scientific and technical challenges of being effective in large, open and dynamic environments. The system needs to scale to high-level of computational and space complexities, go beyond the traditional methods for trajectory mining and pattern detection that allow sparsity of the openness of the environment to hinder the process of learning patterns, and be adaptable to the dynamic nature of the environment as the static analytical methods and models would fail.

Such issues endanger the long term as well as short term security of people as well as valuable objects. Therefore, we intend to provide a robust anomaly detection system that will perform real-time analysis of the multi-dimensional data coming in real-time streams.

### 1.3 Research Question and Hypotheses

Given the complex nature of the problem, we formulate our research question and the corresponding hypotheses as follows:

**Question:** How can we have a simple, efficient model represent and learn events and have it analyze incoming events in real-time for abnormal behavior?

**Hypotheses:**

a. Analyzing the *context* of an event is imperative to make an informed decision to determine the *normality* of it.

b. Contextual events can be represented as a tensor and learned by its latent factors upon factorization.

c. A log-linear formulation models complex non-linear relations representing count data by effectively dealing with sparsity and non-negativity.

d. Trained latent factors are sufficient to represent the environment and analyze for abnormal behaviors.

### 1.3.1 Contributions

The main contribution of this work is the novel and extensible Log-Linear Tensor Factorization (LLTF) method. Based on this method, we have proposed an approach for *real-time contextual anomaly detection in a large, open environment*, which:

a. Represents data in the form of a higher-order model called tensor, which allows us to incorporate any number of contextual dimensions considered necessary to analyze the behavior in a given environment.

b. Factorizes the tensor using the LLTF method, which models the joint probability of the events (co-occurrence of an element across all dimensions), unlike existing tensor factorization approaches, which focus on detecting global anomalies such as works by Kim *et al.* (2009); Sun *et al.* (2006b) or distance-based outliers as proposed by Hayashi *et al.* (2010); Tork *et al.* (2012).

c. Uses a simple log-linear formulation which automatically imposes non-negativity (since we cannot have negative occurrences of events) and helps model complex relations between the contextual dimensions that helps to analyze for behavior, more precisely.

d. Involves optimization using Nesterov's accelerated gradient method leading to a speedy training of the model.

e. Can calculate joint as well as conditional probabilities in real-time, once trained. Joint probabilities may have a small values considering all the possible observed events. Therefore, calculating conditional probabilities of an observation may lead to detailed analysis. For example, if the probability of the person being the one observed, is really low, given the corresponding hour and the location of the observation, then we know that an abnormal activity has occurred.

These above mentioned properties make a significant contribution towards the advancement of the RTLS systems because most of the existing, state-of-the-art methods are either too complex with many constraints or limited to a closed, static environment.

Through this proposed approach, we have validated the aforementioned hypotheses, to an extent, and demonstrated its efficiency through experimental validation, showing it outperform some state of the art methods for detecting abnormal behaviors. A part of this work was presented at the 19th Pacific Asia Knowledge Discovery and Data Mining Conference (PAKDD, 2015) (Rank A). This work was presented under the long presentation category for which, the acceptance rate was about 6%.

In the following chapter, we'll introduce some of the relevant concepts along with a brief review of related literature in this domain.

# CHAPTER 2

# RELATED WORK

In this chapter, we reviewed some of the related approaches taken to tackle the complexities of environment and data, as explained in Section 1.2. The presentation of these relevant works is divided into three main sections, followed by a summary. The first section presents a brief review of various data collection, storage and processing methods applied to increase the business value of RFID based RTLS. The second section presents the popular approaches used to detect anomalies/abnormal behaviors, which are broadly categorized into probabilistic, non-probabilistic and trajectory based models. Towards the end of this section, we briefly summarize the rationale for choosing Latent Semantic Analysis (LSA) approach based Tensor Factorization method for modeling our behavior analysis approach on RTLS systems. A brief introduction to the concept of LSA and tensor factorization, along with some of the related work that have applied LSA and tensor factorization based models in the past, including, for the purposes of anomaly/abnormal behavior detection are reviewed in the third section of this chapter. We recapitulate the key ideas behind these works mentioned in this chapter and explain how we integrated these ideas in our proposed method, in the final summary section.

## 2.1 Data collection, storage and processing

Technology, today, enables location tracking of a particular device via Global Positioning Systems (GPS), Geographic Information System (GIS), Radio Frequency Identification (RFID), Wireless Local Area Network (WLAN) or similar technologies. Novel user interfaces and new applications in smart environments, surveillance, emergency response, military missions, and many such applications are being gradually inspired by activity-aware systems [Choudhury *et al.* (2008)].

Motivation for abnormal behavior detection systems mainly revolves around avoiding various types of emergencies which mainly consider danger to life and health of humans. It is important that we create novel methods to analyze any abnormal behavior and take corresponding actions

immediately. From a machine learning point of view, we can say that the abnormal behavior detection process, like for any machine learning application, involves creating a template of a normal behavior with respect to an environment, i.e. generating a dataset for base modeling. This is done in different ways in different experiments as deemed appropriate since the observations could be in the form of video surveillance, sensor-based data, logs of network server, etc. Upon training a particular classification model based on that template, it is expected that classification model will be able to detect anything out of the ordinary, thus, making it possible to take any actions, should there be any need for it.

Datasets for anomaly detection have been created in different environments by using various methodologies. The popularity of RTLS based on RFID technology, because of its low-cost and robust architecture, has stimulated interest in the development of scalable system for storing and Analyzing the RFID data. The general structure of the RTLS systems is explained in section 1.1. Other methods for data collection included sequences of images from a video [Huang *et al.* (2007); Chung and Liu (2008); Chin-De Liu and Chung (2010); et. al. (2005); Lin *et al.* (2011)], infrared sensors for human presence detection [Hara *et al.* (2002a)] and heart rate monitors to estimate intensity, so that activities like walking and running could be distinguished, [Reiss and Stricker (2012)]. RFID tags have been used by [Hui-Huang Hsu and Chen (2009)] whereas smartphone software tracking the phone's accelerometer data is used by [Frank; Frank *et al.* (2011)].

Data collection methods usually begin with a hardware setup which is the main source of statistical information, within an environment. In many works, a closed environment such as a nursing home [Chung and Liu (2008)] [Chin-De Liu and Chung (2010)] [et. al. (2005)], private apartments [Hara *et al.* (2002a)], gymnasium [Frank], etc. is considered. The activities conducted in these environments are customized to every participating individual's daily routines. Hence, it can be assumed that predicting an individual's activity, at a particular time and location within the environment, is possible. For instance, if an individual prepares tea in the kitchen at 4 p.m., every day, we can detect an abnormality if some day, at 4 p.m., the individual is not in the kitchen.

The information is gathered in such closed spaces, in many ways. Infrared based small motion detectors, charge-coupled device based cameras, passing sensors, window open/close sensors and operation detectors that register usage of almost all electric appliances were used to create an "Intelligent House" wherein the participating resident's routine was recorded [Hara *et al.* (2002a)]. Participants carried RFID-based readers with them while residing in a house equipped with tags in every room, sensing for signals from those readers. This transmitter-receiver like setup was used to record the participant's location within the house [Hui-Huang Hsu and Chen (2009)]. Physical activities were detected by having the participants wear inertial measurement units and a heart-rate monitor whilst performing various activities [Reiss and Stricker (2012)].

Application software, installed in a portable device, that is designed to record real-time participant activity information was a data collection method chosen by the authors of Frank *et al.* (2011). The participant carried a mobile phone to have the software act like an accelerometer and magnetometer and give information such as the acceleration magnitude. of the participant who carried this device whilst performing various physical exercise-related activities [Frank *et al.* (2011)] [Frank].

For processing the collected data to analyze, authors of Derakhshan *et al.* (2007) proposed efficient data cleaning, dealing with redundancies and other data management features in RFID based systems. In a similar topic of generic research challenges and possible solutions of RFID data, authors of Wang and Liu (2005) have highlighted certain drawbacks related to the spatio-temporal nature of the data, large streams of incoming data, integration complexities, etc. They proposed an expressive temporal-based data modeling of RFID data, using a Dynamic Relationship ER Model (DRER), which they built as an integrated RFID data management system. Targeting a specific feature of RFID that limits it to being widely used in the business area of supply-chain management, authors of Lee and Chung (2008) proposed an approach for efficient query processing system in database management systems.

Considering data compression principles, authors of Gonzalez *et al.* (2006) proposed the RFID-Cuboid, a data structure for storing aggregated data in the RFID warehouse which consisted of three tables: (1) Information, which stored product information for each RFID tag, (2) Stay, which stored information on items that stay together at a location, and (3) Map, which stored path information necessary to link multiple stay records. For analyzing, they used compression strategies on the tables of the cuboid by aggregating at different abstraction levels. The authors stated that through their data structure, trend analysis, outlier detection, path clustering, etc. could be conducted efficiently.

All these works focus on optimizing the storage and data processing capabilities of the existing systems. We can safely assume that these systems continuously generate huge amounts of data in real-time challenging the storage, management and analysis capacities of current technologies. The above mentioned works have provided us an insight on the For real-time analysis, it is essential to have quick and efficient diagnosis of the situation. This is one of the motivations for our proposed approach where we have ensured that our proposed model is fast, scalable and space efficient. To achieve this, we looked primarily into the approaches taken to *analyze* the data collected in real-time environments. Specifically, we focused on the approaches taken to address the problem complexities of *environment* and *data*, as described in the previous chapter. The next section provides a brief review of these approaches.

## 2.2 Abnormal behavior detection

The various approaches proposed to detect abnormal behaviors can be roughly divided in three categories: approaches based on probabilistic models, non-probabilistic models and those using trajectory patterns to find anomalies. Figure 2.1 shows some of the approaches categorized correspondingly.

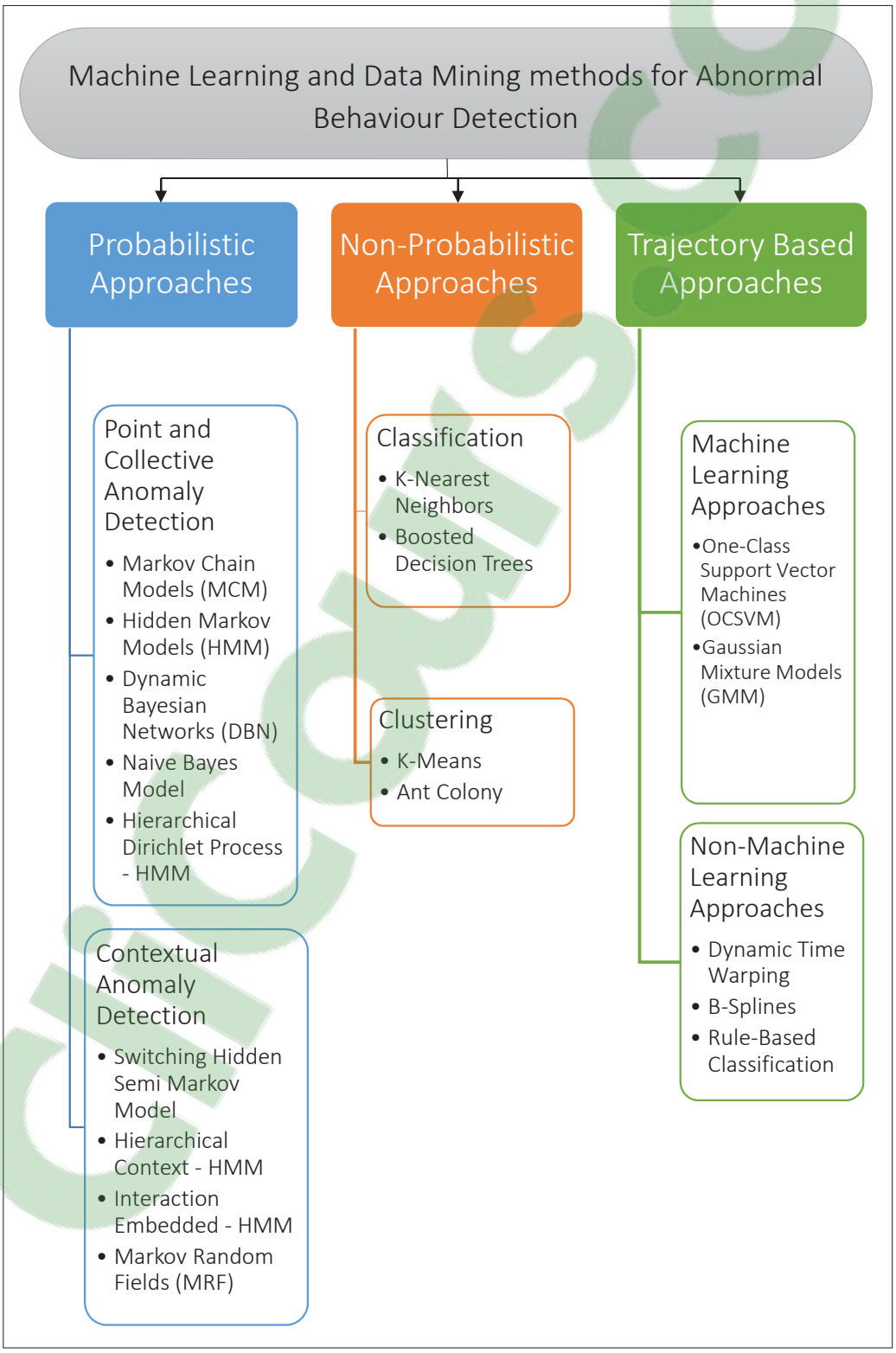



Figure 2.1 Some of the Machine Learning and Data Mining approaches for abnormal behavior detection

### 2.2.1 Probabilistic Approaches

In this category of approaches, a generative model is usually used to determine the likelihood of a sequence of observed events or actions. Sequences with low probability are considered as abnormal behaviors. Approaches based on Markov processes, such as Hidden Markov models (HMMs), as well as models based on Bayesian formulation, such as Dynamic Bayesian networks (DBNs), have been widely used for state model-based approaches. State model-based approaches are the sequential approaches that represent a human activity as a model composed of a set of states. In both cases, an activity is represented in terms of a set of hidden states. A human is assumed to be in one state at each time frame, and each state generates an observation (i.e., a feature vector). In the next frame, the system transitions to another state considering the transition probability between states. Once transition and observation probabilities are trained for the models, activities are commonly recognized by solving the evaluation problem. The evaluation problem is the problem of calculating the probability of a given sequence (i.e., new input) generated by a particular state-model. If the calculated probability is high enough, the state model-based approaches are able to decide that the activity corresponding to the model occurred in the given input [Aggarwal and Ryoo (2011)].

Hara *et al.* (2002b) aimed to detect unusual human behavior in an intelligent home using Markov process. The data was collected from three sources - a two-storied house, an apartment house and daily action record (DAR) of the participants. In the intelligent two-storied house, the occupant was monitored via infrared-based small motion detectors (SMDs), cameras, passing sensors, window open/close sensors and operation detectors that register usage of almost all electric appliances. The apartment house consisted of SMDs only whereas the DAR was a self-report for a period of two months of starting and ending times of sleep, restroom use, bathing, dining, and leaving and returning to the home. The DAR was intended to be a macroscopic description of daily human activity. The authors used vector quantizing method since the number of SMD signal states was too large, and thus defined clustered sequences of sensor patterns, which were modeled. They defined two probability distributions for defining human activity - state transition probability of sensor states based on MCM and duration time

distribution of Markov chain events acquired from starting and ending times of two successive states. The averaged log likelihood of the cluster sequence was computed given a duration time sequence. If either of the likelihood values were found to be below a threshold, the activity was labeled as unusual. From their results, the authors concluded that the likelihood measure can be applied to different houses and different types of data. Moreover, they also concluded that a human daily activity can be described through a probabilistic model.

In a work by Hu *et al.* (2009), a three phased approach for abnormal activity recognition is proposed. The approach initially applies Hierarchical Dirichlet Process (HDP) incorporated in a Hidden Markov Model (HMM). This allows an infinite number of states and the optimum number of hidden states is determined automatically. The second phase incorporates a Fisher kernel into the model with One-Class Support Vector Machine (OCSVM) used to filter out normal activities. Thus, they combine the generative nature of the HDP-HMM with the discriminative one of OCSVM. In the last phase, the abnormal activity model is derived in an unsupervised manner. They applied their approach on two real-world datasets consisting of activities such as Cleaning, yardwork, laundry, dish washing, etc. To detect abnormal activities, they manually labeled activities with low probabilities as abnormal and based their results upon detection of these activities. They also had a user simulate the effect of carrying out several abnormal activities in order to evaluate their approach. Their results, based on these experimental designs, demonstrate that their framework involving HDP and Fisher Kernel improves overall performance, empirically. The authors comment that by combining the generative and discriminative properties of HDP-HMM and OCSVM with Fisher Kernel into their framework, they achieved a good performance.

As mentioned earlier, apart from models based on Markov processes, probabilistic models based on Bayesian networks, such as Dynamic Bayesian Networks (DBNs) have also been applied for the task of detecting activities.

Du *et al.* (2006) used DBNs to recognize activities wherein human interactions took place. Their model consisted of three states: global activity state, which generated global features as-

sociated with activities in a large spatial scale or relations between people, etc.; duration state, which indicated the remaining duration of the global activity state; and local state, which generated local features associated with motion details or posture of each person. The authors said that the model including such three kinds of states could model an interacting activity, completely. The authors used this model to identify human interactions taking place in the parking lot environment using a digital video camera. They used five different kinds of interacting activities, for testing the model. Their results showed that DBNs gave a good performance by recognizing the interacting activities, more significantly, identifying the complex activity where two people approach and meet, one puts an object on the ground and goes away, and the other takes this object and goes away. The authors concluded that their approach gave a superior performance when compared to HMMs.

Several attempts have been made to include contextual information in a generative model. The following works have focused on integrating contextual information as various extensions to basic HMM framework.

A Switching Hidden Semi-Markov Model (S-HSMM) was proposed for activity recognition and abnormality detection in et. al. (2005). The authors added a semi-Markov extension to a 2-layered Hierarchical HMM. The design was such that for each top-level state, there was a corresponding state (considered as its child state) in the bottom-level in a way that a transition to a state in the top-level will initiate a Markov chain at the bottom-level over the states that are children of the top-level states. It is possible that two different parent states might share some common children. The dynamics and the duration parameters of the HSMM at the bottom layer were not time invariant, but were "switched" from time to time as determined by the switching variable (parent state) at the top. After defining the top and the bottom layers of the framework, the model was viewed as a Dynamic Bayesian Network (DBN) for learning method. They constructed the parameters of the DBN from the parameters of the S-HSMM in a way similar to Hierarchical HMM. The authors described that with this design, the bottom layer would capture the atomic activities such as spending time at the stove, fridge, etc. and the top level would represent a higher-level activity such as cooking, which comprised of the

atomic activities captured in the bottom layer. Thus, transitioning from one top-level state to another would represent sequences of high-level activities that are typical in a human daily routine. The authors applied the S-HSMM to identify activities in a home environment. The results showed a satisfactory performance and the authors concluded that for more complex domains, this model could be extended to a full-hierarchical HSMM. The authors stated that S-HSMM outperformed models like HSMM without information about activity hierarchy and a two-layer HMM without duration modeling. As the authors believed that both hierarchy and duration information are needed to build accurate activity models in home, S-HSMM proves superior in performance as compared to other models mentioned.

A Hierarchical Context Hidden Markov Models (HC-HMM) was proposed by Chung and Liu (2008). The authors described human behavior being composed of three components which are surrounding environment, human activities and temporal information. Based on these 3 components, the authors formed 3 corresponding reasoning modules – spatial context reasoning module (SCR), behavior reasoning module (BR) and temporal context reasoning module (TCR). They designed these modules into 3 layers of the HC-HMM model with SCR being the topmost, BR being in the middle and TCR being at the bottom. Additionally, a Duration-HMM (DHMM) was introduced as a layer between the SCR module and BR module with the intention of composing human activities without being affected by the variation of time duration of activities resulting from different people.

The authors implemented their model in a nursing home environment wherein they used video sequences from a week of daily life to train the parameters of the HC-HMM model. They defined 5 types of daily behaviors based on those sequences and considered abnormal situations as “faint”, “sleeping too long”, etc. by categorizing these situations as unreasonable activity under SCR, BR or TCR. The authors showed a high performance of this model for detecting all the defined abnormal behaviors. The authors stated that unlike HMMs that, in some cases, performed behavior recognition based on only data sequence representations with none, one or two out of the three above mentioned modules, HC-HMM considered all three modules which helped achieve a better performance for recognizing activities.

In a follow-up work, Chin-De Liu and Chung (2010), as another extension of HMM, Interaction-Embedded Hidden Markov Models (IE-HMM), was proposed for analyzing the interactions within a group of people. This IE-HMM framework included 3 main modules, which were Switch Control (SC) module, an Individual Duration HMM (IDHMM) module and an Interaction-Coupled Duration HMM (ICDHMM) module. The SC module was used to identify all the atomic behavior units within a scene, where each unit related to a particular human behavior, such as an individual behavior or a group interaction. IDHMM module was used to correctly classify an individual human behavior by analyzing both the activities that made up a behavior and their respective durations. Finally, the ICDHMM module was used to handle the requirements for identification and classification of a particular human interaction, which the authors outlined as analyzing the relative poses, motions and physical correlations of various participants in the interaction and also analyzing the duration for which the poses and physical correlations between the participants were maintained. In the architecture of this framework, various units were embedded within each of these modules. The SC module embedded components such as Interaction Detection Component (IDC) mechanism and Unit Extraction Component (UEC) mechanism to process the contents of a scene. The IDHMM module embedded multiple Duration HMMs (DHMMs) to identify the behavior corresponding to each individual behavior unit identified by the UEC Mechanism in the SC module. The ICDHMM module embedded multiple Coupled Duration HMMs (CDHMMs), wherein each CDHMM was responsible for recognizing a particular mode of group interaction. This model was implemented in the same environment as of Chung and Liu (2008). They tested it on situations where people were interacting within a single group with and without audio, situations where multiple groups were interacting concurrently and with changing compositions of the group interactions. The IE-HMM model gave satisfactory results by identifying activities in all the cases. The authors stated that IE-HMM is capable of recognizing both individual actions and multiple group interactions within the same scene. This model, thus, overcomes the drawbacks of layered HMMs and other similar models previously implemented for recognizing activities.

Contextual anomaly detection using a model based on Markov Random Field (MRF) has been attempted by Benezeth *et al.* (2009). Considering a complex environment with data taken in from video, the authors propose performing activity analysis and abnormal behavior detection on cluttered raw video or motion labels before, object extraction and tracking. They emphasize the importance of modeling the spatio-temporal correlations for improving false alarms, misses as well as detecting the abnormality of event sequences, such as a car making an illegal U-turn, person dropping a bag, etc. In this case, an event corresponds to movement detected at a given location and time in a video sequence. Their proposed approach involves learning, at first, the co-occurrence matrix which represents two active labels. The two active labels correspond to two pixels within a close spatio-temporal neighborhood. It can be thought of as a summary of every trace left by moving objects in the sequence. Hence, the co-occurrence matrix not only contains the average behavior across spatio-temporal observations, but also implicitly contains information about direction, speed and size of objects usually passing through one (or more) key pixel(s). A spatio-temporal neighborhood is a sub-video sequence. The authors model the likelihood of normal observations which are the ones constituting a spatio-temporal neighborhood. They do this by using the MRF model with co-occurrence matrix as one of the parameters. The formulation excludes stationary objects.The trained model describes the probability of observations within a same spatio-temporal context. To detect abnormal behaviors, the authors compare the likelihood of the observed sequence with the likelihood estimated by the model. Through experiments, they show detection of an illegal U-turn, pedestrian crossing on a busy street as well as a person dropping a bag and abandoning it. The shortcomings of this method arose when large sized objects, such as trucks, left huge group of pixels taking over the neighborhood that was being analyzed which made it difficult to identify neighboring pixels' co-occurrence. But the authors argue that with simple heuristics, such false anomalies can be eliminated. Hence, by encoding the co-occurrence probability of two events within a small spatio-temporal video, using MRF model, the authors could distinguish between normal and abnormal event in a complex environment.

### 2.2.2 Non-probabilistic approaches

Clustering algorithms were used for activity recognition, for identification as well as classification purposes. Grouping similar activity features together to highlight one particular activity was the idea used by [Huang *et al.* (2007) and Hui-Huang Hsu and Chen (2009)].

Ant colony algorithm is the simulation of cooperation course of real ant group. Each ant searches the solution independently in the candidate solution space, meanwhile leaves some information on the found solution. Ants leave more information on the solution with better performance, which has more possibility to be selected. All the solutions have the same information on it in the elementary period of the algorithm. With the progress of the computing, the better solution gets more and more information. Therefore, the algorithm reaches the optimization or approximately optimization solution [Zhang *et al.* (2006)].

