



MACHINE LEARNING-BASED APPROACHES FOR LARGE-SCALE TEMPORAL DATA ANALYTICS

A thesis submitted
for a degree of Doctor of Philosophy
by

SEYED ALI MAJID ZONOOZI

School of Computer Science and Engineering
Nanyang Technological University

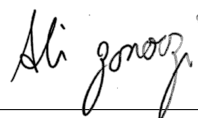
August 2018

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

2 March 2019

Date



Seyed Ali Majid Zonoozi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

March 5 2019

Date



Prof. Cong Gao

Authorship Attribution Statement

This thesis contains material from 2 paper(s) published in the following papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Zonoozi, Ali, Qirong Ho, Shonali Krishnaswamy, and Gao Cong. "ConTrack: A Scalable Method for Tracking Multiple Concepts in Large Scale Multidimensional Data." In 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 759-768. IEEE, 2016.

The contributions of the co-authors are as follows:

- Assoc. Prof. Shonali Krishnaswamy provided the initial project direction and commented on the manuscript drafts.
- I did the literature review on the problem and identified the research gaps.
- Dr. Qirong Ho provided assistance in formulation of the problem and designing the experiments.
- I co-designed the study with Dr. Qirong Ho, implemented the model and performed all the experimental studies at the School of Computer Science and Engineering in NTU and the Data Analytics Department of Institute for Infocomm Research, A*STAR. I also analysed the data and results together with Dr. Qirong Ho and Assoc. Prof. Shonali Krishnaswamy.
- I created the synthetic and partly-synthetic datasets and collected the real data for the experimental studies.
- I prepared the manuscript draft. The manuscript was further revised by Dr. Qirong Ho and Prof. Cong Gao.

Chapter 4 is published as Zonoozi, Ali, Jung-jae Kim, Xiao-Li Li, and Gao Cong. "Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns." In IJCAI, pp. 3732-3738. 2018.

The contributions of the co-authors are as follows:

- Prof. Cong Gao provided the initial project direction and revised the manuscript.
- I did the literature review on the problem and identified the research gaps.
- I wrote the drafts of the manuscript. The manuscript was revised together with Dr. Kim Jung-jae and Assoc. Prof. Li Xiaoli and Dr. Cong Gao.
- I formulated the problem and designed the model with assistance of Dr. Kim Jung-jae and Assoc. Prof. Li Xiaoli.
- I collected the data from Institute for Infocomm Research, A*STAR and further processed the datasets for the experimental studies.
- I implemented the model and conducted all the experimental studies at the School of Computer Science and Engineering in NTU. I also analysed the results together with Dr. Kim Jung-jae, Assoc. Prof. Li Xiaoli and Prof. Cong Gao.

2 March 2019

Date



Seyed Ali Majid Zonoozi

Acknowledgments

I would like to express my sincere gratitude and appreciation towards my supervisors Dr. Cong Gao from Nanyang Technological University (NTU) and Dr. Li Xiaoli and Dr. Shonali Krishnaswamy from Institute for Infocomm Research (I2R), A*STAR, for their support, encouragement and guidance that they provided throughout my PhD studies. It was a great honor for me to work with them and learn from them during the past few years. I am also sincerely grateful to all my advisors and collaborators, especially Dr. Kim Jung-Jae, Dr. Qirong Ho, Dr. João Bártolo Gomes and Dr. Wu Wei, who shared their knowledge and expertise with me unconditionally and provided invaluable advice and aspiring comments throughout different phases of my studies. Completion of this dissertation would not have been possible without their help and support. I would also like to thank A*STAR Graduate Academy (A*GA) for supporting me through the Singapore International Graduate Award (SINGA).

I would like to thank all my supportive friends, Iranians and non-Iranians, in NTU and outside NTU who made my stay in Singapore memorable. A special thanks goes to my housemates Meshal, Biswa, Adam and Viktor who were amazing and supportive friends. I would also like to thank all my labmates from Data Management and Analytics Lab (DMAL) in NTU for sharing their knowledge and ideas with me. Last but not least, I would like to sincerely thank my amazing family for their endless love, support, encouragement and prayers. Words cannot express how grateful I am for their presence in my life.