In Huang *et al.* (2007), the authors chose this method for human posture estimation wherein they used the ant-colony algorithm to cluster useful training images by nearest Neighbor measure, and based on the established prototypes, they classified a new image according to the relative distance between new unknown images. They then used HMMs to train and recognize activities related to human motion. However, the authors concluded that in their experiment, the information of each moving object could not be extracted if the objects were near or obstructed by each other.

K-Means clustering technique was considered to overcome the problem of training the model from the data containing just the "normal" behavior by authors of Hui-Huang Hsu and Chen (2009). The authors chose to form clusters related to each behavior and assumed any new behavior, that didn't match the properties of the clusters within trained model, as an anomaly. The data was collected using RFID tags. The authors presented preliminary results wherein the model performed satisfactorily.

A behavior model to accurately analyze, predict and cluster social network of a population was proposed in [Eagle and Pentland (2009)]. The authors of this work focus on the principal

components of the complete behavioral dataset, calling the characteristic vectors as *eigenbehaviors*. The focus was on clustering by detecting groups of people sharing similar attributes and predicting an individual's day's remaining behaviors given the behavior until present time. It is an interesting work where the principal components are a set of vectors that span a behavior space and characterize the behavioral variation between each day. The eigenbehaviors are the eigenvectors of the covariance matrix of behavior data. The dataset used is the Reality Mining study, which is explained in section 4.1.2.1. The authors experimented to determine the behavioral similarity between individuals and groups, resulting in 90% classification accuracy of community affiliations. They also approximated an individual's behavior over a specific day by weighted sum of his/her primary eigenbehaviors. They calculated the weights halfway through a day and achieved 79% accuracy for the test subjects, on an average, by predicting the behavior of the next 12 hours given previous 12 hours of information. However, for this model to be efficient, behavioral regularity is important to get more accurate results.

Reiss and Stricker (2012) presented a physical activity monitoring dataset recorded from 9 subjects, wearing 3 inertial measurement units and a heart rate monitor, and performing 18 different activities. They defined four classification problems (tasks) based on the activities recorded. The tasks were: intensity estimation task, having activity-efforts classified as *light*, *moderate* and *vigorous*; basic activity recognition tasks, having classes such as *lying*, *sitting/-standing*, *walking*, *running* and *cycling*; background activity recognition task, having a class, *other*, which consisted of activities such as ironing, vacuum cleaning, ascending and descending stairs, Nordic walking and rope jumping, in addition to the ones from the previous task;and all activity recognition task, having all the 12 classes, *lying*, *sitting*, *standing*, *walking*, *running*, *cycling*, *ironing*, *vacuum cleaning*, *ascending stairs*, *descending stairs*, *Nordic walking*, *rope jumping*. The authors applied methods such as Decision Trees, Bagging Decision Trees, Boosting Decision Trees and K-Nearest Neighbors (KNN) to four classification problems. They also used a Naïve Bayes classifier in the same experiment for comparison. However, their results showed that the best performance was achieved by KNN and Boosted Decision Tree Classifiers.

But, the authors reported that on more complex classification problems, improved approaches should outperform their results.

### 2.2.3 Trajectory based approaches

A trajectory is a sequence of events corresponding to a single person/object observed within a time-frame. Finding frequent trajectory patterns is a very popular approach in spatio-temporal data. The approach for detecting abnormal behaviors involves representing behaviors as trajectories through space, and consider as anomalies the trajectories that are significantly different from commonly observed ones.

For example, in the work of Piciarelli *et al.* (2008), the authors chose to analyze the trajectories in video surveillance and traffic monitoring applications. The trajectories are of the moving objects from video sequences. They essentially take the outlier detection approach by clustering groups of trajectories sharing similar features and categorize the ones left out as anomalous. A One Class Support Vector Machine (OCSVM) model is used to learn the class of normal trajectories. The focus of the authors was initially on optimal estimation of the $\nu$ parameter for the model followed by independency from the optimal estimation. The latter focus was aimed to increase the computational efficiency of classification and improved detection since upon comparing their proposed method with OCSVM trained with $\nu = 0.2$, there was an increase in true outliers detection and decrease in false normal classification. They achieved this independency from optimal estimation of the $\nu$ parameter by removing, from the training set, the support vectors defining the boundaries of the classification function. This is based on the hypothesis that outliers are few in number and have an underlying distribution different from normal data. Hence, given a support region initially forced to include all data, its hypervolume in feature space would shrink rapidly when removing real outliers, while it would skrink slowly when removing normal data. So the proposed approach detected the point of change in shrinking speed and thus, led to an optimal support region discarding only real outliers. The computations were made in feature space by means of gaussian kernel. Because of this approach, only a single training was required for SVM, unlike their previous work. For experimental evalu-

ation, they tested their method on synthetic as well as real life dataset of pedestrian crossings with simulated anomalies. Their proposed method was method gave promising results based on their experiments.

In Jiang *et al.* (2011), the authors proposed a method to integrate contextual information into trajectories. Their notion of context was in terms of type of anomalies. They analyzed the trajectories to detect anomalies in context of point, sequence, interaction and co-occurrence. The first two involved analysis of a single entity (person/object) while the latter two involved analysis of multiple entities. The detection of interaction anomaly involves multiple objects with complicated temporal logic and was not evaluated. They focused mainly on point, sequential and co-occurrence anomalies. For all kinds of analysis, the authors considered the frequency (count) of observations corresponding to each entity's spatio-temporal behavior, which they called as an event. Hence, an event would be the entity observed at a particular location at a particular time. Events with high frequency were exclusively considered regular rules while those deviating from these rules were detected as anomalies. They conducted their experiments on traffic surveillance datasets where the zones were very clearly defined with respect to the rules such as direction (determined from spatio-temporal trajectory). They chose to analyze point anomaly based on a single event and sequence based on the trajectory covered in a single frame. For co-occurrence, they observed illegal moves as defined by traffic regulations and detected anomalous movements such as wrong direction, vehicle turning right while there is a left-turning traffic going to the same lane, vehicle turning left in front of incoming straight-going traffic. The approach is heavily dependent upon smoothness of the trajectories and the clear definition of the paths and corresponding rules. Hence, this approach is applicable mostly to monitor the traffic of cars on roads or the movements of pedestrians along clearly defined paths.

A sparse reconstruction analysis of trajectories is proposed in the work by Li *et al.* (2011). They represent trajectories as fixed-length parametric vectors based on control points of Least-squares Cubic Spline Curves Approximation (LCSCA). The representation is based on cubic B-spline curves which allows control points and weight factors, flexible enough to shape simple

as well as very complicated curves. Since trajectories are not always smooth, the strategy seems fitting. The authors use the B-spline control points to represent the shape as well as spatio-temporal profile of a trajectory. These temporal profiles are stored in a dictionary structure with parameters representing number of control points and the length of the trajectory. The sparse reconstruction analysis is done by reconstructing the test trajectory as a sparse linear combination of the dictionary templates of trajectory samples trained upon. The authors tested this approach on the CAVIAR dataset [Robert Fisher (2004)], which contains the series of behaviors in the entrance lobby of the INRIA laboratory. The authors considered traversal directions and aimed to detect abnormal behaviors such as people fighting, falling down and leaving or collecting packages. They measured accuracy and true positive rates. The accuracy was in the range of 80%-90% while the true positive rate was in the range of 60% - 70%. While the approach seems well-reasoned, the experimental evaluation proved it to be heavily reliant on the parameter values, more specifically, the number of control points. The authors commented the need for more robustness for their future work.

The above mentioned work was inspired from the work by authors of Sillito and Fisher (2008). They had chosen to encode trajectories using cubic B-splines parameterized by control points. However, the approach involved using Gaussian Mixture Models (GMMs) to learn the normal distribution and estimate the distribution of the observed trajectory. If the estimate was too low, the observation was categorized as an anomaly. They, too, experimented in a similar environment of car parking and tracking the paths between entrance/exit doorways in the above mentioned CAVIAR dataset [Robert Fisher (2004)]. The approach was incremental and involved a human in the loop to make the final decision regarding the observed trajectory being normal/abnormal. If it was found normal, the trajectory will be given to the model to update, thus, incrementally learning. The authors conclude that the proposed method provides no principled safeguards against anomalies being incorporated into the learning algorithm. However, if regarded as an alternative to an entirely unsupervised learning algorithm, it is clear that their method could at worst perform equivalently to such algorithms.

Authors of Lin *et al.* (2011) chose Dynamic Time Warping (DTW), a non-machine learning approach, for Analyzing human motions by tracking the feature points technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Intuitively, the sequences are warped in a nonlinear fashion to match each other [Müller (2007)]. Authors of Lin *et al.* (2011) recorded the motion trajectory, and then analyzed the features of trajectory by using DTW to calculate the score and show the similarity between the two trajectories. The data was a video capturing the yoga sports and the authors cut a clip from a film wherein the motion was raising both arms parallel to the floor. The trajectory consisted of points where each point represented a pixel within the digital image. DTW managed to produce a perfect score indicating the similarities and differences quite clearly.

### 2.2.4 Discussion

All these works mentioned above have presented efficient approaches involving machine learning and data mining methods for abnormal behavior detection. For contextual anomaly detection, probabilistic methods have proved to be more popular over non-probabilistic ones, while trajectory based ones were the most preferred if the data was spatio-temporal in nature. In our application, trajectory based methods will fail to give efficient performance since we cannot clearly define zones and assign corresponding rules. This is because there are a large number of zones observing a high volume of activities. Moreover, trajectory based methods mostly apply clustering strategies which rely on repetition of same patterns in the exact same manner. This is not possible when the environment is open and unstructured. A user with the same intention or performing the same action can have several different trajectories. Similarly, for contextual information analysis, some of the efficient probabilistic approaches, proposed in the literature, have several layers integrated in the model, each layer corresponding to a context. In order to incorporate additional contexts, the entire model design will need to be revisited.

For these reasons, we aim to analyze a higher-level latent information, that models complex relations without requiring clear definitions. The following section briefly introduces to the area of Latent Semantic Analysis using tensor decomposition. We look into some of the ap-

plications of tensor based methods in the field of machine learning and data mining as well as some of the popular tensor decomposition based techniques for abnormal behavior detection.

## 2.3 LSA with Tensor analysis

Latent Semantic Analysis (LSA) was first introduced in Dumais *et al.* (1988) and Deerwester *et al.* (1990) as a technique for improving information retrieval in documents. Most approaches to retrieving information depended on a lexical match between words in the user's query and those in documents. Due to that, upon searching, there was a tremendous amount of irrelevant information retrieved while failing to find much of the existing relevant material. The key insight in LSA is to reduce the dimensionality of this information retrieval problem. A technique from linear algebra called *singular value decomposition* was used to accomplish the dimension reduction. This unsupervised learning technique has been applied to many problems related to information retrieval, including text classification, text clustering, and link analysis. It appears to be especially useful for problems where input is noisy (such as speech input) and standard lexical matching techniques fail [Dumais (2004)]. In this project, we use this LSA theory to analyze the complex patterns in a large amount of data. We use tensors for representing the data, factorize it to extract latent information and retrieve the complex patterns from this latent information for analysis.

### 2.3.1 Introduction to tensors

Tensors are multidimensional arrays that represent the relations between different dimensional elements. They are usually viewed as a multidimensional or $N$-way array. More formally, an $N$-way or $N^{th}$-order tensor is an element of the tensor product of $N$ vector spaces, each of which has its own coordinate system. A first-order tensor is a vector and a second-order tensor is a matrix [Kolda and Bader (2009)].

A tensor that has three or more dimensions is called a higher-order tensor. An example of a third-order tensor $\mathcal{X}$, is shown in Figure 2.2 where $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

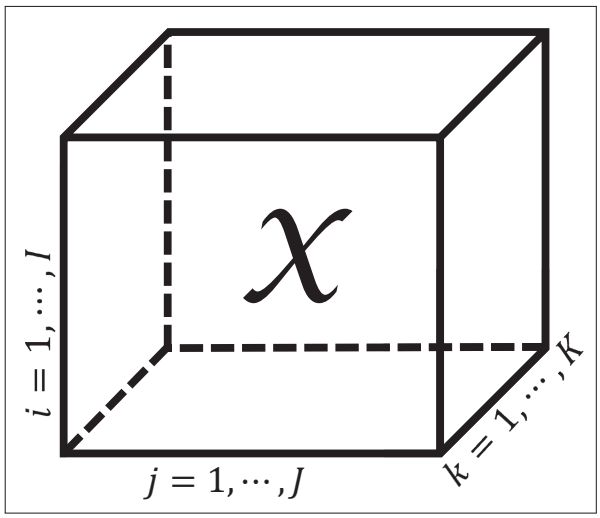



Figure 2.2 A third order tensor

### 2.3.2 Tensor Decomposition

Tensor Decomposition is an extension of matrix factorization to multi dimensional data. A dataset can be represented as a tensor by having each dimension represent an attribute. For our application, each cell within the tensor will contain the frequency of occurrences of specific events while each dimension will be providing a context. When this tensor is decomposed, we will have a summary of all contents into latent factors that will provide a high-level description of the dimensional relationships. For instance, a tensor with 3 dimensions: person, time and location can help describe the person's job or role in that particular environment, based on the frequency of observations at a particular location, for the duration (time-slot/shift). This is a high-level description of the dimensional context [Kolda and Bader (2009)].

There are several ways of achieving a decomposed tensor. They are based on the decomposition (or, factorization) of matrices, such as Singular Value Decomposition (SVD), Rank factorization, Cholesky decomposition, etc. [Oono (2012)]. The tensor decompositions that are mostly applied in data mining are:

- Canonical Decomposition (CANDECOMP) and Parallel Factors (PARAFAC) (also known as CANDECOMP/PARAFAC or CP), which decomposes a tensor as a sum or rank-one tensors.
- PARAFAC2, which is a variant of CP that can be applied to a collection of matrices that each have the same number of columns but a different number of rows.
- Tucker Decomposition, which is a higher-order form of Principal Component Analysis (PCA).

Amongst these, CP and Tucker decompositions can be considered to be higher-order extensions of the matrix SVD. All these factorization methods use the reconstruction error as the objective to minimize, and allow negative values in the tensor [Kolda and Bader (2009)].

For implementation convenience, we *matricize* a tensor (transform a tensor into a matrix) by *unfolding* across any one of the dimensions. Kolda and Bader (2009) explain that the unfolding is the process of reordering the elements of an $N$-way array into a matrix. For instance, a $2 \times 3 \times 4$ tensor can be arranged as a $6 \times 4$ matrix or a $3 \times 8$ matrix, and so on. The mode-$n$ matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathbf{X}_{(n)}$ and arranges the mode-$n$ fibers to be the columns of the resulting matrix.

### 2.3.3 Applications of Tensors in Data Mining so far

The application of tensor decomposition methods, in data mining, has varied from text analysis to network modeling. All the applications focus on extracting latent information for analysis by understanding the complex high-level relationships between the attributes that are often hidden or vague in other analytical methods.

Authors of Acar and Camtepe (2005) addressed the problem of discussion disentanglement in online chat rooms. They aimed to extract structure information without understanding the contents of chat messages, i.e. without semantic information to determine how many topics are discussed, or which chatters belong to the same conversation topics. The authors chose

to use three dimensional chat room tensors with user $\times$ keyword $\times$ time as the dimensions. They compared their approach with that of a SVD and proved that their tensor contained a multilinear structure which couldn't be detected by SVD. They also compared the performance of techniques, based on the basic decomposition techniques, such as Tucker1, Tucker3 and PARAFAC and found performance of Tucker3 as the best technique for their application.

In a follow up work [Acar *et al.* (2006)], they chose to use a collective analysis algorithm which consisted of constructing small tensors instead of a single one representing all the information. They applied decomposition techniques and found that their spatial complexities and computational cost reduced significantly for their chat room analysis.

In text analysis, Bader *et al.* (2008) applied a nonnegative tensor factorization algorithm to extract and detect meaningful discussions from electronic mail messages for a period of one year. They used the publicly released Enron electronic mail collection, wherein they encoded a sparse term-author-month array for subsequent three-way factorization using the PARAFAC decomposition method. Their results showed that nonnegative tensor factorization (implemented by PARAFAC) could be used to extract meaningful discussions from email communications. It helped assess the term-to-author (or term-to-email) associations both semantically and temporally via three-way decompositions.

To avoid having non-negative values in the tensor, a Non-negative Tensor Factorization (NTF) approach, called Positive Tensor Factorization (PTF), was proposed by Welling and Weber (2001). This approach is an extension of the well-known non-negative matrix factorization (NMF) algorithm presented in Lee and Seung (2001). The authors initialized the factors to random positive values and proposed multiple update rules and reconstruction error formulations to have the factors learn the tensor based on mimization of the reconstruction error. They even mentioned two generalizations for which, their formulated update rules could be derived. To explore the potential applications of their proposed PTF model, they conducted three experiments. All these experiments consisted of analyzing four neoplastic (Mondiran) paintings. They cut out patches of a fixed size and organized those patches into a data matrix. They ap-

plied PTF and SVD to reconstruct the palette and found that PTF, since the factors are positive, they are easier to interpret as it produces sparser features and also allows extraction of multiple groups. The motivation for having a positivity constraint was for analyzing intrinsically positive data such as pixel values of color images, where a negative value cannot be used to define a proper color scheme.

Another work motivated the need for having positivity constraint in the tensor to perform analysis of count data. This work, by Chi and Kolda (2012), highlights the representation of count data as a Poisson distribution, especially, when the counts are sparse. They stated that a Poisson model, compared to a Gaussian, is a much better explanation for the zero observations that we encounter in sparse data as these zeros just correspond to the events that were very never/unlikely to be observed. A factorization approach called Alternating Poisson Regression (APR) was proposed for their problem of multilinear modeling of *sparse count* data. This approach was based on non-negative Poisson regression. Several layers of complexity were added inorder to make the method non-negative. They also proposed minimizing Kullback-Leiber (KL) divergence instead of the Least squares error function, for count data. They applied this model on the ENRON email dataset [Bader *et al.* (2008)] by creating a three dimensional tensor with sender $\times$ receiver $\times$ month as dimensions. Since there cannot be any negative value for the emails sent/received, the data is strictly a count data and sparse in nature. Through their factorization model, they managed to analyze the components describing the gender, job profile, department, etc. They performed similar analysis on the SIAM publication dataset containing author, year and journal. The method seems apt for factorizing count data but is quite complex to comprehend.

### 2.3.4 Anomaly detection using tensor factorization

A few studies have used tensors to detect anomalies. Sun *et al.* (2006a) proposed Window-based Tensor Analysis (WTA), a tensor decomposition method involving Tucker Decomposition. This opened doors to analyzing highly-dimensional data coming in streams. It allowed finding patterns within and across multiple contexts, simultaneously, in an incremental stream.

WTA summarized the tensor windows efficiently, using small core tensors associated with different projection matrices, where core tensors and projection matrices were analogous to the singular values and singular vectors for a matrix. Two variations of the algorithms for WTA were proposed: 1) Independent-window tensor analysis (IW) which treated each tensor window independently and 2) Moving window tensor analysis (MW) which exploited the time dependence across neighboring windows to reduce computational cost. The authors also proposed Multi-Aspect Correlation Analysis (MACA), which simultaneously found correlations within a single and across multiple aspects. They performed their case studies on the Sensor dataset which comprised of data collected from 52 MICA2 Mote sensors placed in a lab, over a period of a month wherein every 30 seconds each Mote sensor sent measurements such as light intensity, humidity, temperature and battery voltage to the central collector via wireless radio. Their proposed methods showed efficient and successful results.

A work on streaming data was proposed by Sun *et al.* (2006b). Here, they propose a Dynamic Tensor Analysis (DTA) and its variants along with Streaming Tensor Analysis (STA) method. They apply those methods for anomaly detection and multi-way Latent Semantic Indexing (LSI). They conducted their experiments on three datasets that represented network flow of data in 2D and 3D and DBLP, which contains bibliographic information on major computer science journals and proceedings. Their proposed DTA and STA incrementally mine and summarize large tensors, saving space and detecting patterns. The authors also claimed that DTA and STA are fast, nimble and fully automatic, without requiring any user-defined parameters.

The authors of Tork *et al.* (2012), used three-dimensional tensor to detect events and anomalies in time-evolving network data. They created two versions of the same data, each spanning 720 hours. In one they chose 102 users and in the other they selected 909 users, where, the former 102 users were included, too. They used Tucker3 decomposition to factorize three-dimensional tensors which modeled users $\times$ features $\times$ hours. Features corresponded to the processed time-series information i.e. number of messages sent per user with time-difference between two consecutive messages. They factorized the tensor using the tucker3 decomposition method, which is a generalized version of two-mode SVD. Upon decomposition using this method, they

obtained 3 matrices, each corresponding to a dimension plus a small core tensor containing the residual errors. They set the latent factor vector size to three so they could plot and view clusters of similar user-behavior. For a better statistical analysis, they computed the Euclidean distance between each user in the projection space and the distances were normalized. Abnormal users were the ones farthest from the center (origin) of the projection space, based on Euclidean distance. They also ranked the users based on their distance and set a threshold to determine abnormal users from normal. So, essentially, the authors chose the decomposition method to cluster similarly behaving users and detect the outlying ones as abnormal users.

A new tensor factorization model was proposed by Hayashi *et al.* (2010), which can manage heterogeneously attributed data by employing an individual exponential-family distribution for each attribute of the tensor. Since the inference process is intractable, a strategy combining the Expectation Maximization (EM) algorithm and the Laplace approximation is presented. Specifically, the authors generalize the likelihood of the Tucker decomposition to model non-Gaussian observations such as heterogeneously attributed array with both real and discrete variables. They state amongst multiple advantages of having an exponential distribution, non-negativity is especially useful. An individual exponential-family distribution, such as Bernoulli, Poisson, etc., is assumed for each attribute on the array data to deal with heterogeneity. Latent variables capture the noise-corrupted heterogeneous array data into a unified lower dimensional parameter space. They use Expectation Maximization (EM) algorithm for parameter estimation. The key parameter is the specification of the distribution corresponding to each attribute. Choosing wrong distributions decline the performance of the model. The authors comment that when statistical properties of the data are completely unknown or the distributions of all the attributes are not members of the exponential family, standard methods assuming Gaussian distribution should be more plausible compared to this proposed approach. However, the proposed methods is a computationally efficient for estimating the factors and this allows derivation of a Bayesian predictive distribution for missing elements as well as anomaly detection. The authors experimented by applying their model to detect anomalies in office-logging data, by considering the dimensional elements (e.g., persons or days) which are

distant from other elements of the same type in the latent factor subspace. The method gave a good performance for the count data but not for real-valued data.

In Kim *et al.* (2009), a Higher-Order Principal Component Analysis (HO-PCA) algorithm, which is based on HO-Singular Value Decomposition (HO-SVD) and HO Orthogonal Iteration (HO-OI). This algorithm is used to perform anomaly detection on traffic data of large computer networks. It is also applicable on smaller sized networks but the improvement is not as significant in comparison to PCA. The tensor represents the network topology which is abstracted from a directed graph, consisting of nodes and links. A traffic matrix, $X = N \times N$ is merely a link information (in terms of weight) between two nodes.The tensor structure allows incorporation of temporal information. Hence, the tensor consists of the traffic matrix with time-series of these matrices ($N \times N \times T$). The decomposition of this tensor, using the proposed algorithm, divides the data into a low-rank subspace which approximates the normal data, and a residual subspace encoding the anomalies. The magnitude of the signal along the residual subspace is thresholded to detect anomalies. Through their experiments, the authors showed that their HO methods outperforms two-dimensional matrix SVD/PCA in terms of probabilities of false alarm and miss. They also demonstrate the scalability and low-complexity of their HOOI implementation.

In a work by Koutra *et al.* (2012), the authors introduce a real-time anomaly detection model, called TensorSPLAT, where, SPLAT stands for Spotting Latent Anomalies in Time. The core decomposition method used is the parallel factors (PARAFAC) decomposition. The approach involves thresholding small values that have little interest in the context of anomalies and thus, performing a truncated tensor decomposition. Among the most interesting components of the decomposition, the authors pick the most interesting ones. From their description, we understand the definition of "interesting" as an outlier, which could be a rare, anomalous or strange observations. Through this approach, they proposed applications for change detection over time, anomaly detection and clustering. They conducted their experiments on three datasets: facebook dataset representing frequency of wall posts by connections, network datasets containing bot attacks and bibliography datasets representing author-conference publication de-

tails. On the bibliography dataset, they presented analysis such as a researcher changing fields based on types of conferences/journals publication, authors who collaborate often and authors publishing in multiple disciplines. They detected events such as birthdays by monitoring the frequency of wall posts on a person's facebook profile. They also detected malicious attacks on the network data spanning 1 hour. But, the authors commented that the approach is heavily dependent on the choice of the components deemed interesting. In this work, the authors manually chose to pick interesting components and intend to provide an automated detection for change, anomalous and rare events, in future. The requirement for human intervention in this manner is a major drawback for this approach but it manages to prove the use of tensor decomposition for anomaly detection.

## 2.4 Summary

In this chapter, we looked into various data management, popular machine learning and data mining approaches for abnormal behavior detection as well as works done in relation to using tensor decomposition for Analyzing latent information. Through these works and considering our application domain, we were impelled to integrate some key ideas that demonstrated the efficacy for abnormal behavior detection. These key ideas foremostly included having a probabilistic approach modeling contextual information. Following that, analyzing a trajectory as a set of discrete events seemed to couple with representing these events as a count data. Moreover, having an event's probability estimated in real-time led to requirement of fast computation with minimum post-processing.

In the next chapter, we will provide a detailed explanation of how we went about building a model incorporating the above mentioned ideas.

# CHAPTER 3

# LOG-LINEAR TENSOR FACTORIZATION

In this chapter, we'll describe our proposed Log-Linear Tensor Factorization (LLTF) model based approach. We initially give a brief description with a flowchart of the general overview of our approach. This is followed by a detailed description of the LLTF model for training the latent factors and corresponding pseudocode. Finally, we describe the online testing by estimating the joint as well as conditional probabilities of the events observed in real-time scenarios.

## 3.1 General Overview

Given the aim of Analyzing *contextual information*, we use tensors for the flexibility of allowing as many dimensions deemed necessary to analyze the event within the given environment. To manage the space and computational complexities, we take inspiration from the concept of LSA; factorize the event tensor and analyze the latent factors. By having LLTF method at the core of our approach, we are able to get latent factors that have learned the empirical probabilities of the observed events and can estimate the joint as well as conditional probabilities of the new events coming in real-time. The flow-chart in figure 3.1 provides an overview of our approach.

Through this flowchart in figure 3.1, we describe our approach in three phases:

a. **Data representation:** Gathering the data from the environment and representing the events as a tensor.

b. **Training:** Batch learning of observed events using LLTF method.

c. **Testing:** Online estimation of joint or conditional probabilities leading to detection of abnormal behaviors.

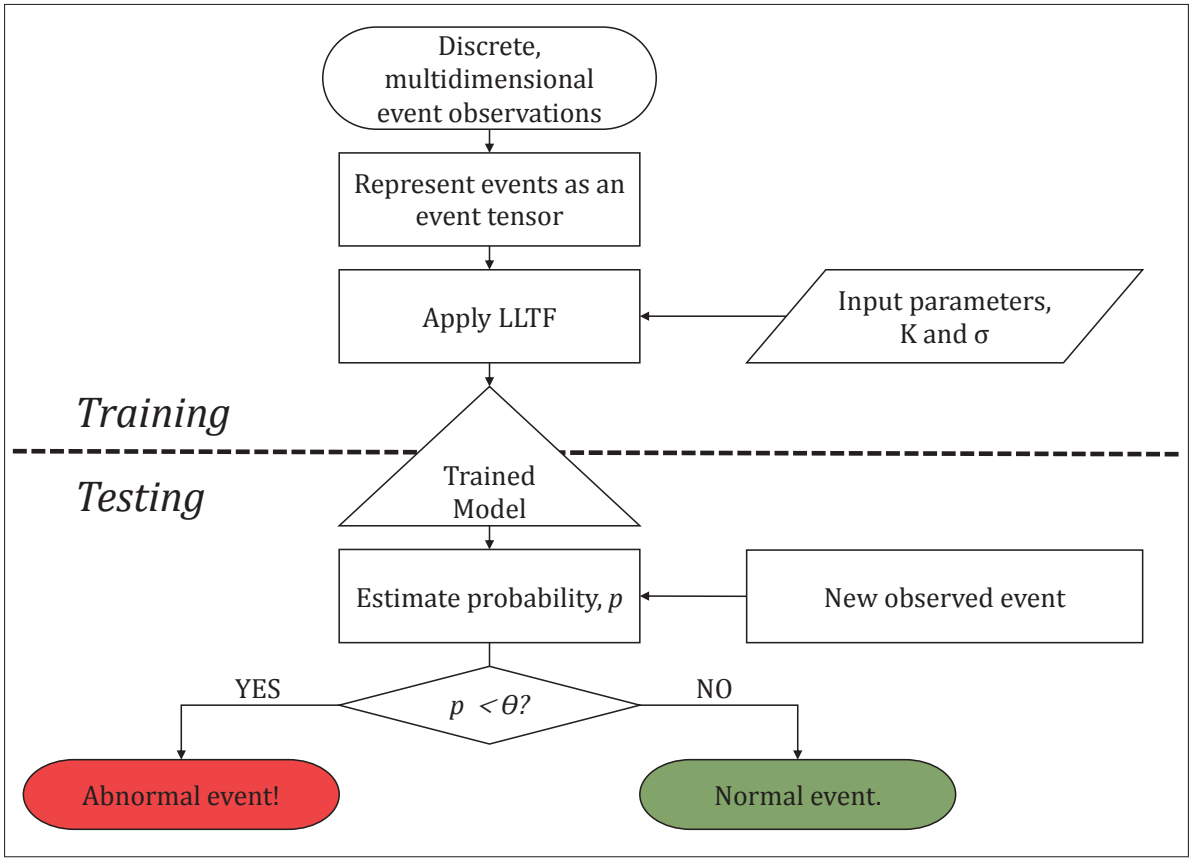



Figure 3.1 Overview of the proposed framework

### 3.1.1 Data representation

For building our framework, we received the information containing real-life events and incidences based on which, we could build a model. This information, ideally, is expected to consist of normal day-to-day activities as well as rare-event situations in a large, open and dynamic environment which we'll identify as abnormal. Hence, for initial experimentation, synthetic and publicly available datasets are used. Ideally, the publicly available datasets are expected to contain spatio-temporal information of people or objects, with features/attributes similar to the dataset expected in a real-life application.

The data, thus, received is expected to be multi-dimensional with each dimension corresponding to a context that will help us make an informed analysis. We define an event as a single

observation across all dimensions. For example, if the contexts are {*person*, *hour of the day* and *location*}, in a university surveillance, an event could be {*John*, *11am*, *classroom*}. Based on this example, we represent the information in a tensor with three dimensions, where,

- Dimension #1 represents all the people being tracked;
- Dimension #2 represents all the 24 hours of the day;
- Dimension #3 represents all the locations being observed.

Given this, the event tensor will be of dimensions Total #people/objects $\times$ 24 (total #hours of the day $\times$ Total #of locations/zones. Now, since we represent count data which corresponds to the frequency of observing a particular event, an observation such as {*John*, *11am*, *classroom*} would merely lead to an increment in the value in this tensor where *John* is in dimension #1, *11am* is in dimension #2 and *classroom* is in dimension#3. The representation is depicted in figure 3.2

Generalizing the above mentioned example, we suppose that the observed data is a stream of discrete events, each event representing the co-occurrence of $D$ dimensional values. The combination of such values constitutes the event context. We model the multi-dimensional context of an event using a set of discrete random variables $\{X_1,\ldots,X_D\}$, each variable $X_j$ having a domain $\Omega_j = \{1,\ldots,N_j\}$. We write $X_j = i_j$ the observation of value $i_j \in \Omega_j$ for dimension $j$, and use $x_{i_j}$ as shorthand notation for this observation. For example, if the first dimension represents people, then $x_{i_1}$ means the observation of person $i_1$, from a group of $N_1$ people, in the event.

Assuming that a set $\mathcal{X}$ of $M$ observed events $(x_{i_1},\ldots,x_{i_D})$ is available, we represent that set as a $D$-dimensional event tensor. In this event tensor, element $(i_1,\ldots,i_D)$ contains the number of events of $\mathcal{X}$ having context $(i_1,\ldots,i_D)$. Figure 3.2 shows the representation of the events as an event tensor.

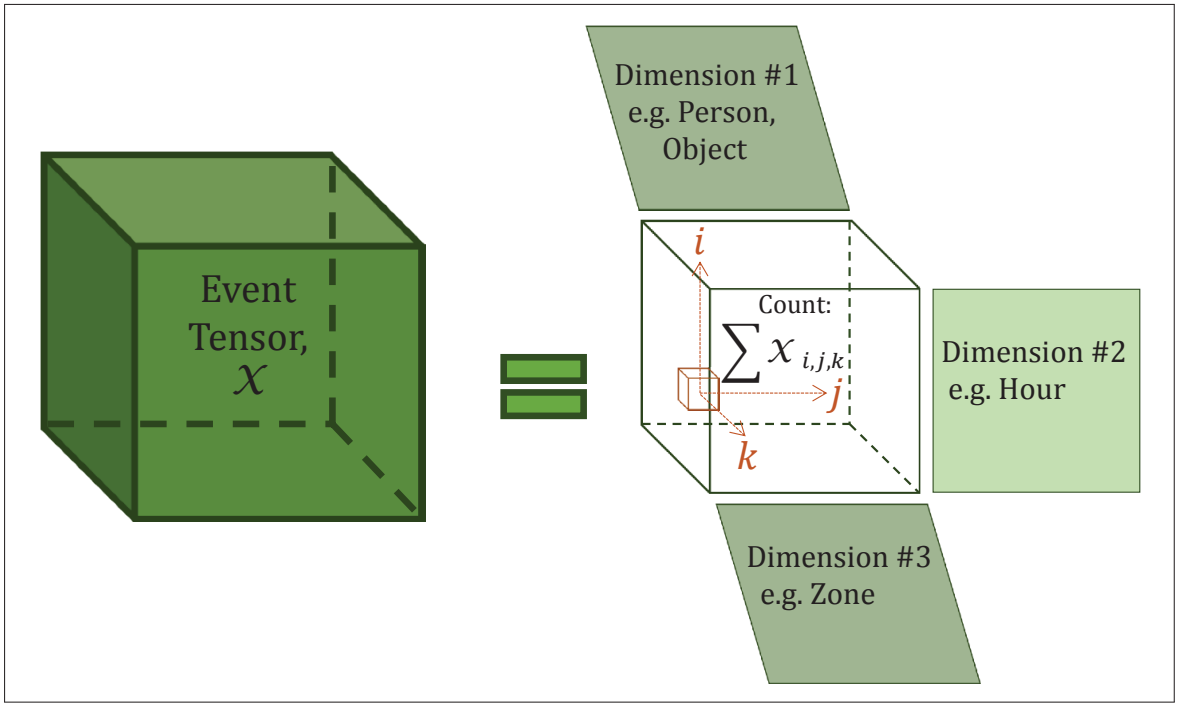



Figure 3.2 Data representation as en event tensor

For example, we organize the received data as follows:

$$\begin{array}{ll} (1,1,7) & 13 \\ (1,2,16) & 17 \\ (2,8,16) & 13 \\ (2,12,7) & 18 \\ (3,12,21) & 3 \end{array}$$

Here, dimension # 1 represents id of the object, dimension # 2 represents hour of the day and dimension # 3 represents the id of the zone. A single record is the observed event which is the co-occurrence of the dimensional values. The corresponding observation frequency of the event is indicated in the column to the right. This means that in the entire duration of the surveillance shown in the above example, person/object with id # 1 was observed at 1am on zone id'ed as # 7, 13 times. The same person/object was also observed at 2am on zone id'ed as # 16, 17 times.

We assume the knowledge of the total number of ids being tracked and the total number of zones. So, the dimensions of the event tensor are fixed sized and the values correspond to the observation frequency. Because only a few events are observed, in comparison to many possible combinations of dimension values, such tensor is normally very sparse. To overcome sparsity and extract semantic information (latent factors), tensor factorization is used.

The next subsection presents the training phase of our anomaly detection framework, using LLTF model.

### 3.1.2 Training

Given the complexities of data and environment, we have motivated our reasons to analyze the high level information. Since probabilistic approaches have proved efficient in determining the normality of an event we chose to model the empirical joint probability distribution of the observed events. This joint distribution of events context values represents what constitutes normal events (in terms of their context). Through some review of related works, we have understood that most tensor factorization methods involve three key elements:

a. **The latent factor vectors** that learn the data, represented as a tensor. They are finite-sized vectors associated with every dimensional value;

b. **The cost function**, also known as the objective function, which we want the model to optimize. Some examples for cost functions include minimization of the reconstruction error of alternating least squares, maximizing the posterior/likelihood, etc.;

c. **Optimization strategy**, which is the technique used by the method to determine if the latent factors have learned the data by optimizing the cost function. One of the popular optimization strategies is the *gradient descent*.

The detailed formulation of the proposed LLTF model is presented in section 3.2 while an overview is as follows.

Given an event tensor containing the count data representing frequency of observing multi-contextual events, we aim to learn the distribution. Such an approach is called density estimation. The latent factors are to learn a supposedly sparse tensor with very few non-zero values. Therefore, they are initialized with random values as independent and following zero-mean normal distribution. We provide the size of the latent factor vectors.

The cost function is derived using the *Bayesian Maximum Likelihood Estimation* approach. We aim to maximize the likelihood of the posterior probability of the factors given the observed training events tensor. We also evaluate the fitness of the model by measuring the Root Mean Squared Error (RMSE). For computational tractability, we use a log-linear formulation and aim to maximize the log-likelihood. As we are dealing with probability values, the log-likelihood measure gives us negative numbers. So, we take the negative of the log-likelihood measure, which gives us a positive number, and call it the Negative Log-Likelihood (NLL). Therefore, our cost function is formulated to minimize the NLL value, which could also be viewed as minimizing the KL divergence between the empirical distribution and the model.

For optimization of the model parameters, we can use the gradient descent method. But, in the proposed LLTF method, we have used Nesterov's accelerated gradient descent method, which is significantly faster than ordinary gradient descent. The optimization strategy is quite straightforward. We compare the empirical distribution and the model. If upon approximation, the NLL value is too high, we update the factor values using Nesterov's accelerated gradient descent. We then compare the empirical distribution again, but this time, with the updated model. This process is continued until the NLL value is low enough and has converged until the factor values no longer differ much from the previous values. We prevent overfitting/underfitting by regularizing the solution.

The trained model consists of the converged latent factor values. This model can now be used to estimate the joint as well as conditional probabilities of the events coming in real-time. The testing is as explained in the next subsection.

### 3.1.3 Testing

The testing simulates the real-time application scenario of the proposed anomaly detection framework. We expect an event observed in the format as in the example of section 3.1.1, which is a vector containing a value corresponding to each contextual dimension. Given this, we can evaluate the joint or conditional probability of this observed event, using the parameters of the trained model. A threshold can be applied to classify the probability values as normal/abnormal based on the probability value being high/low.

## 3.2 The proposed tensor factorization model

Recall from section 3.1.1 that the observed data is a stream of discrete events, each event representing the co-occurence of $D$ dimensional values constituting the event context. Presuming that the observation of events depends on a set of latent factors $\mathcal{Z} = \{Z_1, \ldots, Z_D\}$, providing high-level information about the dimension values, we define the latent factor vector corresponding to the $i_j$-th value of dimension $j$ as $z_{i_j} \in \mathbb{R}^K$. Using the example in section 3.1, $z_{i_1}$ would be the latent factor vector of person $i_1$. The latent subspace dimension $K$ is a user-supplied parameter.

We propose modeling joint probability of dimensional values as the following non-linear model:

$$p(x_{i_1}, \ldots, x_{i_D} \,|\, \mathcal{Z}) \;=\; \frac{\exp\left(\langle z_{i_1},\, z_{i_2},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle\right)}{\sum_{i'_1=1}^{N_1} \ldots \sum_{i'_D=1}^{N_D} \exp\left(\langle z_{i_1},\, z_{i_2},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle\right)}, \tag{3.1}$$

where $\langle z_{i_1},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle$ is the inner product between $D$ vectors of $K$ dimensions:

$$\langle z_{i_1},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle \;=\; \sum_{k=1}^{K} z_{i_1,k} \otimes z_{i_2,k} \otimes \ldots \otimes z_{i_{D-1},k} \otimes z_{i_D,k}. \tag{3.2}$$

To learn the model parameters, we suppose that a set $\mathcal{X}$ of $M$ observed events $(x_{i_1}, \ldots, x_{i_D})$ is available. This set can be also represented as a $D$-dimension tensor, in which element $(i_1, \ldots, i_D)$ contains the number of events of $\mathcal{X}$ having context $(i_1, \ldots, i_D)$. We call this structure the *event tensor*.

For example, assuming, in a three-dimensional tensor $x_{i_1}$ = person, $x_{i_2}$ = 12pm, $x_{i_3}$ = zone of cafeteria, we'll count the number of times person x has been detected in the cafeteria at 12pm, in our training set. If, say, person x has been to the cafeteria at 12pm for 23 times, $\mathcal{X}_{i_{1,2,3}} = 23$.

### 3.2.1 Learning the empirical probability through factors

The latent factors $\mathcal{Z}$ can be found using the *maximum a posteriori* (MAP) estimate:

$$\begin{aligned} \mathcal{Z}_{MAP} &= \arg\min_{\mathcal{Z}} \; p(\mathcal{Z} \,|\, \mathcal{X}) \\ &= \arg\min_{\mathcal{Z}} \; \frac{p(\mathcal{X} \,|\, \mathcal{Z})\, p(\mathcal{Z})}{p(\mathcal{X})}. \end{aligned} \tag{3.3}$$

Considering the events as i.i.d., the observation likelihood of discrete events corresponds to

$$\begin{aligned} p(\mathcal{X} \,|\, \mathcal{Z}) &= \prod_{(x_{i_1}, \ldots, x_{i_D}) \in \mathcal{X}} p(x_{i_1}, \ldots, x_{i_D} \,|\, \mathcal{Z}) \\ &= \prod_{i_1=1}^{N_1} \cdots \prod_{i_D=1}^{N_D} p(x_{i_1}, \ldots, x_{i_D} \,|\, \mathcal{Z})^{M_{i_1, \ldots, i_D}}, \end{aligned} \tag{3.4}$$

where $M_{i_1, \ldots, i_D}$ corresponds to the number of events of $\mathcal{X}$ with context $(i_1, \ldots, i_D)$. Since the number of events is small compared to the size of the multi-dimensional event space (i.e. $N_1 \times \ldots \times N_j$), only a few of these values are expected to be non-zero. In other words, the event tensor should be very sparse. To regularize the solution, we suppose the latent factors vectors as independent and following a zero-mean normal distribution with uniform variance:

$$p(\mathcal{Z}) = \prod_{j=1}^{D} \prod_{i_j=1}^{N_j} \mathcal{N}\left(z_{i_j}; 0, \sigma^{-1} I\right). \tag{3.5}$$

The latent factors $\mathcal{Z}$ are found using the *maximum a posteriori* (MAP) estimate. Therefore, to find $\mathcal{Z}_{MAP}$, we maximize the log-posterior. Ignoring the evidence $p(\mathcal{X})$ and other constants, this is equivalent to maximizing the following function:

$$\begin{aligned} f(\mathcal{Z}) &= \log p(\mathcal{X} \mid \mathcal{Z}) + \log p(\mathcal{Z}) \\ &= \sum_{i_1=1}^{N_1} \dots \sum_{i_D=1}^{N_D} M_{i_1,\dots,i_D} \log p(x_{i_1}, \dots, x_{i_D} \mid \mathcal{Z}) \\ &\quad - \frac{\sigma}{2} \sum_{j=1}^{D} \sum_{i_j=1}^{N_j} ||z_{i_j}||^2. \end{aligned} \tag{3.6}$$

### 3.2.2 Optimization

Since the cost function of Eq. (3.6) is both non-linear and non-concave, obtaining globally optimum parameters is an intractable problem. Therefore, we can optimize it using an iterative approach like the gradient ascent method, like as follows:

Let $R_{i_1,\dots,i_D}(\mathcal{Z})$ be the difference between the observed number of events in context $(i_1, \dots, i_D)$ and the expected one according to parameters $\mathcal{Z}$:

$$R_{i_1,\dots,i_D}(\mathcal{Z}) = M_{i_1,\dots,i_D} - M \cdot p(x_{i_1}, \dots, x_{i_D} \mid \mathcal{Z}). \tag{3.7}$$

The gradient with respect to $z_{i_j}$ is given by

$$\frac{\partial f}{\partial z_{i_j}}(\mathcal{Z}) = \sum_{i_1=1}^{N_1} \dots \sum_{i_{j-1}=1}^{N_{j-1}} \sum_{i_{j+1}=1}^{N_{j+1}} \dots \sum_{i_D=1}^{N_D} R_{i_1,\dots,i_D}(\mathcal{Z}) \cdot \hat{z}_{i_j} - \sigma z_{i_j}, \tag{3.8}$$

where $\hat{z}_{i_j}$ is defined as the Hadamard (i.e., element-wise) product of all factor vectors expect the one of dimension $j$:

$$\hat{z}_{i_j} = \left(z_{i_1} \otimes \dots \otimes z_{i_{j-1}} \otimes z_{i_{j+1}} \otimes \dots \otimes z_{i_D}\right), \tag{3.9}$$

Starting from randomly initialized latent factor vectors, each GA iteration modifies these vectors as follows:

$$z_{i_j}^{(t+1)} \leftarrow z_{i_j}^{(t)} + \eta \frac{\partial f}{\partial z_{i_j}}(\mathcal{Z}^{(t)}), \tag{3.10}$$

where $\eta > 0$ controls the step size.

This approach has a rate of convergence of order $1/t$ after $t$ steps i.e. convergence rate is linear. However, because the cost function of Eq. (3.6) is also concave with respect to *each* factor vector, we can instead use Nesterov's accelerated gradient method Nesterov (2013) which attains a rate of order $1/t^2$ i.e. convergence rate is quadratic.

Unlike gradient ascent, Nesterov's method performs two different steps at each iteration. The first step is a simple gradient ascent step of size $\eta$ from the current solution $z_{i_j}^{(t)}$ to a intermediate solution $y_{i_j}^{(t+1)}$:

$$y_{i_j}^{(t+1)} = z_{i_j}^{(t)} + \eta \frac{\partial f}{\partial z_{i_j}}(\mathcal{Z}^{(t)}). \tag{3.11}$$

We perform gradient ascent as explained in equation 3.8. Following that, the second step, as per Nesterov's method, then finds the next solution $z_{i_j}^{(t+1)}$ as a convex combination of the two last intermediate solutions:

$$z_{i_j}^{(t+1)} = (1-\gamma_t) y_{i_j}^{(t+1)} + \gamma_t y_{i_j}^{(t)}, \tag{3.12}$$

where, $\gamma_t$ are constants controlling the search momentum (e.g. see Algorithm 3.1) . In other words, the first step performs a simple gradient ascent from $z_{i_j}^{(t)}$ to $y_{i_j}^{(t+1)}$, and then it 'slides' a little bit further than $y_{i_j}^{(t+1)}$ in the direction given by the previous point $y_{i_j}^{(t)}$. [Bubeck (2013)] Moreover, authors of Sutskever *et al.* (2013) explain that while ordinary gradient ascent requires the step size/learning rate $\eta$ to be pre-defined and fixed, Nesterov's accelerated gradient ascent uses the learning rate $\eta_t$ which is adapted to always be smaller than the reciprocal of the "observed" gradient around the trajectory of the optimization.

The integration of this Nesterov's accelerated gradient method in our proposed model for optimization along with the adaptation of the $\eta$ parameter is explained through a pseudo-code presented in the next subsection. Ideally, we continue iterating until the stopping criteria of a converged log posteriori is reached.

### 3.2.3 Algorithm summary and pseudocode

The complete inference process is summarized in Algorithm 3.1. For a greater efficiency in a Matlab-implementation, we group latent factor vectors of each dimension $j$ in a single matrix $\mathbf{Z}_j$, and use standard matrix operations. We start by initializing the factor matrices randomly following the prior distribution of parameter $\sigma$ (lines 2-5). Then, at each iteration of Nesterov's method, the tensor $\hat{\mathcal{X}}$ of expected counts is reconstructed using the current intermediate factors $\mathbf{Y}_j$ (line 8). Operator $\odot$ corresponds to the Khatri-Rao product and $\mathbf{X}_{(j)}$ denotes the unfolding of tensor $\mathcal{X}$ along dimension $j$ (see Kolda and Bader (2009) for more information). For each dimension $j$, the gradient is then computed using the residual between the observed and expected counts (line 12), and used to update factor matrix $\mathbf{Z}_j$ (line 13). If the step size $\eta$ is too large, the solution may diverge. A strategy is thus added to detect such problem and adjust $\eta$ automatically (lines 14-15), thereby eliminating the need to tune $\eta$ manually. This strategy is known as *backtracking line search*. The momentum constant and intermediate factors are finally updated as per Nesterov's method (lines 18-21). The process is repeated until converge is attained, or a maximum number of iterations is exhausted.

### 3.2.4 Computational complexity

The computational complexity of this algorithm is as follows. For each iteration, reconstructing the expected tensor $\hat{\mathcal{X}}$ takes $O(K \cdot \prod_{j=1}^{D} N_j)$ operations.

Likewise, updating the latent factors for each dimension can be done in $O(K \cdot \prod_{j=1}^{D} N_j)$. Therefore, the total complexity is $O(T_{\max} \cdot K \cdot D \cdot \prod_{j=1}^{D} N_j)$, where $T_{\max}$ is the maximum number of iterations.

Algorithm 3.1: Parameter inference using Nesterov's method

**Input**: The event tensor $\mathcal{X}$ and latent factor size $K$;
**Input**: The regularization parameter $\sigma$ and initial gradient step size $\eta$;
**Output**: The factor matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_D$;

**for** $j = 1, \ldots, D$ **do**
  Initialize the rows of $\mathbf{Z}_j^{(0)}$ following $\mathcal{N}(0, \sigma^{-1}I)$;
  $\mathbf{Y}^{(0)} := \mathbf{Z}_j^{(0)}$ ;
**end**
Set $t := 0$ and $a := 0$;
**while** $f(\mathcal{Z}^{(t)})$ *not converged* **do**
  Reconstruct estimated tensor $\hat{\mathcal{X}}$ (unfolded along dim. 1):
  $\hat{\mathbf{X}}_{(1)} := \frac{1}{T} \exp\left(\mathbf{Y}_1^{(t)} \left(\mathbf{Y}_D^{(t)} \odot \ldots \odot \mathbf{Y}_2^{(t)}\right)\right)$, where $T$ is such that sum of elements in $\hat{\mathbf{X}}_{(1)}$ is $M = |\mathcal{X}|$;
  **for** $j = 1, \ldots, D$ **do**
    StepOK := *false* ;
    **while** StepOK = *false* **do**
      $G_j := \left(\mathbf{X}_{(j)} - \hat{\mathbf{X}}_{(j)}\right)\left(\mathbf{Y}_D^{(t)} \odot \ldots \odot \mathbf{Y}_{j+1}^{(t)} \odot \mathbf{Y}_{j-1}^{(t)} \odot \ldots \odot \mathbf{Y}_1^{(t)}\right) - \sigma \mathbf{Y}_j^{(t)}$;
      $\mathbf{Z}_j^{(t+1)} := \mathbf{Y}_j^{(t)} + \eta\, G_j$ ;
      **if** $f(\mathbf{Z}_j^{(t+1)}) \leq f(\mathbf{Y}_j^t) + \mathrm{Tr}\left(G_j^\top \left(\mathbf{Z}_j^{(t+1)} - \mathbf{Y}_j^t\right)\right) + \frac{1}{2\eta}||\mathbf{Z}_j^{(t+1)} - \mathbf{Y}_j^t||_F^2$ **then**
      $\eta := 0.5\,\eta$ ;
      **else** StepOK := *true* ;
    **end**
  **end**
  $a_{t+1} := \frac{1}{2} + \frac{1}{2}\sqrt{1 + 4a_t^2}$ ;
  **for** $j = 1, \ldots, D$ **do**
    $\mathbf{Y}_j^{(t+1)} := \frac{a_{t+1} + a_t - 1}{a_{t+1}} \mathbf{Z}_j^{(t+1)} - \frac{a_t - 1}{a_{t+1}} \mathbf{Z}_j^{(t)}$ ;
  **end**
  $t := t + 1$;
**end**
**return** $\mathcal{Z} = \{\mathbf{Z}_1^{(t)}, \ldots, \mathbf{Z}_D^{(t)}\}$ ;

### 3.2.5 Abnormal Event Detection

We use the latent factors learned during training to evaluate the joint probability of new events in real-time, and mark as abnormal those that have a low probability. Let $\theta$ be a given prob-

ability threshold, an event $(x_{i_1}, \ldots, x_{i_D})$ will be marked as abnormal if $p(x_{i_1}, \ldots, x_{i_D} \,|\, \mathcal{Z}) < \theta$. By pre-computing the denominator of Eq. (4.1), evaluating the joint probability of an incoming event requires only $O(D \cdot K)$ operations. But, if we have a lot of dimensions or the factor vectors are too large, computing this value becomes costly.

Hence, to detect specific types of anomalies, we can instead evaluate the probability of a single dimensional value, conditioned on all other dimensional values. For example, we could evaluate the probability that the event occurs in a certain zone, given the person, day, and time of day corresponding to that event. If the conditional probability of the observed zone is much lower than that of other zones, the event would then be marked as abnormal. Suppose, without loss of generality that the query dimension is $j = 1$. The probability of $x_{i_1}$, conditioned on all other dimensions, is given by

$$p(x_{i_1} \,|\, x_{i_2}, \ldots, x_{i_D}, \mathcal{Z}) \;=\; \frac{\exp\left(\langle z_{i_1}, \ldots, z_{i_D} \rangle\right)}{\sum_{i_1=1}^{N_1} \exp\left(\langle z_{i_1}, \ldots, z_{i_D} \rangle\right)}. \tag{3.13}$$

To evaluate the computational complexity of this query, we note that the inner product can be decomposed as $\langle z_{i_1}, \ldots, z_{i_D} \rangle \;=\; \langle z_{i_1}, \hat{z}_{i_1} \rangle$. Thus, if we pre-compute $\hat{z}_{i_1}$, each inner product computation has a time complexity in $O(K)$, where $K$ is the size of the latent subspace. Since we have to compute $N_1$ of these inner products, the total cost of evaluating the query is in $O(N_1 \cdot K)$.

The next chapter presents some of the experiments conducted to evaluate the performance of this model.

# CHAPTER 4

# EXPERIMENTAL SETUP

This chapter introduces the setup used in the experiments. We designed the main sets of experiments around the following questions, to evaluate the performance of our model in comparison to some popular state-of-the-art approaches. The questions are:

**1:** Can our tensor factorization method outperform top factorization approaches on the low-rank approximation of sparse count data following a log-linear as well as poisson distribution? And is this performance robust to noise?

**2:** Can our factorization model predict the occurrence of future events in real-life data, more accurately than competing factorization approaches?

**3:** Can our proposed method outperform state-of-the-art approaches on the task of detecting abnormal events from their context?

In this chapter the important and unique aspects of these experimental setups are described. The chapter is broadly divided into three categories:

a. **Datasets:** In this section, we'll explain the generation of synthetic datasets. We'll also introduce the real-life datasets used along with the pre-processing strategies applied to obtain the event tensors. Given the unavailability of ground-truth anomalies, we've assumed the datasets to contain only normal records and have altered some of the records in the real-life datasets to represent abnormal behavior. We'll explain the process of this alteration in this section, too.

b. **Performance Measures:** This section will explain the metrics used to evaluate the performance corresponding to each characteristic of the proposed approach.

c. **Models used for comparative analysis:** To prove the advantages of our proposed approach as an LSA model as well as an abnormal behavior detection method, we have

compared the performance with some existing state-of-the-art approaches commonly used for the purposes of LSA or anomaly detection. We'll explain the rationale for choosing these particular methods along with the way they were applied for experimentation, in this section.

## 4.1 Datasets

For some preliminary experiments and low-rank approximation, we used synthetic dataset; whereas, for comparative analysis we used two real-life datasets of reality mining and geo-life taxi trajectories. We altered these real-life datasets to contain abnormal behaviors in order to perform abnormal behavior detection.

### 4.1.1 Synthetically generated dataset

We generated 11 sparse tensors of $100 \times 100 \times 100$, following 2 types of distributions:

a. Log-Linear Distribution: For generating the log-linear distribution, the following formulation was used:

$$p(x_{i_1}, \ldots, x_{i_D} \,|\, \mathcal{Z}) \;=\; \frac{\exp\left(\langle z_{i_1},\, z_{i_2},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle\right)}{\sum\limits_{i'_1=1}^{N_1} \ldots \sum\limits_{i'_D=1}^{N_D} \exp\left(\langle z_{i_1},\, z_{i_2},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle\right)}, \tag{4.1}$$

where $\langle z_{i_1},\, \ldots\, , z_{i_j},\, \ldots\, , z_{i_D}\rangle$ is the inner product between $D$ vectors of $K$ dimensions.

b. Poisson Distribution: The formula for generating Poisson distribution was as follows:

$$p(a \,|\, \mathcal{Z}) = \frac{\exp^{-\mathcal{Z}} \mathcal{Z}^a}{a!} \tag{4.2}$$

where,

- $\mathcal{Z}$ was the Poisson parameter.
- $a = 0 \ldots 100 \times 100 \times 100$ is the independent discrete Poisson observations.

To build each 3 dimensional tensor, we generated three sets of 100 latent factor vectors of size $K = 20$, each one drawn from a zero-mean Gaussian distribution with a covariance of $\Sigma = \sigma^{-1}I$. To achieve a sparsity level of around 70%, values of $\sigma = 1.75$ for log-linear distribution and $\sigma = 3$ for Poisson distribution, were used.

From the equation 4.1, we then used the latent factors to construct the joint probability distribution. Finally, this distribution was converted into an event count tensor by multiplying each probability value by the desired total number of events, $M = 10^6$, and rounded the results to the nearest integer.

**Adding noise**

We represent count data which is best represented as a Poisson distribution. This is why, to conduct experiments for testing the robustness to noise, we added random poisson noise to all the tensors. Specifically, each tensor value $M_{ijk}$ was modified as follows:

$$M'_{ijk} = M_{ijk} + \alpha \cdot \text{Poisson}(\lambda), \tag{4.3}$$

where $\lambda$ is the Poisson distribution parameter and $\alpha$ is a scale factor. To have a noise of reasonable sparseness and magnitude, we used $\lambda = 0.1$ and $\alpha = 10$.

### 4.1.2 Real-Life dataset

We focused on two datasets:

- Reality Mining published by N. Eagle and Lazer (2009)
- Geo-Life Taxi Trajectories pulished by Yuan *et al.* (2010)

The records of both these datasets were first sorted based on the time-stamp information and then formatted, as explained in section 3.1.1, to be represented as:

[person id, hour of the day, zone id] # of observations.

This helped us create an event tensor for each, training, validation and test sets.

The training set contained the first 60% of the records, the validation set contained the following 20% of the records and the test set contained the final 20% of the records. This ensured that the training was done based on the historical data and the test instances were chronologically observed after the instances used for training and validation.

In the experiments not involving tuning of the parameters, the dataset was simply split into train-test with 80% of records in the training set and the final 20% of the records in the test set.

#### 4.1.2.1 Reality Mining

**The Reality Mining (RM)** project was conducted from 2004-2005 at the MIT Media Laboratory. The RM dataset consisted of the tracking information of 106 individuals (students and faculty members from the MIT Media Laboratory and Sloan Business school), several of whom did not participate for a significant amount of time. Each individual was expected to use mobile phones pre-installed with several pieces of software that recorded and sent the researcher data about call logs, Bluetooth devices in proximity of approximately five meters, cell tower IDs, application usage, and phone status. The authors presented the visualization of the dataset as shown in figure 4.1. Observations using these measurements over the course of nine months were collected.

##### 4.1.2.1.1 Pre-processing

From the 106 students, we picked all the students having at least 7 days of data, giving a total of **87** students. For the locations, we used the **1027** unique cell-tower IDs, corresponding to the attribute *areaID.cellID* in the data. The timestamps of the tracking events were encoded using **24** discrete values, one for each hour of the day. Combining these three dimensions,

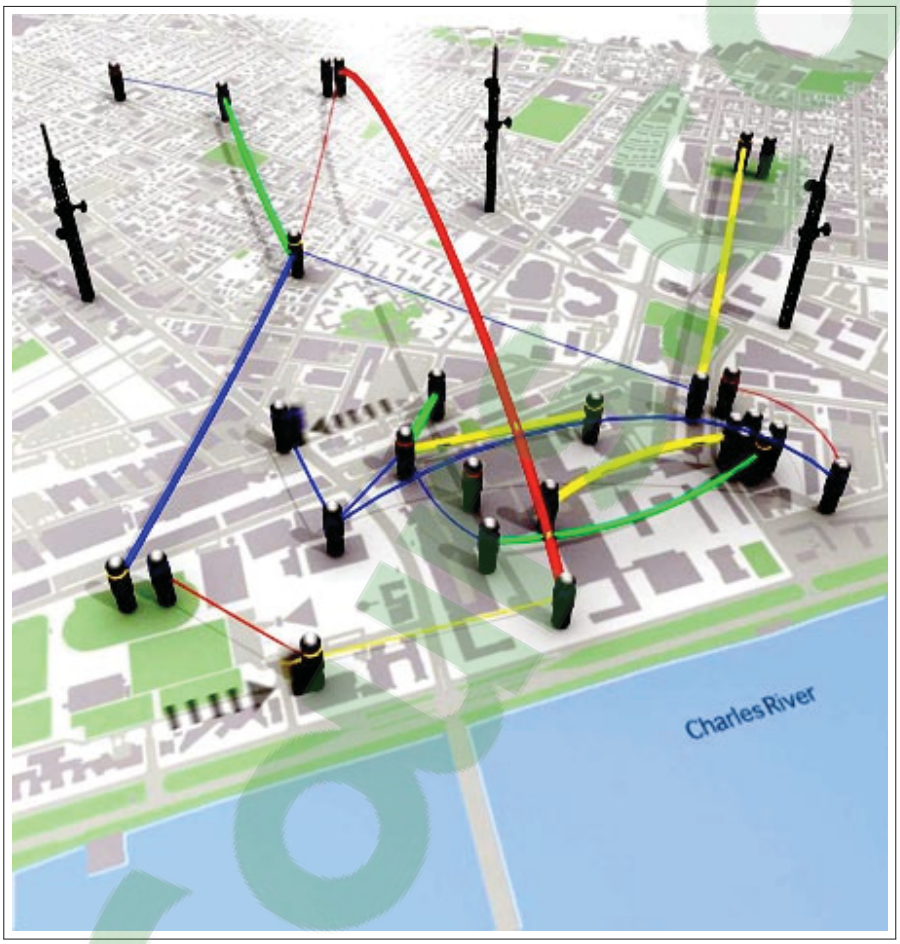



Figure 4.1 Visualization of Reality Mining Dataset
Taken From: N. Eagle and Lazer (2009)

we obtained a $\mathbf{87 \times 1027 \times 24}$ tensor, each cell containing the number of times a person was recorded as being near a given cell tower, at a given time of the day.

#### 4.1.2.2 Geo-Life Taxi Trajectories

**Geo-Life Taxi Trajectories Dataset** dataset contains the GPS trajectories of 10,357 taxis during the period of Feb. 2 to Feb. 8, 2008 within Beijing.

#### 4.1.2.2.1 Pre-processing

From the 10,357 taxis, we picked **259** taxis as the remaining ones had either less than 5000 records of temporal locations or no records at all. In the dataset, the locations were given by the values of longitude and latitude.

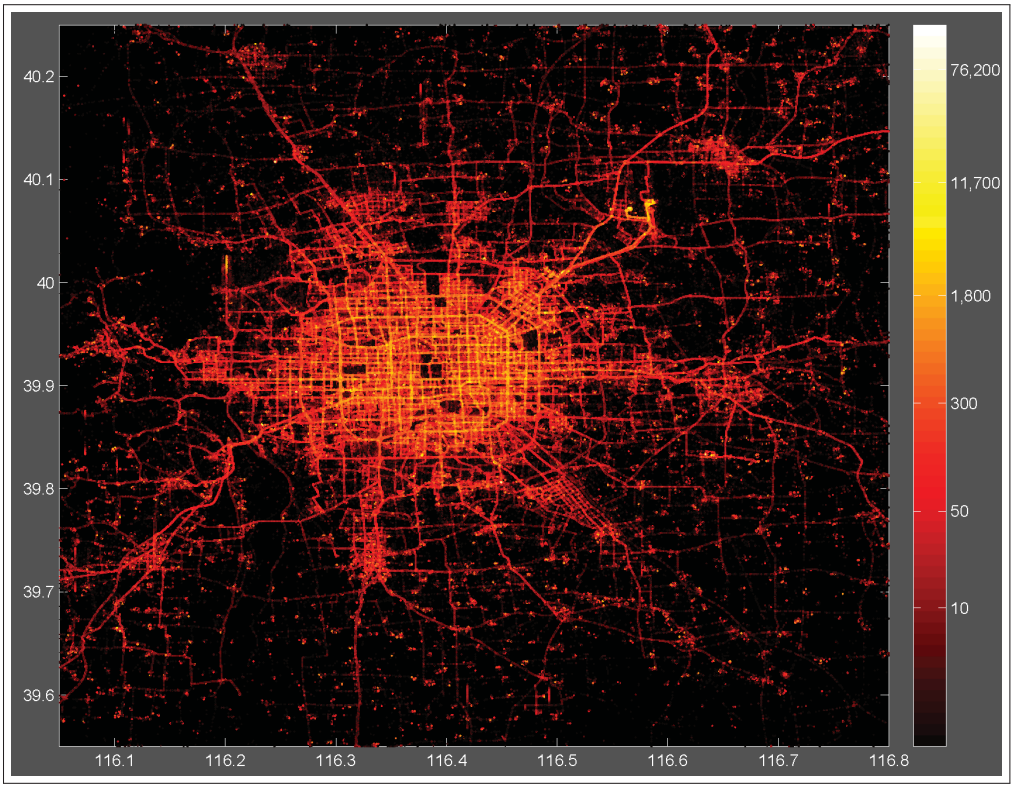



Figure 4.2 Raw Taxi Data
Taken From: Yuan *et al.* (2010)

The majority of the data was concentrated around the city center (Within the 5th ring-road) of the city, as shown by the histogram in figures 4.3 and 4.2. To have a more uniform distribution of events in the map, we converted the zones from their Cartesian coordinates (longitude and latitude) to their log polar equivalents (log radius, $\theta$(angle)) with the Beijing city center's coordinates as origin. We divided the entire space into zones by binning along both, radius and theta. We chose to divide the theta values 30° apart giving us total of 12 bins. We transformed

the radius into log-scale values and chose 10 radius values in such a way that each bin corresponding to a radius value would have at least $1/10^{th}$ of the total amount of records. Thus, with the radius and theta combinations, we get **120** bins which we define as our zones.

Histogram of the distribution of data with respect to the polar coordinates and the labeling of the zones are as shown in figures 4.4 and 4.5.

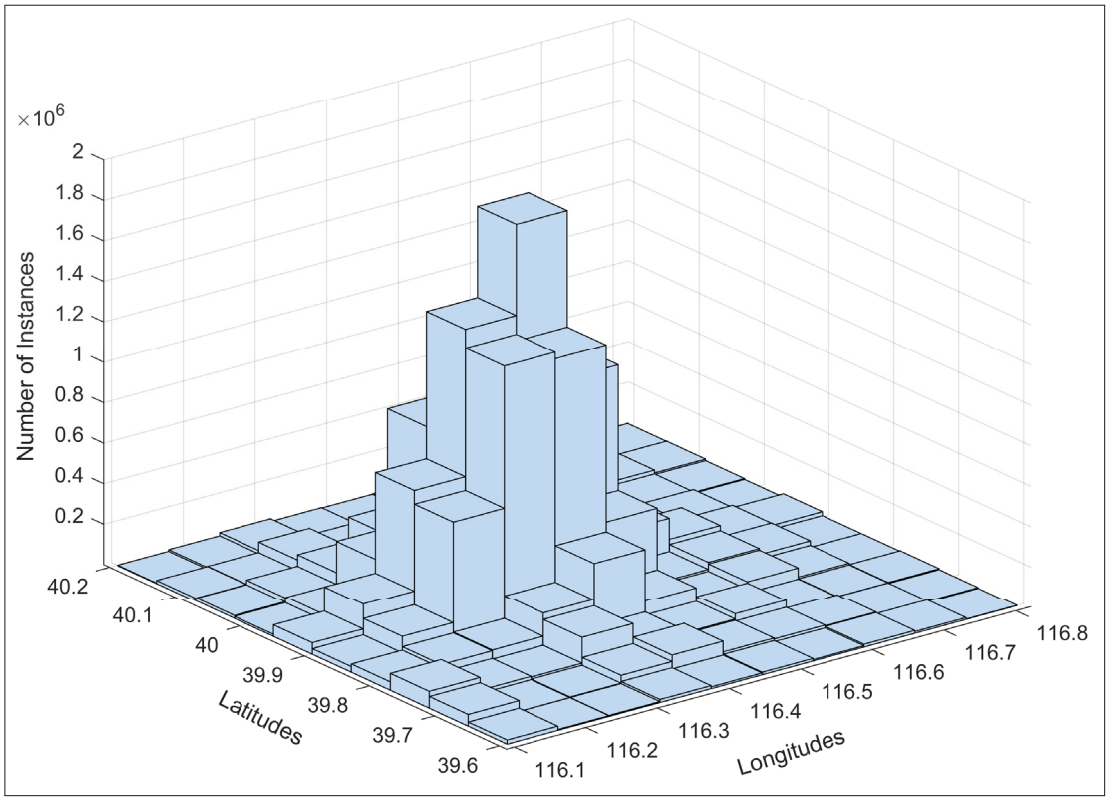



Figure 4.3 Histogram of raw taxi trajectory data

Similar to the RM dataset, we encoded the time-stamps using **24** discrete values, one for each hour of the day. Combining these three dimensions, we obtained a $\mathbf{259} \times \mathbf{120} \times \mathbf{24}$ tensor, each cell containing the number of times a taxi was recorded as being in the zone, at a given time of the day.

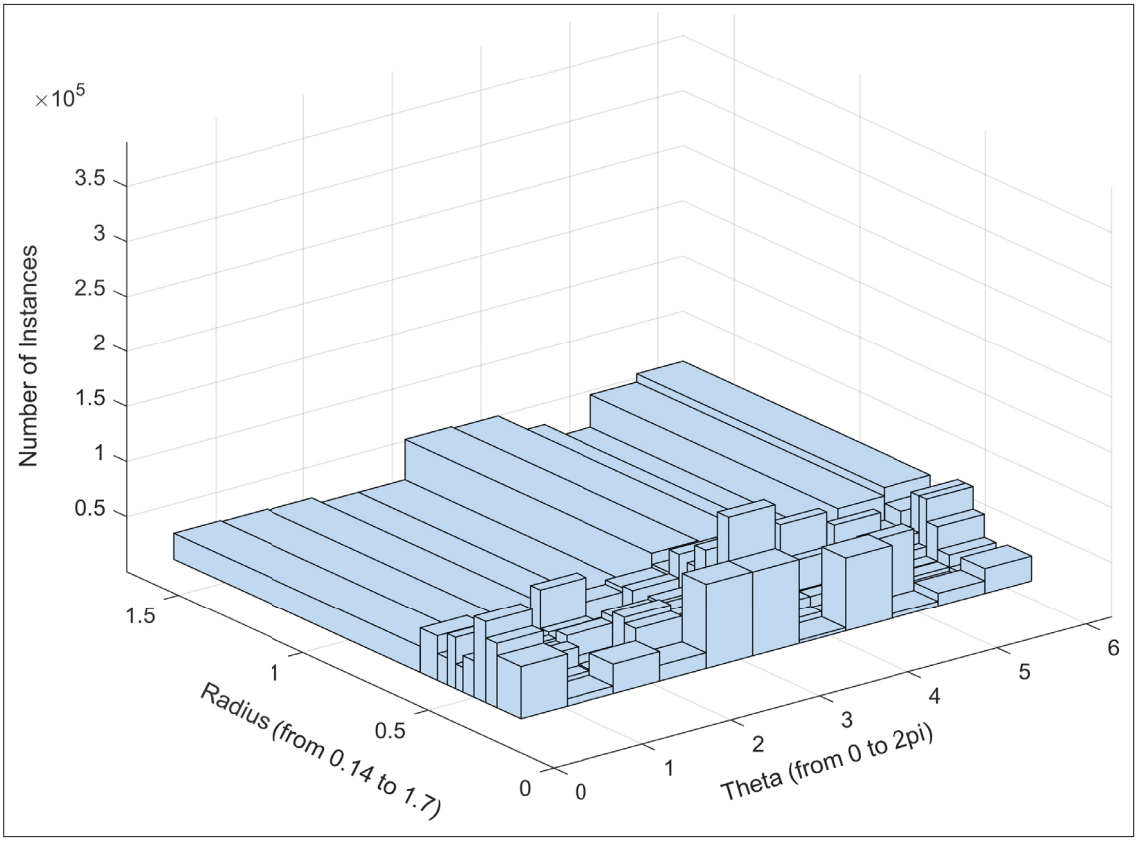



Figure 4.4 Histogram of raw taxi trajectory data distribution based on log-polar coordinates

### 4.1.3 Anomalous Real-Life datasets

For detecting abnormal behaviors, we refer to anomalies as abnormal behaviors. We decided on introducing 3 types of anomalies:

a. **Swap People** Swap events between different people i.e. A person *x* is seen *behaving* like person *y* and vice versa.

b. **Swap Hours** Within the same person, swap hours i.e. A person may behave differently during the different hours of the day.

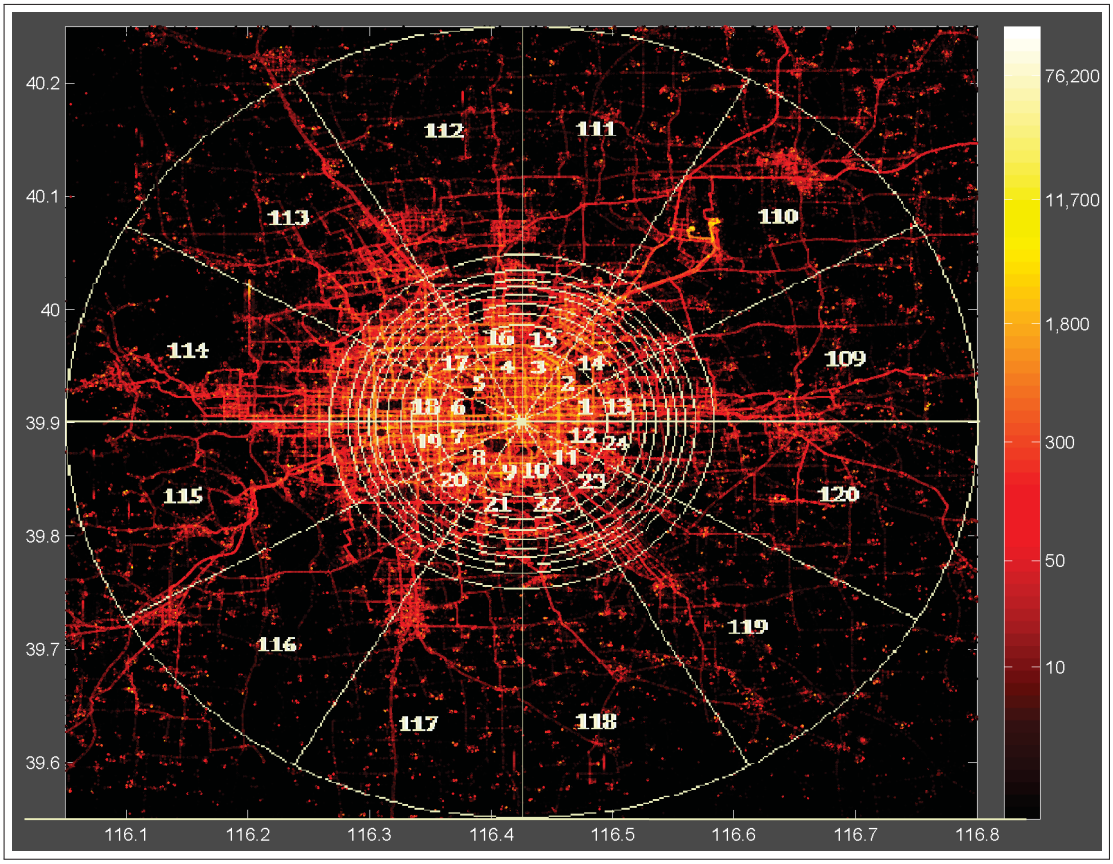



Figure 4.5 Relative estimate of zones in Taxi Trajectory Dataset

c. **Swap Zones** Within the same person, swap zones i.e. A person is not seen as frequently as in a zone where he/she is expected to be (based on the training set) and is rather seen in an unexpected zone.

#### 4.1.3.1 Determining pairs to swap

To choose the persons/zones/hours to swap, we needed to ensure that the distribution of the corresponding events is significantly different. Hence, we calculated the KL divergence since KL divergence is a popular approach for measuring the difference between two distributions. The higher the divergence value, the more the difference.

$$D_{XY}(X\|Y) = \sum_i X(i) \log \frac{X_i}{Y_i} \tag{4.4}$$

As the divergence from $X$ to $Y$ can be different than the divergence from $Y$ to $X$, we decided to use the symmetric KL divergence given by:

$$\text{Divergence} = D_{XY}(X\|Y) + D_{YX}(Y\|X) \tag{4.5}$$

We obtained a $N_d \times N_d$ symmetric matrix containing the divergence values between each pair of values belonging to the same dimension. For example, for the hour dimension, we obtained a $24 \times 24$ matrix containing divergence values between hours of the day with no difference, $value = 0$ between the same value of the hour.

To ensure that pairs with the highest difference get selected for swapping, we picked 200 pairs having the highest divergence values and used those values as weights for random sampling of the corresponding pair to be swapped in the dataset.

#### 4.1.3.2 Swapping People

In both the datasets, we swapped ids of 5 people chosen based on their KL divergence values. Upon processing the dataset, the event tensor's first dimension represented ids of the people, dimension two represented the hours of the day while the third dimension represented the zone-ids. For swapping people, we thus obtained a $N_1 \times N_1$ symmetric matrix containing the divergence values between each pair of values belonging to the same dimension. For example, for the reality mining dataset, we obtained a $87 \times 87$ matrix containing divergence values between people.

#### 4.1.3.3 Swapping Hours and Zones

For swapping hours and zones, we picked 3 people at random. For each person, we generated a KL divergence matrix of $24 \times 24$ for hour dimension or $N_3 \times N_3$ for the zone dimension based on the chosen person's hourly or location based observations. We then swapped 3 values corresponding to their hours or zones based on the weighted KL divergence strategy explained above.

## 4.2 Performance Measures

This subsection will present some of the measures used to evaluate the performance of our model and provide a comparative analysis of the corresponding results. Since the Low-Rank Approximation and Future Event Prediction experiments are about comparing the original and the reconstructed tensors, we used Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the performance. The MAE and RMSE are described as follows. The MAE measures the *average* over the absolute values of differences between the reconstructed/predicted and the original observations. It is computed as follows:

$$MAE = \frac{\sum_{i_1=1}^{N} \cdots \sum_{i_j=1}^{N_j} \cdots \sum_{i_D=1}^{N_D} |\mathcal{X}_{i_1,\ldots,i_D} - \hat{\mathcal{X}}_{i_1,\ldots,i_D}|}{N_1 \times \cdots \times N_j \times \cdots \times N_D} \tag{4.6}$$

The RMSE measures the *difference* between the reconstructed/predicted and original observation, which is squared and then averaged over the total observations. Since the errors are squared before they are averaged, RMSE gives a relatively high weight to high errors. RMSE is measured as follows:

$$RMSE = \sqrt{\frac{\sum_{i_1=1}^{N_1} \cdots \sum_{i_j=1}^{N_j} \cdots \sum_{i_D=1}^{N_D} \left(\mathcal{X}_{i_1,\ldots,i_D} - \hat{\mathcal{X}}_{i_1,\ldots,i_D}\right)^2}{N_1 \times \cdots \times N_j \times \cdots \times N_D}} \tag{4.7}$$

We also measured the Negative Log-Likelihood (NLL) measure as described in section 3.1.2. Similar to MAE and RMSE scores, NLL also follows the negative orientation i.e. lower values correspond to a superior performance. The NLL is calculated as follows:

$$NLL = -\sum_{i_1=1}^{N_1} \dots \sum_{i_D=1}^{N_D} M_{i_1,\dots,i_D} \log p(x_{i_1}, \dots, x_{i_D} \mid \mathcal{Z}) \tag{4.8}$$

For validating the performance of abnormal behavior detection we'll use area under curve (AUC) measures. They are derived from a $2 \times 2$ contingency table, called the *confusion matrix*. The confusion matrix is comprised of True Positive (TP) value which indicates the number of relevant instances retrieved; True Negative (TN) value which indicates the number of irrelevant instances not retrieved (successfully ignored); False Positive (FP) value which is the number of irrelevant instances retrieved; and False Negative (FN) value which is the number of relevant instances ignored or not retrieved [Manning *et al.* (2008)]. The following table represents the corresponding confusion matrix, where the rows will correspond to the retrieval status and the columns will correspond to the relevance of the instances.

Table 4.1 Confusion Matrix

| | |
|---|---|
| TP | FP |
| FN | TN |

Based on this confusion matrix a Receiver Operating Characteristics (ROC) curve is defined which gives us the Area Under the Curve (AUC) measure. An ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers, we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated AUC [Fawcett (2006)]. Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. However, because random guessing produces the diagonal line between (0, 0) and (1, 1), which has an area of 0.5, no realistic classifier should have an AUC less than 0.5. The AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [Fawcett (2006)]. The following figure gives an example of AUC [Fawcett (2006)]. Figure shows the area under two ROC curves, A and B. Classifier B has greater area and therefore better average performance [Fawcett (2006)].

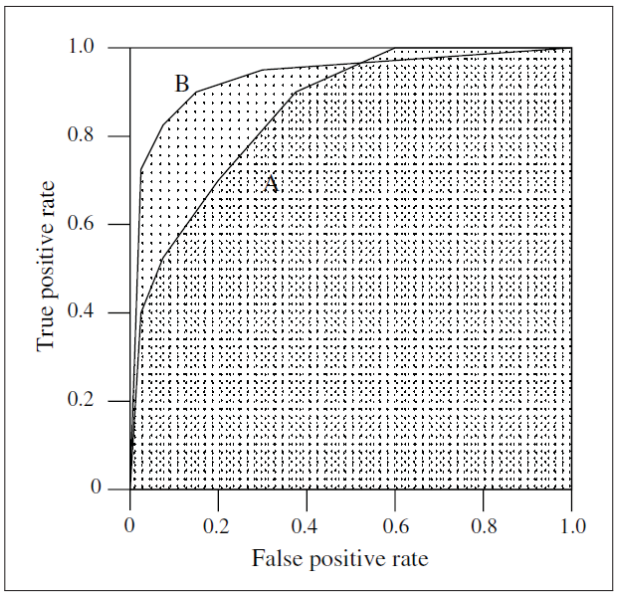



Figure 4.6 Graph of area under two ROC curves, A and B
Taken from: Fawcett (2006)

Thus, such performance measure will help us evaluate the abnormal behavior detection feature of our model.

## 4.3 Models used for comparative analysis

In this section, we have explained the implementation of the state-of-the-art methods chosen to provide a comparative analysis for our approach. As explained in above, NTF and APR were applied for Low-Rank Approximation and Future Event Prediction while OCSVM and KDE were applied for abnormal behavior detection. All the experiments were conducted within the Matlab numerical computing environment.

### 4.3.1 Non-negative Tensor Factorization

With the aid of Tensor toolbox provided by Bader *et al.* (2015), we applied the function available for non-negative tensor factorization.This function computes an estimate of the best rank-K PARAFAC model of a given tensor with nonnegative constraints on the factors. The

version uses the Lee & Seung multiplicative updates from their NMF algorithm presented in Lee and Seung (2001).

### 4.3.2 Alternating Poisson Regression

Alternating Poisson Regression (APR), as proposed by Chi and Kolda (2012), is quite similar to our approach in terms of minimizing KL-Divergence instead of reconstruction error and models count data. However, it differs in its formulation by having added complexity for enforcing non-negativity constraints and is designed to model Poisson distribution, unlike our log-linear approach. For implementation, we once again used the function available in the Tensor toolbox provided by Bader *et al.* (2015). This function computes an estimate of the best rank-K CP model of a given nonnegative tensor using the APR algorithm proposed by Chi and Kolda (2012).

### 4.3.3 One-Class SVM

As seen in section 2.2, OCSVM has been used previously to either get a class of normal activities in the work by Hu *et al.* (2009), or to model and analyze trajectories by clustering the normal ones as explained in Piciarelli *et al.* (2008). Given this popularity of OCSVM for learning the normal class, we applied the method to attempt to learn the normality of the events in the event tensor.

For implementation, we used the well-known *libsvm* toolbox for Matlab which is coded by Chang and Lin (2011). Moreover, we converted the discrete dimensional values, (e.g., *personID*, *zoneID*, etc.) into binary features using an indicator function. This means we converted $N_1 \times N_2 \times N_3$ sized tensor into a $Rows \times Columns$ matrix, where *Rows* represented the # observations in the corresponding train/validation/test set while the $Columns$ represented the dimensional values converted to binary features, hence, of size $N_1 + N_2 + N_3$. So, for RM dataset, we had $87 + 1027 + 24 = 1138$ binary features, whereas for Taxi dataset, we had $259 + 120 + 24 = 403$ binary features. We applied Principal Component Analysis (PCA) on

this matrix of binary features and obtained components representing 95% of variance. These components were normalized to have uniform variance.

Upon training the model, we used the signed distance to the hyperplane to evaluate the normality of test examples, while computing the ROC curves. In simple terms, to obtain the ROC curve, we moved the decision boundary threshold of the SVM by using the decision values parameter of the *svm_predict* function provided in the *libsvm* toolbox. The decision values give us the score instead of a clear class which is desirable since our proposed method also outputs a score in terms of probability instead of a clear class decision. By moving the threshold, that classified the observation in the test set as normal/abnormal, based on the decision values, we could initially classify all observations lower than the lowest decision value as abnormal and then steadily update the threshold until all the observations higher than the highest decision value were considered abnormal. We used this process to obtain the ROC curve evaluating anomaly detection through the OCSVM trained on training set and tested on the anomalous test sets.

### 4.3.4 Kernel Density Estimation (KDE)

KDE is a popular density estimation method applied for anomaly and outlier detection by authors of Latecki *et al.* (2007). Since our approach is viewed as a density estimation approach, we decided to compare its performance with KDE. To implement KDE, we used the same data, as for OCSVM, which was processed into binary feature matrix with PCA applied giving components representing 95% of variance and normalized. Based on the data represented in this manner, we evaluated the probability of a test example $x$ (projected in PCA space) as

$$p(x) \propto \frac{1}{N}\sum_{n=1}^{N}\exp\left\{-\frac{1}{h}||x-x_n||^2\right\}, \tag{4.9}$$

where $x_n$ are the training examples (in PCA space) and $h$ is the kernel bandwidth parameter, tuned on the validation data.

We obtain the ROC curve using the same strategy as of OCSVM by moving the threshold, classifying observations lower than a certain value of probability, as abnormal.

# CHAPTER 5

# EXPERIMENTS AND RESULTS

In this chapter, we present the experimental results conducted as per the setups mentioned in the previous chapter. We initially present some characteristics of our LLTF model and follow that with the main experiments which explore the three questions mentioned in the previous chapter.

## 5.1 Characteristics of the model

### 5.1.1 Visualization of latent factors

To illustrate the principle of factorization, we visualized the latent factors. We factorized the event tensor of the Reality Mining dataset with $K = 3$. The trained factors of the reality mining corresponding to the *hour* dimension were plotted as shown in figure 5.1.

For simplicity of presentation, figure 5.1 represents the latent factor vector values corresponding to two out of three columns. All the 24 vectors from the hour dimension have learned the high-level behavior from the environment within the dataset. It can be observed that the latent factors corresponding to the afternoon hours from 12 to 17 are quite similar whereas the early morning hours of 4 to 6 are similar and very different from the afternoon hours. This implies that the daytime activities vary significantly from the night-early morning activities in this environment.

### 5.1.2 Convergence Analysis

Figure 5.2 illustrates the convergence rate of our Nesterov-based method on a sample tensor, compared to simple gradient ascent (GA). We see that the convergence in terms of NLL (blue curves) is attained within 60 iterations, with an average time of 0.3 seconds per iteration, whereas GA has not converged after 100 iterations.

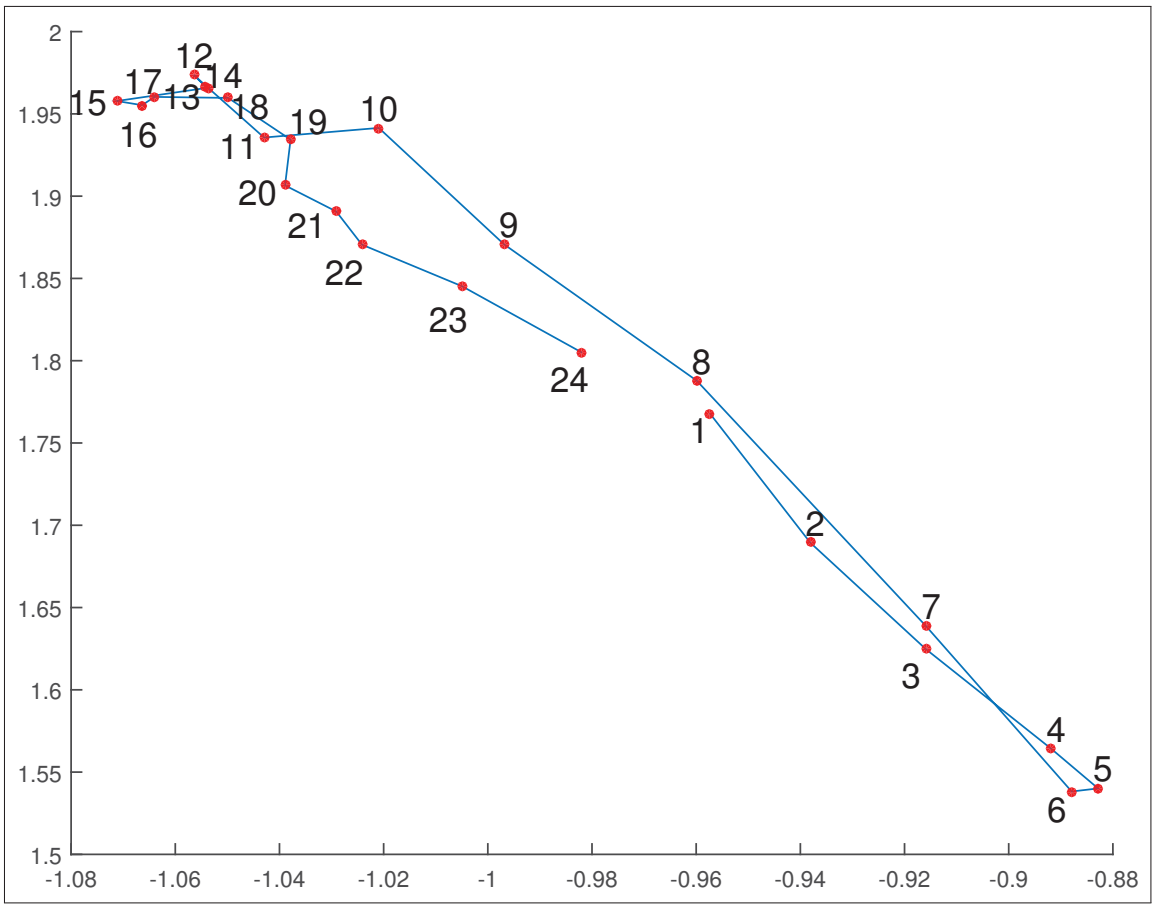



Figure 5.1 Plot of the latent factors corresponding to the hour dimension, trained on Reality Mining dataset

### 5.1.3 Parameter impact

Since the latent factors learn the tensor and are expected to be **concise**, our hypothesis is, *"Our model can better approximate a given empirical distribution of events using the same number of parameters (i.e. latent factors)"*. To prove this hypothesis, we factorized 10 $100 \times 100 \times 100$ log-linear tensors to factor vector sizes of 5, 10, 15 and 20. We reconstructed the tensor from these four sizes of latent factors and compared it with the original tensor. The comparative performance measures used were *Mean Absolute Error (MAE)*, *Root Mean Squared Error (RMSE)* and the *Negative Log-likelihood (NLL)*. Figure 5.3 show the results, averaged over all 10 tensors with variance.

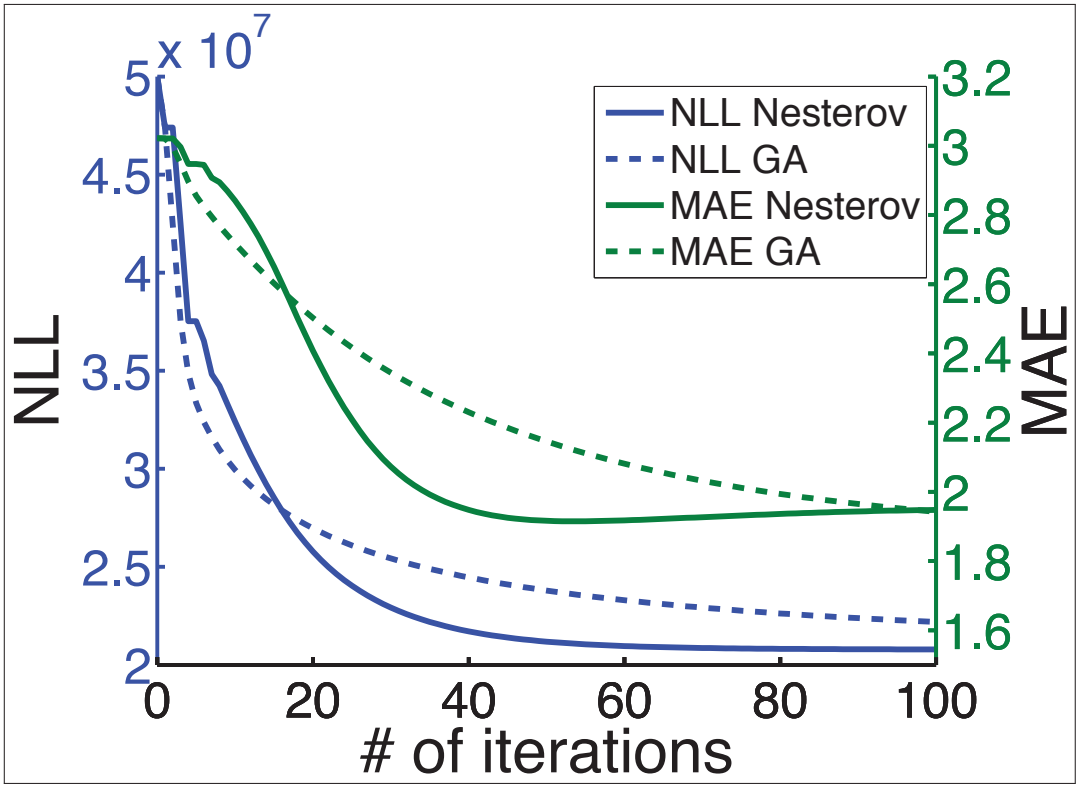



Figure 5.2 Convergence of our Nesterov-based method on a sample tensor, compared to simple gradient ascent (GA).

As expected, the higher the size of vector, the better the results with lower reconstruction error. This is because more information is represented if the size of factor vectors increases. This leads to a requirement for tuning this parameter as we want to prevent overfitting/underfitting the data. If a large $K$ is used we will learn the training data very well but will have a poor generalization to the testing data (overfit). If $K$ is too small, then the model is not powerful enough to learn the distribution (underfit).

## 5.2 Parameter tuning

For the proposed approach, we required tuning of the latent factor vector size $K$ and the regularization parameter $\sigma$. For OCSVM, two parameters required tuning: $\nu$, which controls the fraction of training examples allowed outside the learned region, and the RBF kernel parameter $\gamma$. Whereas, for KDE, the kernel bandwidth parameter $h$ required tuning.

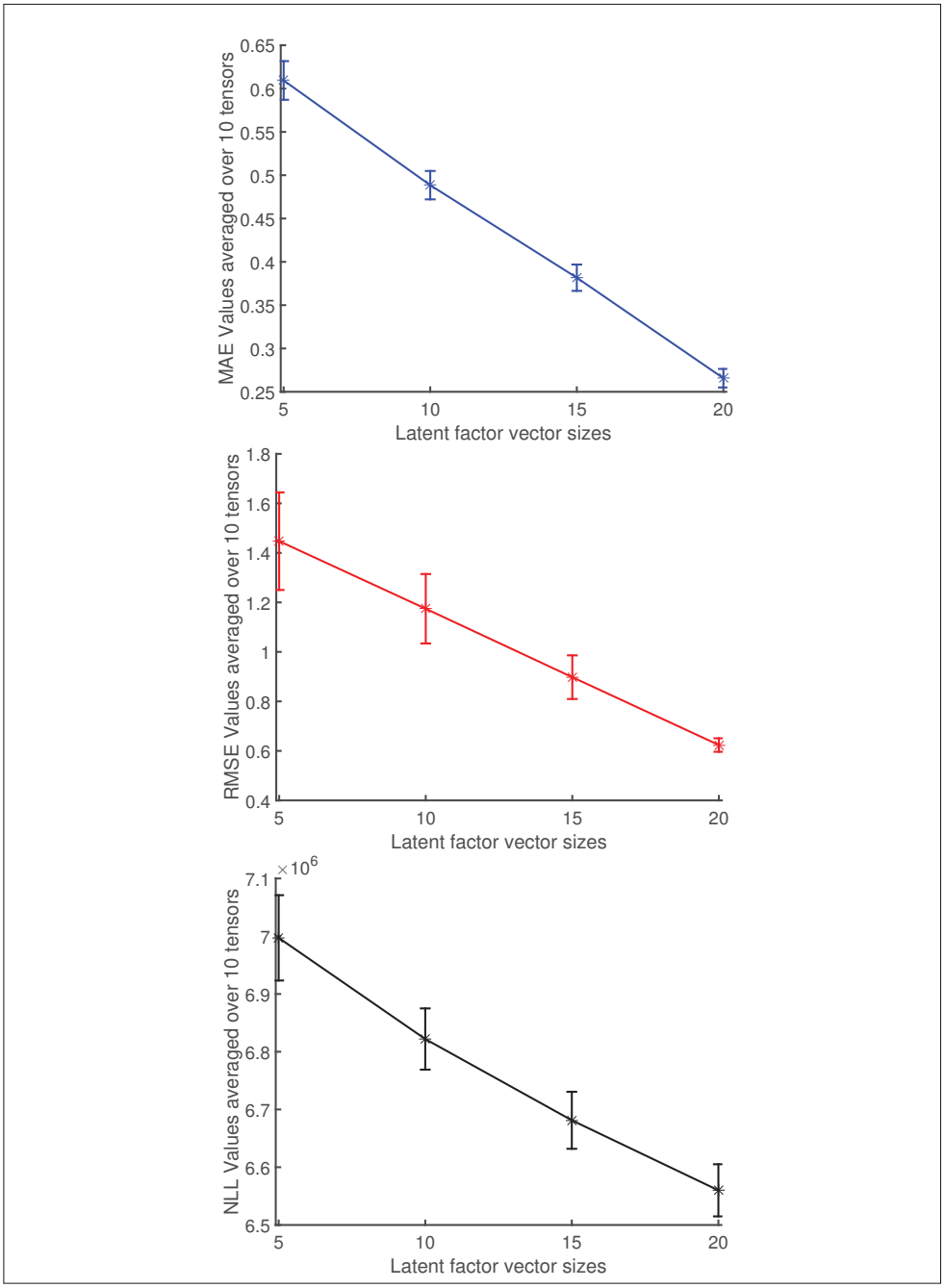



Figure 5.3 Impact of Latent factor vector size on minimization of error

For this purpose, the validation test was used. Given our method's application in the experiment of Low Rank Approximation, we generated the data for the validation set in the same way as explained in section 4.1.1. We initially provided a range of values for $K$ and $\sigma$ and used a grid

search strategy for identifying the $K$ and $\sigma$ values-combination that gave the best NLL on the validation set. For the experiment predicting future events real-life datasets are used. These data-sets were divided into train, validation and test sets, chronologically. This means, the events in the test set were the most recently observed ones while those in the training set were the oldest observed. The validation set events were observed more recently than the training set ones but historic in comparison to the events in the test set.Due to this chronological nature of the events in the dataset, we could have only 1 validation set and hence, we used one-fold cross validation. However, since our model initializes with random values for latent factors, we validated over the validation set by training our model multiple times, each time, with new initializations of latent factors. We followed the same strategy of grid search for finding the best $K$ and $\sigma$ values-combination based on the NLL value. The chosen values were the ones which, upon training the model multiple times using the same parameters but different initializations, gave the best NLL value on the validation set.

For the abnormal behavior detection experiment, we used the 10 anomalous validation sets which were generated from the real-life datasets as explained in section 4.1.3. It has to be noted that there were three kinds of anomalies introduced. Hence, we had validation sets corresponding to each containing only one kind of anomaly. Through this experiment, we also noted the robustness of our method. We tuned in the same way as for the experiment predicting future events but on the anomalous validation sets containing the *swap people* anomaly. However, for OCSVM and KDE, we tuned the models on all the validation sets containing all the anomalies before testing on the corresponding anomalous test set. In other words, we tuned our model on the *swap people* anomaly, trained the model using these parameters and applied it on the test set containing *swap people*, *swap hours* and *swap zones* anomalies. But, for OCSVM and KDE, we tuned the models to have the best parameters corresponding to detection of each kind of anomaly. So we had an OCSVM model especially tuned and trained for each *swap people*, *swap hours* and *swap zones* kind of anomaly. Similarly, for KDE, we had the $h$ parameter tuned for each kind of anomaly before applying it correspondingly on the test

set for evaluation. On the contrary, we trained our model using the same parameters deemed best on one kind of anomaly before applying it on all kinds of anomalies.

### 5.3 Low-Rank Approximation

The aim of this experiment was to measure the ability of our method to fit the event tensor using a small number of parameters. We also wanted to measure the robustness of this method to noise in the data.

We performed this experiment by reconstructing the tensor from the latent factors as follows:

$$T = \sum_{i_1=1}^{N_1} \dots \sum_{i_j=1}^{N_j} \dots \sum_{i_D=1}^{N_D} \exp\left(\mathbf{Z_1} * (\mathbf{Z_D} \odot \cdots \odot \mathbf{Z_j} \odot \cdots \odot \mathbf{Z_2})\right) \tag{5.1}$$

with $M$ = total number of events in the original event tensor, $\mathcal{X}$, we get reconstructed tensor, $\hat{\mathcal{X}}$ as:

$$\hat{\mathcal{X}} = \frac{M}{T} * \exp\left(\mathbf{Z_1} * (\mathbf{Z_D} \odot \cdots \odot \mathbf{Z_j} \odot \cdots \odot \mathbf{Z_2})\right) \tag{5.2}$$

We performed this experiment by reconstructing the tensor from the latent factors and comparing the original tensor with the reconstructed one. Since we expect a larger size of the vector to represent more information and hence result in a lower reconstruction error, we conducted this experiment four times with factor vector sizes varying from $K = 5$, 10, 15 and 20. The synthetic data consisted of 11 tensors in each category corresponding to following four types of distributions:

a. Log-linear distribution without noise (Pure Log-Linear)

b. Poisson distribution without noise (Pure Poisson)

c. Log-linear distribution with noise (Noisy Log-Linear)

d. Poisson distribution with noise (Noisy Poisson)

For all these four types of tensors, we compared tensors reconstructed from trained factors with vector sizes varying from $K = 5$,10,15 and 20. From the 11 tensors of each type, we chose the first tensor to tune the $\sigma$ parameter, which is a regularization parameter, corresponding to the distribution and the $K$ value. Upon noting the Negative Log-Likelihood (NLL) performance corresponding to the $\sigma$ giving us the best value, we chose that $\sigma$ value with the corresponding $K$ and tensor distribution on the remaining 10 tensors of that category. We have reported the averaged approximation error through MAE, RMSE and NLL.

In order to have the comparative analysis, we applied two factorization approaches: Alternating Poisson Regression (APR) and Non-Negative Tensor Factorization (NTF) on the same set of tensors and noted the approximation error through the same measures.

The Mean Absolute Error (MAE) over the log-linear distribution and poisson comparing the distributions with (noisy) and without noise (pure) are shown in figure 5.4 and 5.5.
The mean absolute error was calculated as per equation 4.6. It can be observed that on log-linear data, our method outperforms both, NTF and APR significantly. However, on pure Poisson data, APR manages to slightly surpass by giving an error of $1.073 \pm 0.018$ compared to LLTF, which gives an error of $1.074 \pm 0.018$. This is justifiable since APR is specifically designed to model Poisson distribution, itself. But, if we add random Poisson noise, we can see that LLTF gives an error of $1.314 \pm 0.017$ while APR gives an error of $1.316 \pm 0.017$, thereby, demonstrating the robustness of our method, to noise. The Root Mean Squared Error (RMSE) over the log-linear distribution and poisson comparing the distributions with (noisy) and without noise (pure) are shown in figure 5.6 and 5.7

The root-mean squared error was calculated as per equation 4.7. It should be observed that LLTF outperformed both NTF and APR on log-linear data, significantly again. However, in terms of RMSE measurement NTF outperformed APR over both kinds of log-linear data. So, APR is unable to perform consistently in terms of MAE and RMSE over reconstruction across different distributions whilst LLTF continues to outperform both over log-linear distributions as well as Poisson data, albeit it was marginally outperformed by APR on pure Poisson dis-

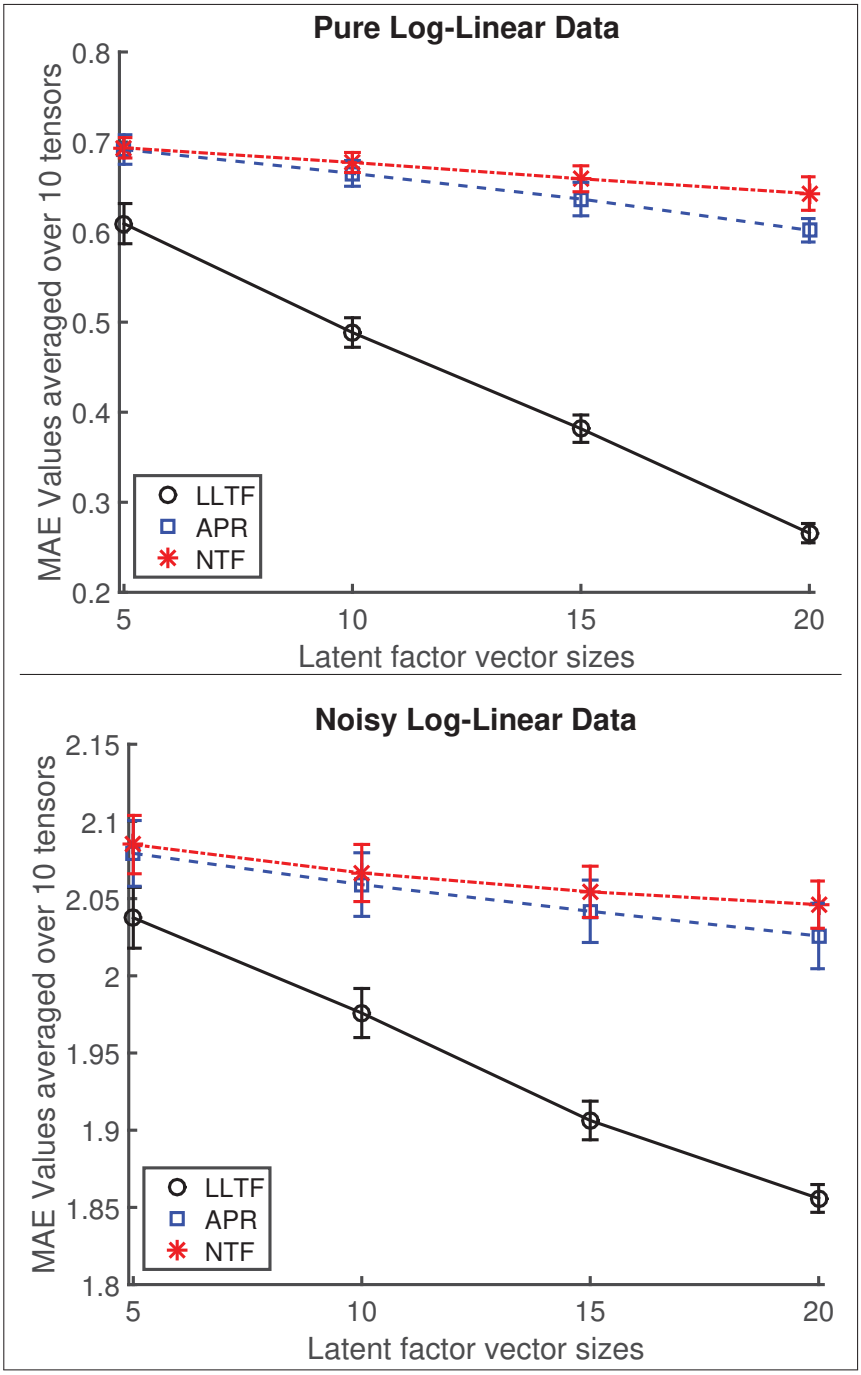



Figure 5.4 Averaged Mean Absolute Error on Log-linear data - with and without noise

tribution in terms of MAE. However, for RMSE, it can be observed that both LLTF and APR gave same errors of $1.413 \pm 0.037$ on the pure Poisson data. But, again, for noisy Poisson data,

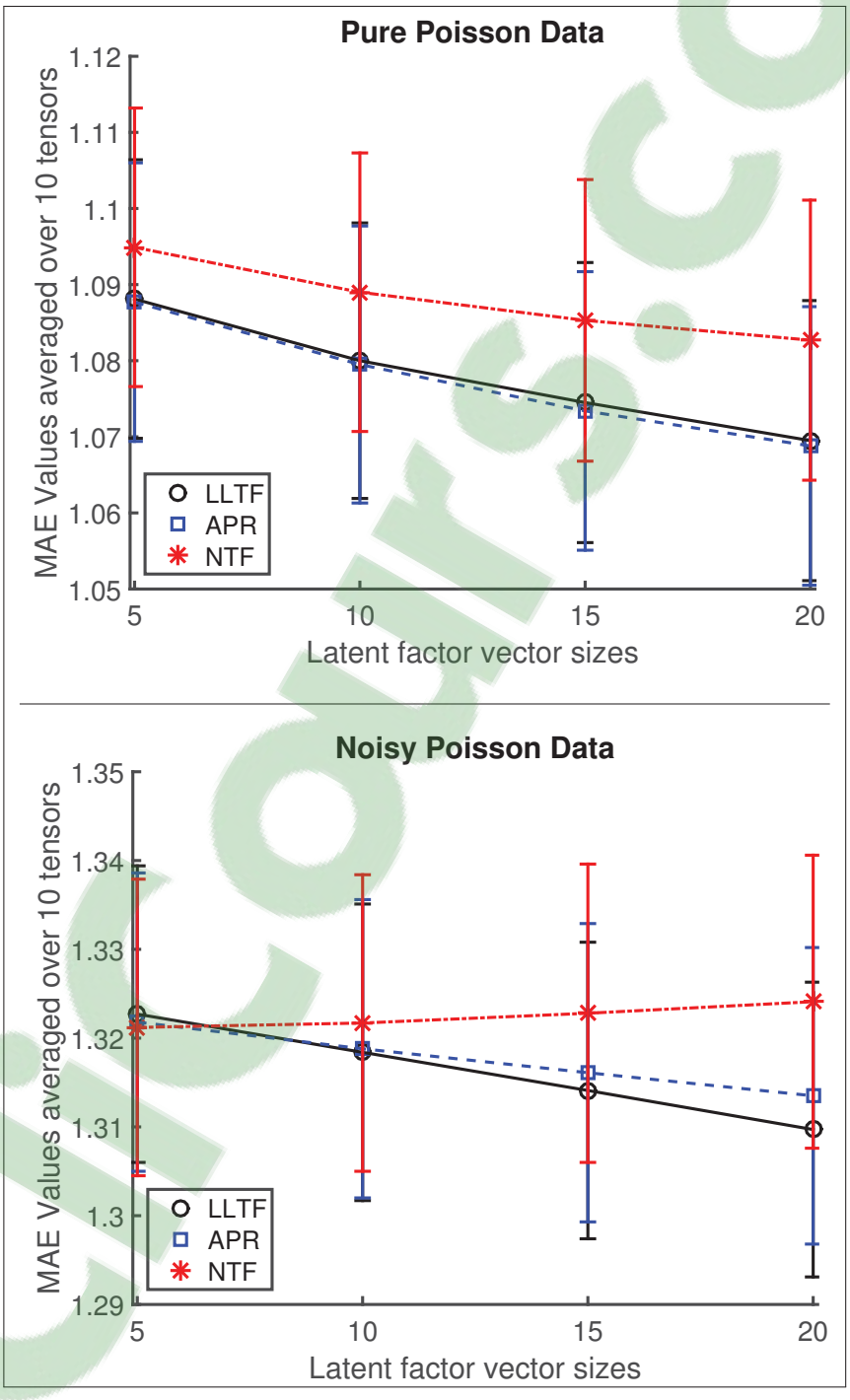



Figure 5.5 Averaged Mean Absolute Error on Poisson distributions - with and without noise

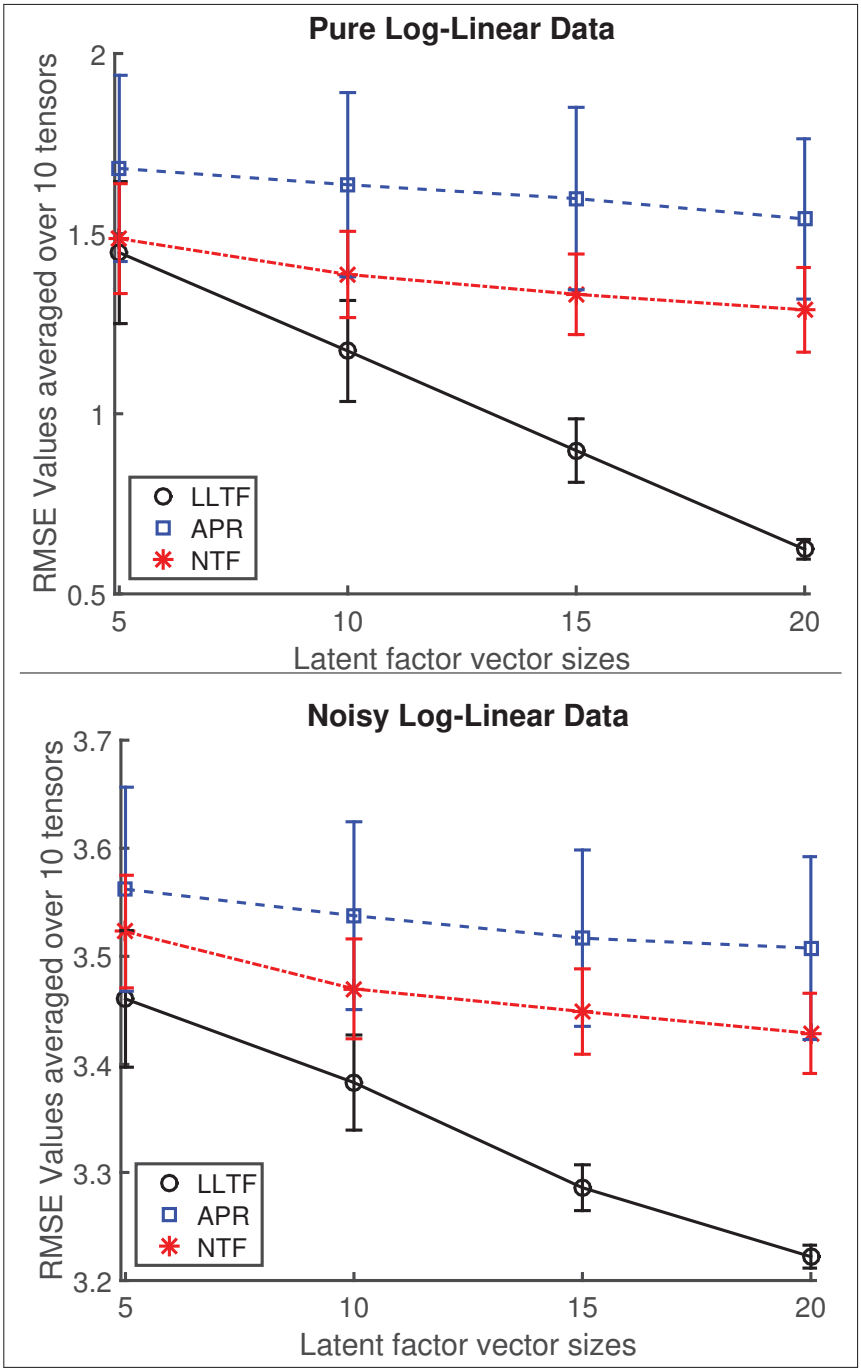



Figure 5.6 Averaged Root Mean Squared Error on Log-linear distributions - with and without noise

LLTF gives a better performance in comparison by giving an error of $2.254 \pm 0.033$ whilst APR gives an RMSE of $2.258 \pm 0.033$.

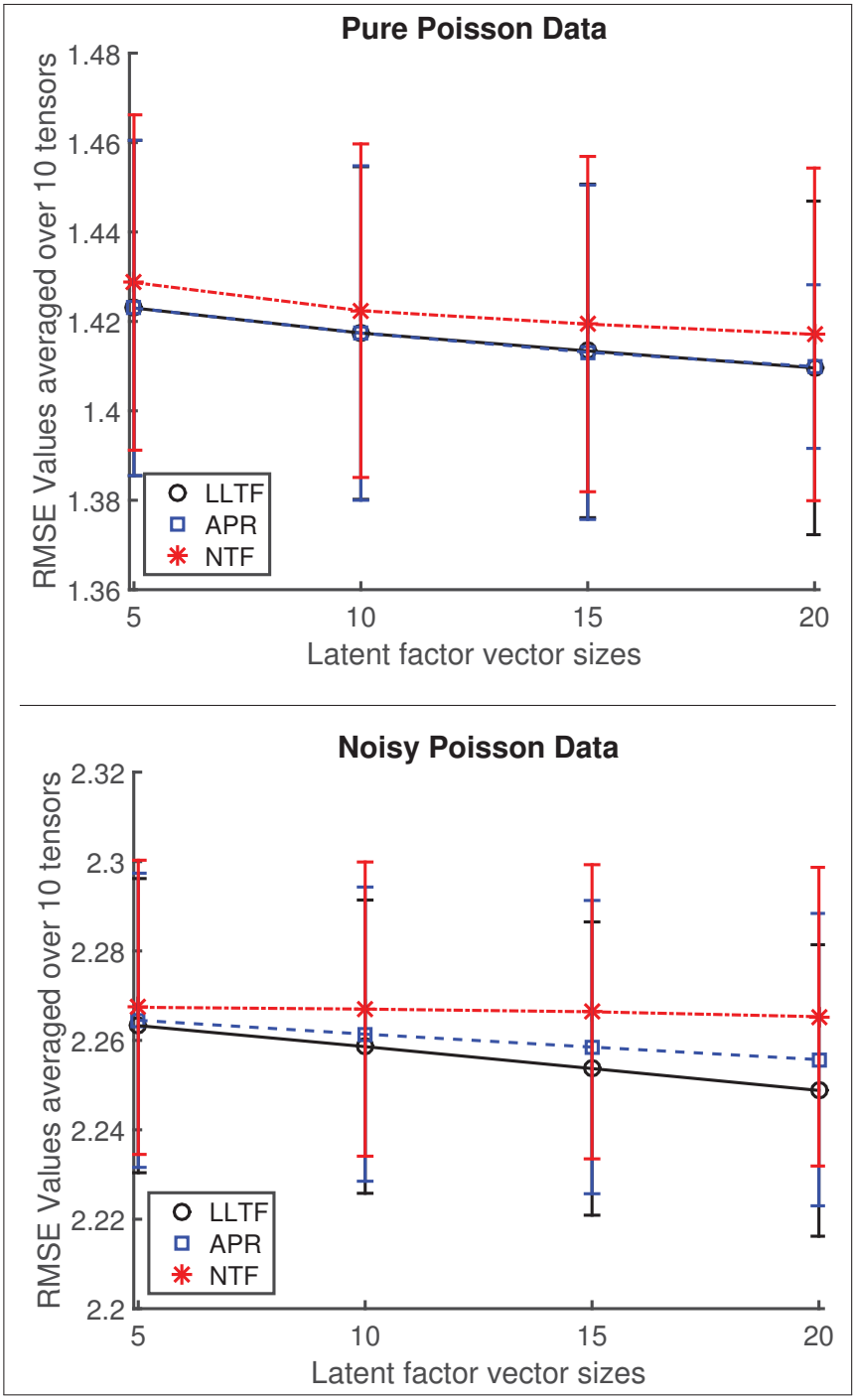



Figure 5.7 Averaged Root Mean Squared Error on Poisson distributions - with and without noise

The Negative Log Likelihood (NLL) over the log-linear distribution and poisson comparing the distributions with (noisy) and without noise (pure) are shown in figure 5.8 and 5.9

The NLL was calculated as per equation 4.8. As with MAE and RMSE, we obtain a lower NLL than APR and NTF on the log-linear data. However, on the pure Poisson data, we can observe that the results of APR and LLTF are exactly the same for all the values of $K$ whilst NTF lags by approximately 0.1% relatively, in comparison. But, again, we see LLTF surpassing performance of that of APR when noise is added to Poisson data, thus proving its robustness to noise, in comparison.

Some of the general observations from these figures, are as follows:

- As expected, that the error values are lower for pure tensors as compared to the ones with added noise.
- The reconstruction error decreases with higher values of latent factor vector sizes, which is also expected.
- Our LLTF method outperforms APR and NTF for log-linear tensors, especially for pure data and larger values of factor vector sizes. Hence, for pure tensors, using $K = 20$, LLTF obtains an average MAE which is 2.26 times smaller than APR and 2.41 smaller than NTF.
- For Poisson data, LLTF performs as well as APR, even though this data is tailored to APR's Poisson model and not LLTF's log-linear one. In summary, because it uses a non-linear model, LLTF is more robust to the data's underlying distribution.
- Since it is not designed for sparse count data, NTF obtains a lower performance than LLTF and APR for both, log-linear and Poisson data.

## 5.4 Future Event Prediction

In this experiment, we evaluate the ability of our method to predict the occurence of future-events in *real-life* data for which, we used the RM and Taxi datasets with the latent factor

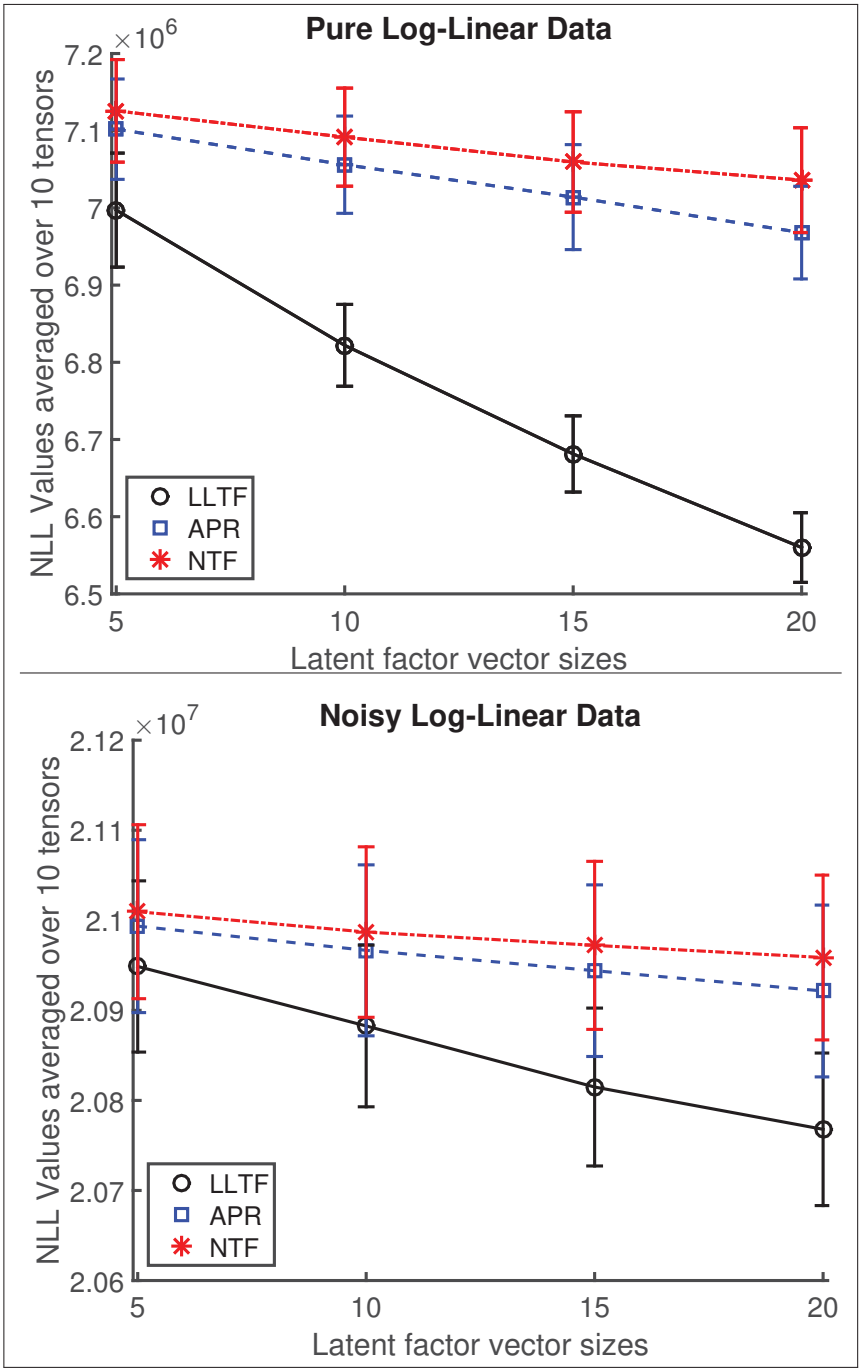



Figure 5.8 Averaged Negative Log Likelihood on Log-linear distributions - with and without noise

vector size $K = 10$. We trained our model on the training sets of RM and Taxi data using the parameters tuned on the corresponding validation sets.

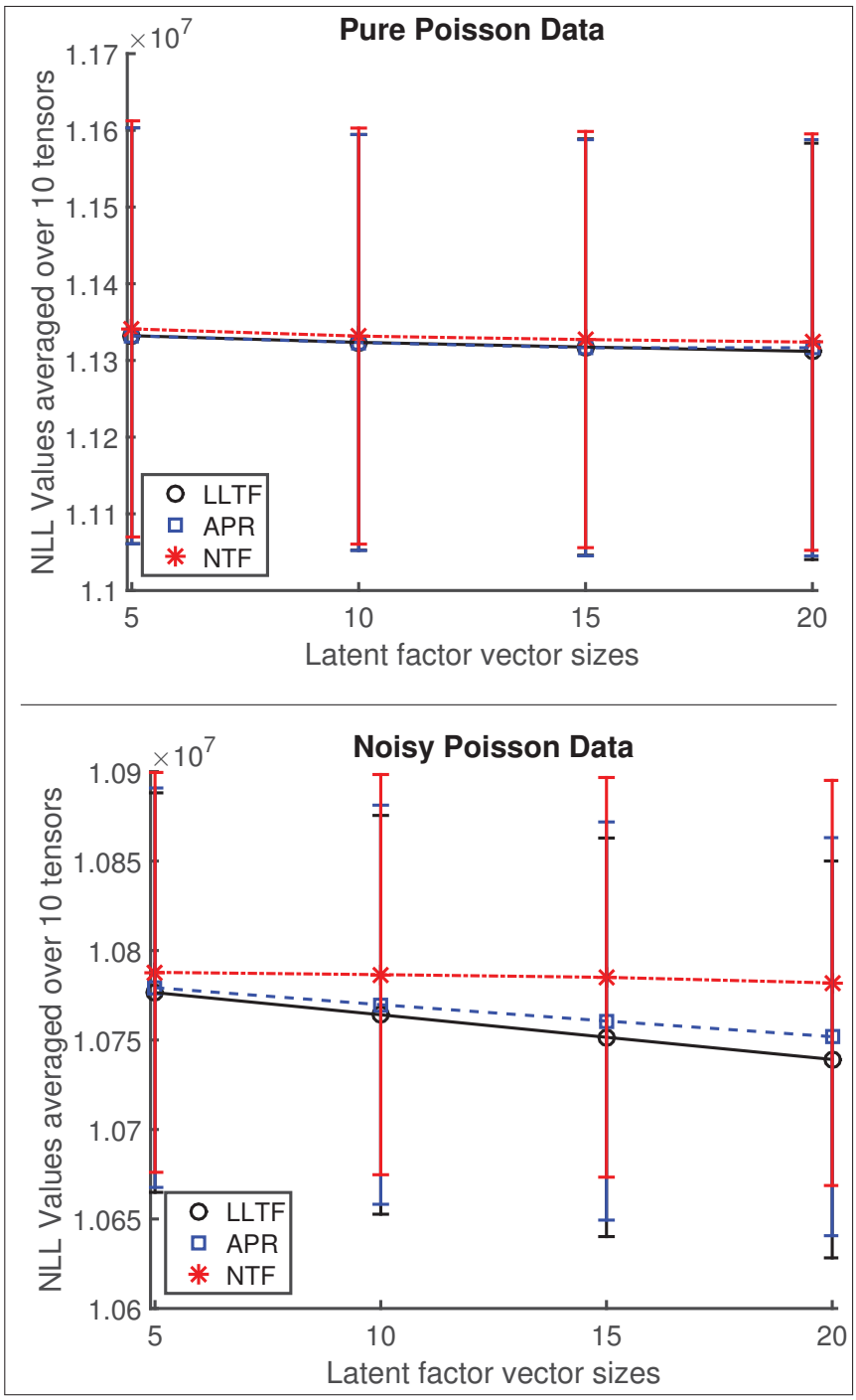



Figure 5.9 Averaged Negative Log Likelihood on Poisson distributions - with and without noise

For testing, we compared the performance with NTF and APR factorization methods. Recall that the Poisson distribution used in APR is specifically tailored to model count data. A latent factor vector size of $K = 10$ was used for all three methods. The MAE, RMSE and NLL were used as the performance measures. The results on the test set are as follows:

Table 5.1 Reality Mining Dataset with $\sigma = 6$

| Measure | LLTF | APR | NTF |
|---|---|---|---|
| MAE | 0.384 | 0.892 | 0.894 |
| RMSE | 8.906 | 19.134 | 19.743 |
| NLL | 7272727.351 | $\infty$ | 7814523.661 |

Table 5.2 Taxi Trajectory Dataset with $\sigma = 4$

| Measure | LLTF | APR | NTF |
|---|---|---|---|
| MAE | 1.691 | 3.549 | 3.084 |
| RMSE | 15.319 | 19.657 | 24.220 |
| NLL | 9158202.736 | $\infty$ | 9719013.830 |

The prediction accuracy of the three tested methods on the RM and Taxi datasets is detailed in table 5.1 and 5.2. We see that LLTF outperforms APR and NTF on both datasets and all three performance metrics. Thus, LLTF obtains MAE 2.32 times lower than APR in the RM dataset and 2.10 times lower in the Taxi dataset.

We also note that APR obtained infinite NLL values. This is because it gave a zero probability to the events in the test set that were not observed in the training set. By regularizing the factors, our model can better predict such unobserved events.

### 5.5 Abnormal behavior Detection

In this last experiment, we assess the usefulness of our method to detect abnormal events from their context. For these set of experiments, we used the Reality mining and the GeoLife Taxi

Trajectory dataset, similar to the previous experiment. Recall, we used the anomalous real-life datasets containing three types of synthetic anomalies:

a. Swap People: Swap events between different people.

b. Swap Hours: Swap time-periods within the same person.

c. Swap Zones: Swap zones within the same person.

Upon tuning, the best value for Reality Mining validation dataset was $K = 25, \sigma = 8$ which gave $AUC = 0.9962$ whereas for Taxi Trajectories validation dataset, $K = 25, \sigma = 2$ that gave $AUC = 0.9638$.

With these values for our parameters for the respective datasets, we trained our model and for every possible event, we tested four variations of our proposed approach. In the first one, called LLTF-Joint, the ROC curves are generated by evaluating the joint probability of test examples, as defined in Eq. (4.1), and then computing the precision/recall for increasing probability thresholds. The other three methods, denoted by LLTF-$D$, where $D = \{\text{Person}, \text{Time}, \text{Zone}\}$, instead use the conditional probability of a single dimension $D$ given the other two dimensions, as described in Eq. (3.13).

We compared the performance based on AUC with OCSVM and KDE.

The mean ROC curves (and corresponding AUC values), computed over the 10 test sets of each anomaly type, are shown in Figures 5.10, 5.11 and 5.12.

Except for the *Swap Time* anomalies, our LLTF-Joint method obtained a higher mean AUC than OCSVM and KDE. In particular, the AUC of LLTF-Joint is 9% to 35% higher than OCSVM, and 10% to 37% higher than KDE, for *Swap People* anomalies. Furthermore, the conditional probability model can improve the detection of specific types of anomalies. For instance, LLTF-Person obtained mean AUC values of 0.97 and 0.93 on the Swap People anomalies, compared to 0.95 and 0.92 for LLTF-Joint. Similarly, LLTF-Time obtained a mean AUC of

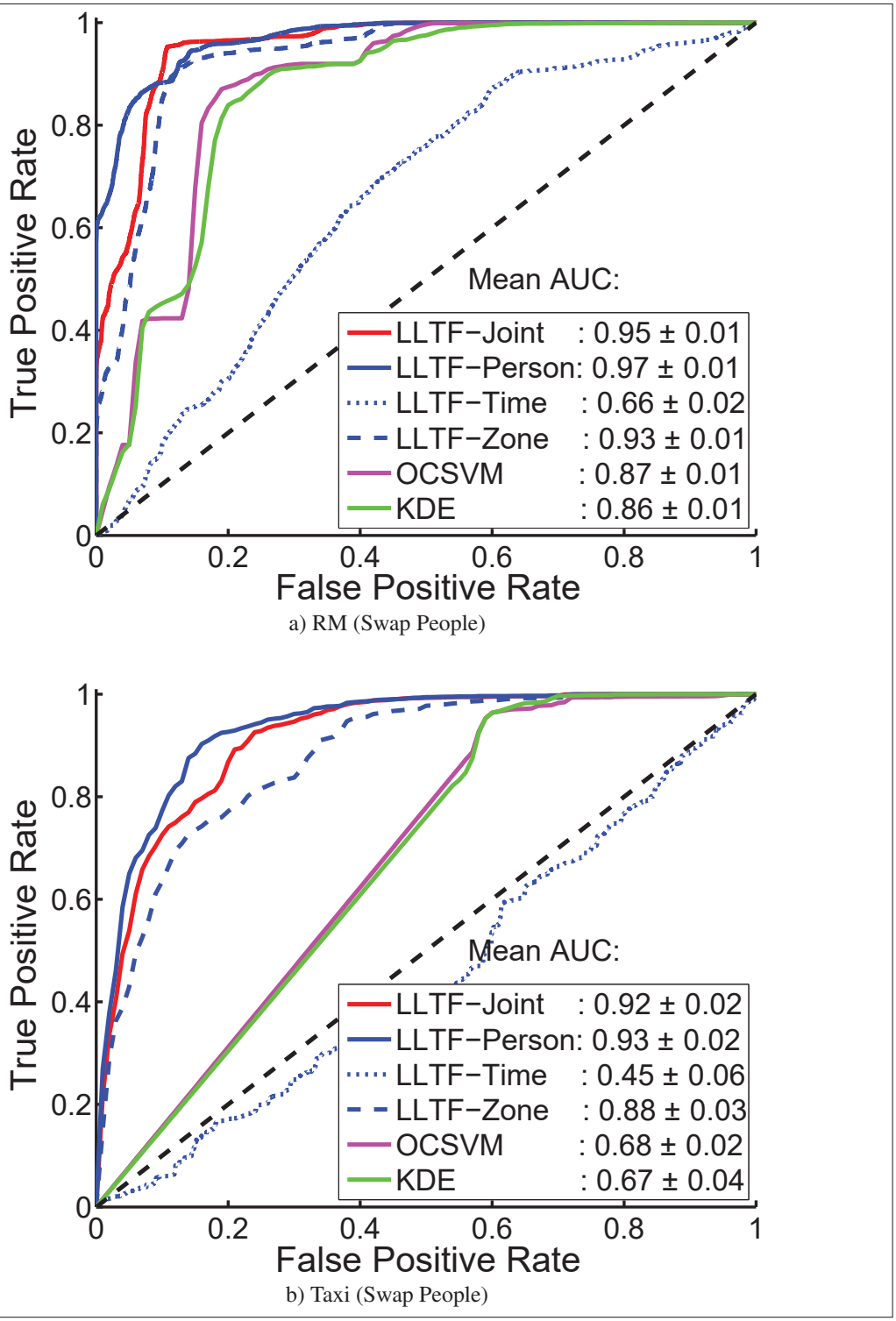



Figure 5.10 Average ROC curves and AUC obtained by LLTF, OCVSM and KDE approaches, on 10 sets of synthetic anomalies generated from the Reality Mining (RM) and Taxi datasets.
Anomaly: Swapping the events of two persons.

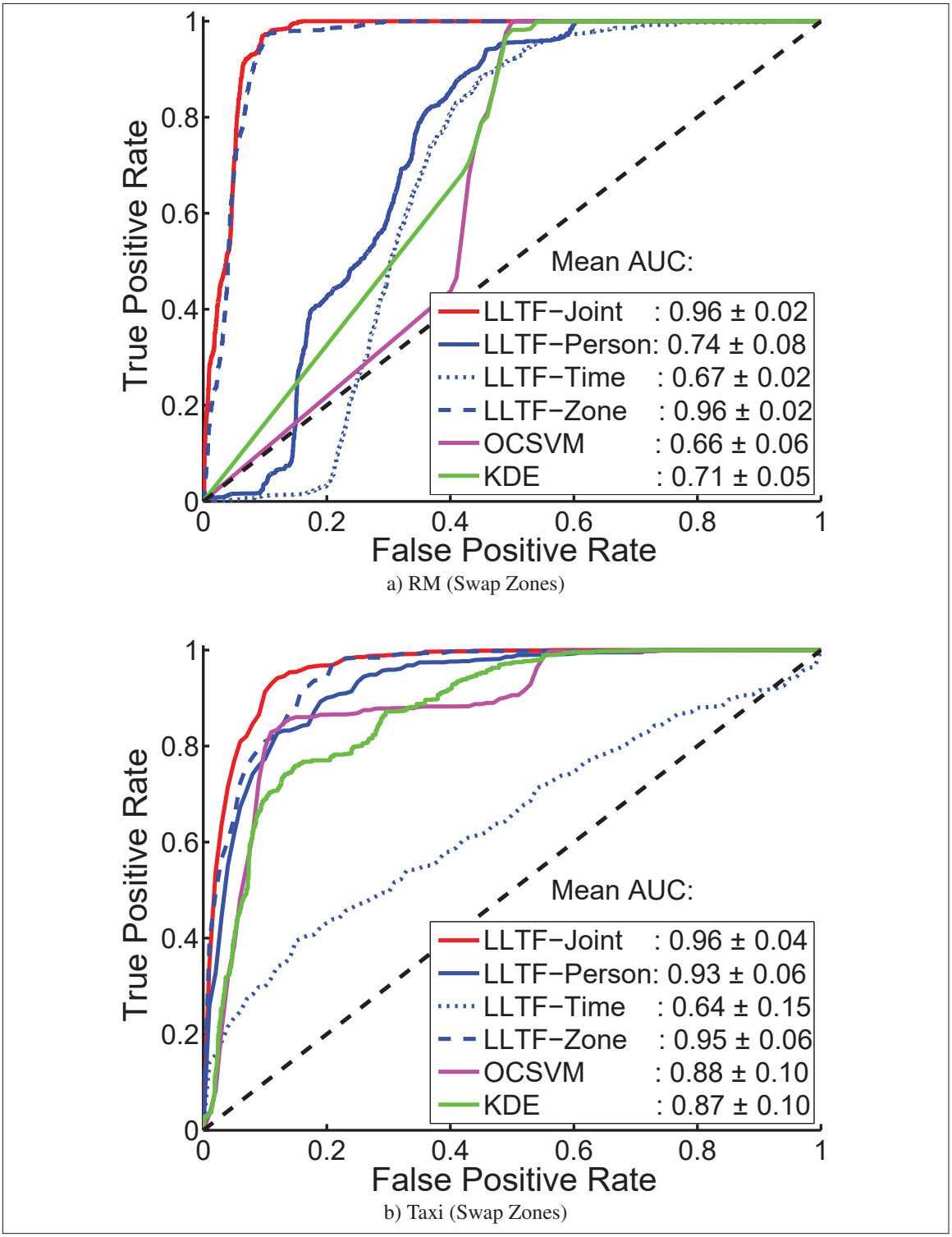
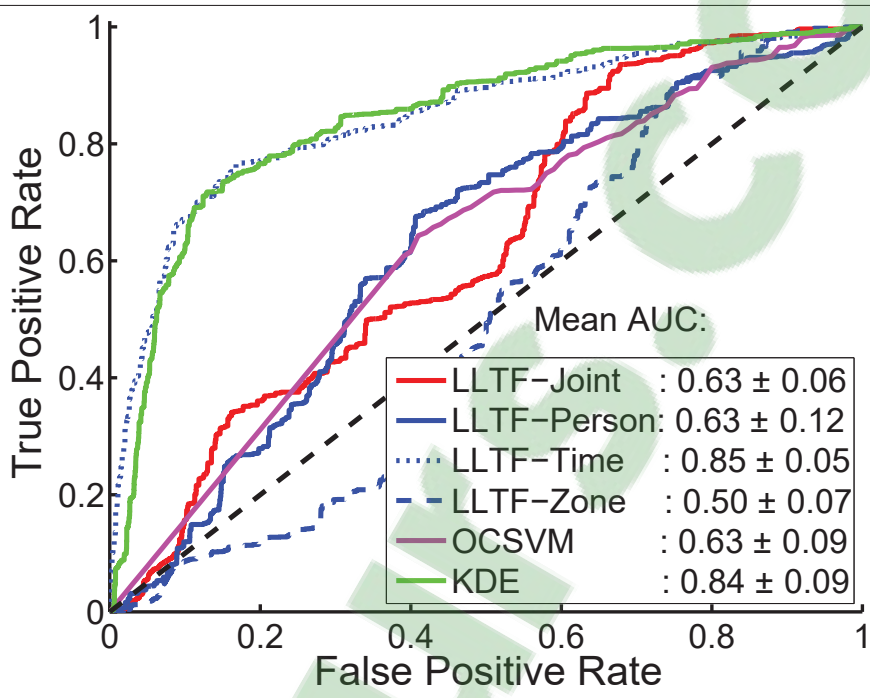
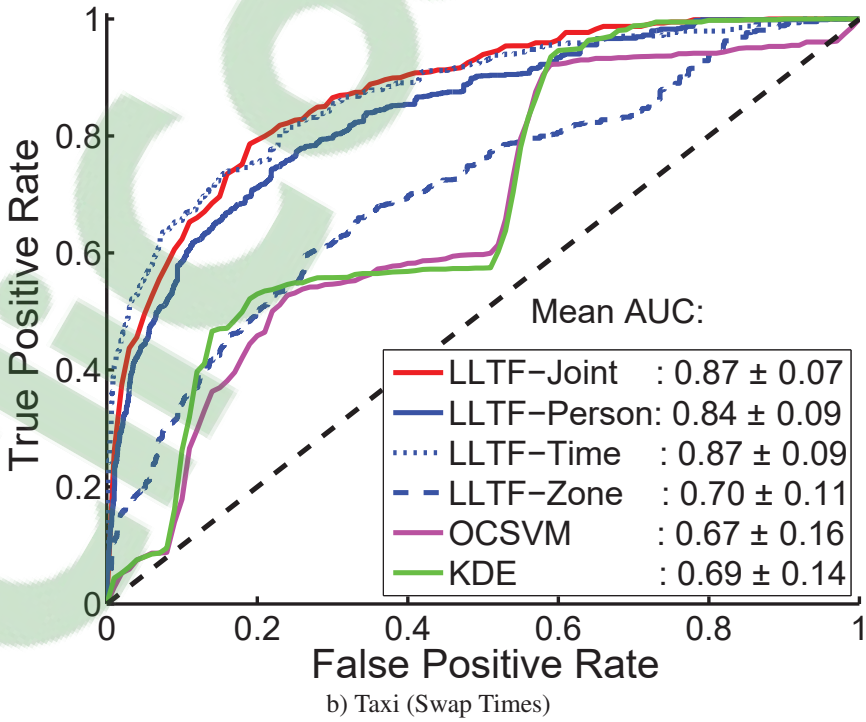



Figure 5.11 Average ROC curves and AUC obtained by LLTF, OCVSM and KDE approaches, on 10 sets of synthetic anomalies generated from the Reality Mining (RM) and Taxi datasets.
Anomaly: Swapping the zone of events corresponding to a person.




a) RM (Swap Times)




b) Taxi (Swap Times)

Figure 5.12 Average ROC curves and AUC obtained by our LLTF, as well as the OCVSM and KDE approaches, on 10 sets of synthetic anomalies generated from the Reality Mining (RM) and Taxi datasets.
Anomaly: Swapping the hour of events corresponding to a person.

0.85 on the Swap Time anomalies of the Reality Mining dataset, whereas this value was only 0.63 for LLTF-Joint.

The proposed method is also faster and more robust than OCSVM and KDE. Thus, training LLTF for the Taxi dataset took less than 10 minutes on a Quad-Core AMD 2.3 GHz processor with 8 GB of RAM, whereas training OCSVM on this dataset required over 7 hours using the same hardware (no training is necessary for KDE). Likewise, predicting anomalies in the test set took on average 0.03 *ms* for LLTF, compared to 61 *ms* for OCSVM and 7 *ms* for KDE. Moreover, while the best parameters for each type of anomaly (as selected in validation) varied greatly in OCSVM and KDE, our method was more robust to the choice of parameters: $K = 25$, $\sigma = 8$ was used for *all* anomalies in the RM dataset, and $K = 15$, $\sigma = 6$ for *all* anomalies in Taxi.

### 5.6 Discussions, Limitations and Recommendations

In this chapter, we presented the efficiency of our proposed approach for decomposing the tensor and learning the factors, predicting future events and mainly detecting abnormal events in real-time. We demonstrated these applications based on the performance of a 3D tensor modeling the environment based on contexts of person/taxi, hour of the day and time.

However, a key limitation of the proposed method is the definition of contextual dimensions. Because the tensor's dimensions need to be clearly defined, incorporating additional dimensions for analyzing events, can be difficult. The environment must thus, be well understood to realize the contexts defining an event.

Another limitation arises with adaptation to changing environments over time. The proposed approach models the event observations by learning a batch of observations. Once the factors have learned the empirical joint probability of the observed events, they are used as is to predict future events and detect abnormal behavior. However, in real-life scenario, the behaviors change and hence, we need a model that adapts to that changing behavior. For this purpose, our model needs to be extended in order to have the latent factor values updated corresponding

to the changing environment. The model in its current batch-learning state, will need to retrain based on all observed events even if the training data contains only one new observations. This approach of having the latent factors learn again from new random initialization is inefficient for long-term application viewpoint. We need to have the latent factors merely *update* upon observing a new normal event and not *re-learn* everything from scratch.

This is where our model's extensibility can be useful. An online version for training the model is presented in Appendix I. Based on this online version, the applications of the proposed approach may be extended to change detection and concept drift.

# CONCLUSION

We presented a new approach, based on LLTF method, for the detection of contextual anomalies. Through this approach, we can perform abnormal behavior detection for *real-time contextual anomaly detection in a large, open environment*.

Real Time Location Systems need to surveil a very large open environment busy with activities. Any of these activities could be malicious and preemptive measures to avoid any undesirable circumstances require almost immediate detection of anything out of the ordinary. The data collected from the environment is also a challenge to organize for analysis. Therefore, we need to analyze the context of these activities/events to make an informed decision corresponding to the *normality* of the event.

From the literature, we realized that probabilistic approaches have proven to be quite popular based on their success for detection of abnormal activities. But their applications have been limited to a closed static environment where it is easy to build a behavioral template of the contextual entities. Trajectory based approaches have also been successful but require repeatable patterns. Due to the application in an open environment, information at a higher-level of complexity, needs to be analyzed. Therefore, we used LSA based tensor factorization models.

The data representation in the form of a higher-order model called tensor allows us to incorporate any number of contextual dimensions considered necessary to analyze the behavior in a given environment. Unlike existing tensor factorization approaches, through our LLTF factorization method, we learn the true probability of events. The log-linear formulation implicitly eliminates need for added complexities corresponding to constraint-integration, such as non-negativity, in the model. Moreover, as demonstrated through the experiments, the model also captures the complex inter-dimensional relations better.

For a faster convergence we show how replacing ordinary gradient descent with Nesterov's accelerated gradient method leads us from a linear to quadratic rate for convergence. We outperformed other methods applied, in terms of training speed. For testing in a real-time environment, LLTF is the fastest in comparison too. For estimating the probability of incoming events and predicting anomalies, LLTF outperformed other methods. It also demonstrated its robustness to the type of anomaly by requiring minimum tuning of parameters, in comparison.

We tested LLTF's application over three problems: the low-rank approximation of synthetic tensors, the prediction of future of events in real-life data and the detection of events representing abnormal behaviors. In all these problems, LLTF has proved to surpass the performance of state of the art methods and showed efficiency over synthetic as well as real-life datasets.

Through our discussions, we have realized certain limitations of the proposed LLTF based approach. One such limitation is of the inconvenience of training the latent factors in batch. To overcome this limitation, we have proposed an online extension of LLTF formulation, which will allow us to update the latent factors as new normal events are observed in the environment. The integration of this extension in the existing approach will extend the application of the proposed method to concept drift and change detection thus, maximizing the productivity as an abnormal behavior detection method.

# APPENDIX I

## ONLINE LOG-LINEAR TENSOR FACTORIZATION

LLTF trains the factors through a batch of events and learns the joint probability. But, as it can be noted from Equation 4.1, the denominator requires computation of all possible events across all the dimensions. In contrast, Equation 3.13 requires only the factors of the observed dimensional entities to get the conditional probability. Given this, we can simply modify by having LLTF model conditional probability instead of joint probability. It could be argued that modeling merely conditional probability won't lead to learning of the true empirical probability of the events. Therefore, we use the concept of *log-pseudolikelihood* which will allow us to have the joint approximation given the conditional probabilities. The details of this model are as follows. For simplicity of explanation, we have initially demonstrated the built of this model using three dimensions and later, generalized to $D$ dimensions.

We have 3 dimensions, $i, j, k$ and their corresponding latent factor matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ (denoted as $\mathcal{Z}$)

A single event observation at instance $x_t = (i_t, j_t, k_t)$

We model the probability of a single dimensional value, conditioned on all other dimensional values as follows:

$$p(i_t \mid j_t, k_t, \mathcal{Z}) = \frac{\exp\left(\langle u_{i_t}, v_{j_t}, w_{k_t} \rangle\right)}{\sum_{i'=1}^{N_1} \exp\left(\langle u_{i'}, v_{j_t}, w_{k_t} \rangle\right)}. \tag{A I-1}$$

Similarly,

$$p(j_t \mid i_t, k_t, \mathcal{Z}) = \frac{\exp\left(\langle u_{i_t}, v_{j_t}, w_{k_t} \rangle\right)}{\sum_{j'=1}^{N_2} \exp\left(\langle u_{i_t}, v_{j'}, w_{k_t} \rangle\right)} \tag{A I-2}$$

and

$$p(k_t \mid i_t, j_t, \mathcal{Z}) = \frac{\exp\left(\langle u_{i_t}, v_{j_t}, w_{k_t} \rangle\right)}{\sum_{k'=1}^{N_3} \exp\left(\langle u_{i_t}, v_{j_t}, w_{k'} \rangle\right)}. \tag{A I-3}$$

We find the latent factors $\mathbf{U},\mathbf{V},\mathbf{W}$ (denoted as $\mathcal{Z}$) using the *log-pseudolikelihood* estimate as follows:

$$\mathcal{G}(\mathcal{Z}) \;=\; \sum_{t=1}^{T} \log p(i_t \,|\, j_t, k_t, \mathcal{Z}) \;+\; \log p(j_t \,|\, i_t, k_t, \mathcal{Z}) \;+\; \log p(k_t \,|\, i_t, j_t, \mathcal{Z}). \tag{A I-4}$$

Based on equations 1,2 and 3, this gives us the cost function with respect to a single event as:

$$\begin{aligned} g_t(\mathcal{Z}) \quad = \quad & 3\,\langle u_{i_t}, v_{j_t}, w_{k_t}\rangle \;-\; \log \sum_{i'=1}^{N_1} \exp\left(\langle u_{i'}, v_{j_t}, w_{k_t}\rangle\right) \\ & -\; \log \sum_{j'=1}^{N_2} \exp\left(\langle u_{i_t}, v_{j'}, w_{k_t}\rangle\right) \;-\; \log \sum_{k'=1}^{N_3} \exp\left(\langle u_{i_t}, v_{j_t}, w_{k'}\rangle\right). \end{aligned} \tag{A I-5}$$

Optimizing it using the SGA approach, we get the derivative of $g_t(\mathcal{Z})$ w.r.t. factor vector $u_i$ as:

$$\begin{aligned} \frac{\partial g_t}{\partial u_i}(\mathcal{Z}) \quad = \quad & \delta(i = i_t)\left[3\,(v_{j_t} \circ w_{k_t}) \;-\; \sum_{j'=1}^{N_2} p(j' \,|\, i_t, k_t, \mathcal{Z})\,(v_{j'} \circ w_{k_t})\right. \\ & \left. -\; \sum_{k'=1}^{N_3} p(k' \,|\, i_t, j_t, \mathcal{Z})\,(v_{j_t} \circ w_{k'})\right] \;-\; p(i \,|\, j_t, k_t, \mathcal{Z})\,(v_{j_t} \circ w_{k_t}). \end{aligned} \tag{A I-6}$$

Similarly, optimizing equation (A I-5) using the SGA approach, we get the derivative of $g_t(\mathcal{Z})$ w.r.t. factor vector $v_j$ as:

$$\begin{aligned} \frac{\partial g_t}{\partial v_j}(\mathcal{Z}) \quad = \quad & \delta(j = j_t)\left[3\,(u_{i_t} \circ w_{k_t}) \;-\; \sum_{i'=1}^{N_1} p(i' \,|\, j_t, k_t, \mathcal{Z})\,(u_{i'} \circ w_{k_t})\right. \\ & \left. -\; \sum_{k'=1}^{N_3} p(k' \,|\, i_t, j_t, \mathcal{Z})\,(u_{i_t} \circ w_{k'})\right] \;-\; p(j \,|\, i_t, k_t, \mathcal{Z})\,(u_{i_t} \circ w_{k_t}). \end{aligned} \tag{A I-7}$$

Finally, optimizing equation (A I-5) using the SGA approach, we get the derivative of $g_t(\mathcal{Z})$ w.r.t. factor vector $w_k$ as:

$$\frac{\partial g_t}{\partial w_k}(\mathcal{Z}) = \delta(k=k_t)\Big[3\,(u_{i_t}\circ v_{j_t}) - \sum_{i'=1}^{N_1} p(i'\,|\,j_t,k_t,\mathcal{Z})\,(u_{i'}\circ v_{j_t}) - \sum_{j'=1}^{N_2} p(j'\,|\,i_t,k_t,\mathcal{Z})\,(u_{i_t}\circ v_{j'})\Big] - p(k\,|\,i_t,j_t,\mathcal{Z})\,(u_{i_t}\circ v_{j_t}). \quad \text{(A I-8)}$$

Starting from randomly initialized latent factor vectors, each SGA iteration modifies these vectors as follows:

$$\mathcal{Z}^{(t+1)} \leftarrow \mathcal{Z}^{(t)} + \eta\,\frac{\partial g_t}{\partial \mathcal{Z}}(\mathcal{Z}^{(t)}) \quad \text{(A I-9)}$$

where $\eta > 0$ controls the step size.

Generalizing this into D dimensions, we have:

$$p(i_d\,|\,\mathcal{I}_{\mathcal{D}\setminus\{d\}},\mathcal{Z}) = \frac{\exp(\langle \mathbf{z}_{i_1},\ldots,\mathbf{z}_{i_D}\rangle)}{\sum_{i'_d}\exp(\langle \mathbf{z}_{i_1},\ldots,\mathbf{z}_{i'_d},\ldots,\mathbf{z}_{i_D}\rangle)}. \quad \text{(A I-10)}$$

where,

$i_d$ is index of dimension $d$, (scalar)

$i_d^t$ is index of dimension $d$ for event $t$, (scalar)

$\mathcal{D} = \{1,\ldots,D\}$ is set of all dimensions, and $N_d$ is the number of elements in dimension $d \in \mathcal{D}$.

$\mathcal{I}_{\mathcal{S}} = \{i_d \mid d \in \mathcal{S} \subseteq \mathcal{D}\}$, i.e. $\mathcal{I}_{\mathcal{S}}$ is the set of indices for a subset of dimensions $\mathcal{S}$.

$\mathcal{Z} = \{Z_1,\ldots,Z_D\}$ is set of all latent factors, where $Z_d \in \mathbb{R}^{N_d\times K}$ is a matrix of $K$ factors corresponding to dimension $d$. We denote as $z_{i_d} \in \mathbb{R}^K$ the factor vector of the $i_d$-th element of dimension $d$, corresponding to the $i_d$-th row of $Z_d$.

Let $\mathcal{X} = \{x^t\}_{t=1}^{T}$ be a set of $T$ events, where $x^t = (i_1^t, \ldots, i_D^t)$. We find the latent factors $\mathcal{Z}$ using the *log-pseudolikelihood* estimate as follows:

$$G(\mathcal{Z}; \mathcal{X}) \;=\; \sum_{t=1}^{T} \sum_{d=1}^{D} \log p(i_d^t \,|\, \mathcal{I}^t_{\mathcal{D}\setminus\{d\}}, \mathcal{Z}) \tag{A I-11}$$

Hence, with respect to a single event $x^t = (i_1^t, \ldots, i_D^t)$,

$$p(i_d^t \,|\, \mathcal{I}^t_{\mathcal{D}\setminus\{d\}}, \mathcal{Z}) \;=\; \frac{\exp(\langle z_{i_1^t}, \ldots, z_{i_D^t}\rangle)}{\sum_{i'_d} \exp\left(\langle \mathbf{z}_{i_1^t}, \ldots, \mathbf{z}_{i'_d}, \ldots, \mathbf{z}_{i_D^t}\rangle\right)}. \tag{A I-12}$$

The log pseudolikelihood estimate w.r.t. event $x^t$ is:

$$\begin{aligned} g_t(\mathcal{Z}) \;&=\; \sum_{d=1}^{D} \log p(i_d^t \,|\, \mathcal{I}^t_{\mathcal{D}\setminus\{d\}}, \mathcal{Z}) \\ &=\; D\,\langle z_{i_1^t}, \ldots, z_{i_D^t}\rangle \;-\; \sum_{d=1}^{D} \log \sum_{i'_d=1}^{N_d} \exp\left(\langle z_{i_1^t}, \ldots, z_{i'_d}, \ldots, z_{i_D^t}\rangle\right). \end{aligned} \tag{A I-13}$$

where, $\langle z_{i_1^t}, \ldots, z_{i'_d}, \ldots, z_{i_D^t}\rangle$ denotes the inner product between the elements.

Optimizing it using the SGA approach, we get the derivative of $g_t(\mathcal{Z})$ w.r.t. factor vector $z_{i_d}$ as:

$$\begin{aligned} \frac{\partial g_t}{\partial z_{i_d}}(\mathcal{Z}) \;=\; & \delta(i_d = i_d^t)\Big[D\,(z_{i_1^t} \circ \ldots \circ z_{i_{d-1}^t} \circ z_{i_{d+1}^t} \circ \ldots \circ z_{i_d^t}) \;- \\ & \sum_{d' \neq d} \sum_{i_{d'}=1}^{N_{d'}} p(i_{d'} \,|\, \mathcal{I}^t_{\mathcal{D}\setminus\{d'\}}, \mathcal{Z})\,(z_{i_1^t}, \circ \ldots \circ z_{i_{d-1}^t} \circ z_{i_{d+1}^t} \circ \ldots \circ z_{i_{d'}} \circ \ldots \circ z_{i_D})\Big] \\ & - p(i_d \,|\, \mathcal{I}^t_{\mathcal{D}\setminus\{d\}}, \mathcal{Z})\,(z_{i_1^t} \circ \ldots \circ z_{i_{d-1}^t} \circ z_{i_{d+1}^t} \circ \ldots \circ z_{i_D^t}). \end{aligned} \tag{A I-14}$$

Thus, with this model, we can attempt to overcome the limitation of batch joint probability learning and eventually, update the factors with a normal event's observation in real-time application.

## BIBLIOGRAPHY


Acar, E. and S. A. Camtepe. 2005. Modeling and multiway analysis of chatroom tensors. Kantor, P. e. a., editor, *Intelligence and Security Informatics*, volume 3495 of *Lecture Notes in Computer Science*, p. 256-268. Springer Berlin Heidelberg. ISBN 978-3-540-25999-2.

Acar, E., S. A. Camtepe, and B. Yener. 2006. Collective sampling and analysis of high order tensors for chatroom communications. *Intelligence and security informatics*, p. 213–224. Springer.

Aggarwal, J. and M. Ryoo. 2011. "Human activity analysis: A review". *ACM Comput. Surv.*, vol. 43, n° 3, p. 16:1–16:43.

Bader, B. W., M. W. Berry, and M. Browne. 2008. Discussion tracking in enron email using parafac. *Survey of Text Mining II*, p. 147–163. Springer.

Bader, B. W., T. G. Kolda, et al. February 2015. "MATLAB Tensor Toolbox Version 2.6". Available online. <http://www.sandia.gov/~tgkolda/TensorToolbox/>.

Benezeth, Y., P.-M. Jodoin, V. Saligrama, and C. Rosenberger. 2009. "Abnormal events detection based on spatio-temporal co-occurences". In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. p. 2458–2465. IEEE.

Bubeck, S. 2013. "Nesterov's Accelerated Gradient Descent". Blog Post. <https://blogs.princeton.edu/imabandit/2013/04/01/acceleratedgradientdescent/>.

Chang, C.-C. and C.-J. Lin. 2011. "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology*, vol. 2, p. 27:1–27:27.

Chi, E. C. and T. G. Kolda. 2012. "On Tensors, Sparsity, and Nonnegative Factorizations". *SIAM Journal on Matrix Analysis and Applications*, vol. 33, n° 4, p. 1272-1299.

Chin-De Liu, Y.-N. C. and P.-C. Chung. sept. 2010. "An Interaction-Embedded HMM Framework for Human Behavior Understanding: With Nursing Environments as Examples". *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, n° 5, p. 1236 -1246.

Choudhury, T., S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, et al. 2008. "The mobile sensing platform: An embedded activity recognition system". *Pervasive Computing, IEEE*, vol. 7, n° 2, p. 32–41.

Chung, P.-C. and C.-D. Liu. 2008. "A daily behavior enabled hidden Markov model for human behavior understanding". *Pattern Recognition*, vol. 41, n° 5, p. 1572 - 1580.

Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. "Indexing by latent semantic analysis". *JASIS*, vol. 41, n° 6, p. 391–407.

Derakhshan, R., M. Orlowska, and X. Li. 2007. "RFID Data Management: Challenges and Opportunities". In *RFID, 2007. IEEE International Conference on*. p. 175-182.

Du, Y., F. Chen, W. Xu, and Y. Li. 0-0 2006. "Recognizing Interaction Activities using Dynamic Bayesian Network". In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. p. 618 -621.

Dumais, S. T. 2004. "Latent semantic analysis". *Annual review of information science and technology*, vol. 38, n° 1, p. 188–230.

Dumais, S. T., G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. "Using latent semantic analysis to improve access to textual information". In *Proceedings of the SIGCHI conference on Human factors in computing systems*. p. 281–285. ACM.

Eagle, N. and A. S. Pentland. 2009. "Eigenbehaviors: Identifying structure in routine". *Behavioral Ecology and Sociobiology*, vol. 63, n° 7, p. 1057–1066.

et. al., T. D. june 2005. "Activity recognition and abnormality detection with the switching hidden semi-Markov model". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*. p. 838 - 845 vol. 1.

Fawcett, T. 2006. "An introduction to ROC analysis". *Pattern recognition letters*, vol. 27, n° 8, p. 861–874.

Frank, J. ". Personal Communication via email.

Frank, J., S. Mannor, and D. Precup. 2011. "Activity recognition with mobile phones". In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III*. (Berlin, Heidelberg 2011), p. 630–633. Springer-Verlag.

Gonzalez, H., J. Han, X. Li, and D. Klabjan. 2006. "Warehousing and Analyzing Massive RFID Data Sets". In *Proceedings of the 22Nd International Conference on Data Engineering*. (Washington, DC, USA 2006), p. 83–. IEEE Computer Society.

Hara, K., T. Omori, and R. Ueno. 2002a. "Detection of unusual human behavior in intelligent house". In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*. p. 697 - 706.

Hara, K., T. Omori, and R. Ueno. 2002b. "Detection of unusual human behavior in intelligent house". In *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing, 2002*. p. 697–706. IEEE.

Hayashi, K., T. Takenouchi, T. Shibata, Y. Kamiya, D. Kato, K. Kunieda, K. Yamada, and K. Ikeda. 2010. "Exponential family tensor factorization for missing-values prediction and anomaly detection". In *2010 IEEE 10th International Conference on Data Mining (ICDM)*. p. 216–225. IEEE.

Hu, D. H., X.-X. Zhang, J. Yin, V. W. Zheng, and Q. Yang. 2009. "Abnormal Activity Recognition Based on HDP-HMM Models.". In *IJCAI*. p. 1715–1720.

Huang, W., J. Zhang, and Z. Liu. 2007. "Activity recognition based on hidden Markov models". In *Proceedings of the 2nd international conference on Knowledge science, engineering and management*. (Berlin, Heidelberg 2007). Springer-Verlag.

Hui-Huang Hsu,Zixue Cheng, S. T. and C.-C. Chen. july 2009. "RFID-Based Personalized Behavior Modeling". In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC '09. Symposia and Workshops on*. p. 350 -355.

Jiang, F., J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos. 2011. "Anomalous video event detection using spatiotemporal context". *Computer Vision and Image Understanding*, vol. 115, n° 3, p. 323–333.

Kim, H., S. Lee, X. Ma, and C. Wang. 2009. "Higher-order PCA for anomaly detection in large-scale networks". In *2009 3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. p. 85–88. IEEE.

Kolda, T. and B. Bader. 2009. "Tensor Decompositions and Applications". *SIAM Review*, vol. 51, n° 3, p. 455-500.

Koutra, D., E. E. Papalexakis, and C. Faloutsos. 2012. "TensorSplat: Spotting Latent Anomalies in Time". In *Informatics (PCI), 2012 16th Panhellenic Conference on*. p. 144–149. IEEE.

Latecki, L. J., A. Lazarevic, and D. Pokrajac. 2007. "Outlier Detection with Kernel Density Functions". In *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*. (Berlin, Heidelberg 2007), p. 61–75. Springer-Verlag.

Lee, C.-H. and C.-W. Chung. 2008. "Efficient Storage Scheme and Query Processing for Supply Chain Management Using RFID". In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. (New York, NY, USA 2008), p. 291-302. ACM.

Lee, D. D. and H. S. Seung. 2001. "Algorithms for non-negative matrix factorization". In *Advances in neural information processing systems*. p. 556–562.

Li, C., Z. Han, Q. Ye, and J. Jiao. 2011. "Abnormal behavior detection via sparse reconstruction analysis of trajectory". In *Image and Graphics (ICIG), 2011 Sixth International Conference on*. p. 807–810. IEEE.

Lin, Y.-S., S.-M. Chang, J. C. Tsai, T. K. Shih, and H.-H. Hsu. 2011. "Motion analysis via feature point tracking technology". In *Proceedings of the 17th international conference on Advances in multimedia modeling - Volume Part II*. (Berlin, Heidelberg 2011), p. 296–303. Springer-Verlag.

Malik, A., 2009. *RTLS for Dummies*.

Manning, C. D., P. Raghavan, and H. Schütze. 2008. "Flat clustering". *Introduction to information retrieval*, p. 350–374.

Müller, M., November 2007. *Information Retrieval for Music and Motion*, p. 69-84. Springer-Verlag New York, Inc., Secaucus, NJ, USA. ISBN 3540740473. Dynamic Time Warping.

N. Eagle, A. P. and D. Lazer. 2009. "Inferring Social Network Structure using Mobile Phone Data". *Proceedings of the National Academy of Sciences (PNAS)*, vol. 106(36), p. 15274-15278.

Nesterov, Y. 2013. "Gradient methods for minimizing composite functions". *Mathematical Programming*, vol. 140, n° 1, p. 125–161.

Oono, K. 2012. "(2012)". <http://www.slideshare.net/KentaOono/tensordecomposition>.

Piciarelli, C., C. Micheloni, and G. L. Foresti. 2008. "Trajectory-based anomalous event detection". *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, n° 11, p. 1544–1554.

Reiss, A. and D. Stricker. 2012. "Creating and benchmarking a new dataset for physical activity monitoring". In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*. (New York, NY, USA 2012). ACM.

Robert Fisher,Jose Santos-Victor, J. C. 2004. "CAVIAR Test Case Scenarios". URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

Sillito, R. R. and R. B. Fisher. 2008. "Semi-supervised Learning for Anomalous Trajectory Detection.". In *BMVC*. p. 1–10.

Sun, J., S. Papadimitriou, and S. Y. Philip. 2006a. "Window-based Tensor Analysis on High-dimensional and Multi-aspect Streams". In *ICDM*. p. 1076–1080.

Sun, J., D. Tao, and C. Faloutsos. 2006b. "Beyond streams and graphs: dynamic tensor analysis". In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. p. 374–383. ACM.

Sutskever, I., J. Martens, G. Dahl, and G. Hinton. 2013. "On the importance of initialization and momentum in deep learning". In *Proceedings of the 30th international conference on machine learning (ICML-13)*. p. 1139–1147.

Tork, H. F., M. Oliveira, J. Gama, S. Malinowski, and R. Morla. 2012. "Event and anomaly detection using Tucker3 decomposition". In *Workshop on Ubiquitous Data Mining*. p. 8.

Wang, F. and P. Liu. 2005. "Temporal Management of RFID Data". In *Proceedings of the 31st International Conference on Very Large Data Bases*. p. 1128–1139. VLDB Endowment.

Welling, M. and M. Weber. 2001. "Positive tensor factorization". *Pattern Recognition Letters*, vol. 22, n° 12, p. 1255–1261.

Yuan, J., Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. 2010. "T-drive: driving directions based on taxi trajectories". In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*. p. 99–108. ACM.

Zhang, X., H. Peng, and Q. Zheng. 16-20 2006. "A Novel Ant Colony Optimization Algorithm for Clustering". In *Signal Processing, 2006 8th International Conference on*.