Contents

Statement of Originality	i
Supervisor Declaration Statement	ii
Authorship Attribution Statement	iii
Acknowledgments	v
Abstract	xi
List of Figures	xiii
List of Tables	xiv
1 Introduction	1
1.1 Background and Motivations	1
1.2 Problems and Research Scope	5
1.2.1 Unsupervised Concept Tracking in Large-Scale Multidimensional Data	5
1.2.2 Next State Prediction of Spatio-Temporal Data with Recurring Periodic Patterns	5
1.2.3 Temporal Anomaly Detection for Geo-Spatio-Temporal Data	6
1.3 Approach and Methodology	7
1.3.1 Concept Tracking (problem 1.2.1)	8
1.3.2 Next States Prediction (problem 1.2.2)	9
1.3.3 Temporal Anomaly Detection (problem 1.2.3)	10
1.4 Summary of Contributions	11
2 Literature Survey	13
2.1 Preliminaries	13
2.1.1 Optimization	13
2.1.2 Neural Networks	15

2.1.3	Convolutional Neural Networks	16
2.1.4	Recurrent Neural Networks	17
2.2	Concept Drift in Temporal Data	20
2.2.1	What is Concept Drift	21
2.2.2	Concept Drift Handling Methods	22
2.3	Unsupervised Methods for Temporal Data	24
2.3.1	Evolutionary Clustering	25
2.3.2	Matrix Factorization based Methods	25
2.3.3	Sparse Coding based Methods	26
2.3.4	Dynamic Topic Models	27
2.4	Predictive Models for Temporal Data	29
2.4.1	Time-Series Forecasting	29
2.4.2	Spatio-Temporal Prediction	32
2.5	Anomaly Detection in Temporal Data	35
2.5.1	Contextual Anomaly Detection	35
2.5.2	Anomaly Detection in Time-Series Data	36
2.5.3	Anomaly Detection in Spatio-Temporal Data	36
3	Unsupervised Concept Tracking In Large Scale Multidimensional Data	40
3.1	Problem Statement	40
3.2	The Proposed Model	42
3.3	Implementation and Optimization	45
3.3.1	Stochastic Gradient Descent Equations	46
3.3.2	Adaptive Gradient	46
3.3.3	Simplex projection	47
3.3.4	Distributed-Parallel Algorithm	47
3.4	Experiments	49
3.4.1	Experimental Setting	49
3.4.2	Evaluation Procedure	51
3.4.3	Synthetic and Semi-Synthetic Experiments	53
3.4.4	Real-World Activity Recognition Experiments	53
3.4.5	Scalability Experiments	55
3.5	Summary	56

4	Next State Prediction of Spatio-Temporal Data with Recurring Periodic Patterns	58
4.1	Problem Statement	58
4.2	The Proposed Method	59
4.2.1	Pyramidal CRN Model	60
4.2.2	Periodic Representations	61
4.2.3	Fusion Process	63
4.3	Implementation and Optimization	64
4.4	Experiments	65
4.4.1	Experimental Settings	65
4.4.2	Next Frame Prediction	67
4.4.3	Representation Update Method and Fusion Mechanism	68
4.4.4	Multi-Step Ahead Prediction	68
4.5	Summary	70
5	Temporal Anomaly Detection in Geo-Spatial Data	71
5.1	Problem Statement	71
5.2	Geo-Spatio-Temporal Anomaly Detection (GSTAD) Model	72
5.2.1	3D Convolutional Autoencoder	73
5.2.2	Temporal metadata prediction	74
5.2.3	Identification of Abnormal Regions	77
5.3	Implementation and Optimization	78
5.4	Experiments	78
5.4.1	Experimental Settings	78
5.4.2	Anomaly Detection Performance	83
5.4.3	Model Architecture Analysis	84
5.4.4	Visualization of Temporal Anomalies	86
5.5	Summary	87
6	Conclusions and Future Work	88
6.1	Conclusions	88
6.2	Future Directions	89

A List of Publications	92
Appendix - List of Publications	92
References	93

Abstract

In many domains such as telecommunications, finance and sensor monitoring, large volumes of unlabeled temporal data are continuously generated in a sequential format, where the time-stamps of the generated records are available. From a data analysis standpoint, there is significant utility to be gained by discovering the patterns in the data, in order to understand the trend of changes in the past and future. In this study, by considering different types of changes that occur in temporal data, including *changes in underlying data distribution*, *recurring concepts* and *outliers*, different machine learning-based models are proposed to track the changes, predict future values and detect abnormal patterns in temporal settings.

The phenomenon of concept drift refers to the changes in the underlying data distribution over time. Many studies have been conducted to handle the concept drift problem in supervised settings, where the changes in the underlying data distribution can adversely affect supervised models. However, limited work has been done from an unsupervised point of view, in order to understand the changes in data distribution in an interpretable way. In our first study, a scalable optimization model is proposed to track multiple concepts and participation of actors in each concept over time, in order to track the changes in underlying distribution of data. The proposed concept tracking method applies to problem settings that cannot be handled by existing concept drift and stream mining methods, and outperforms popular unsupervised baselines from the wider Data Mining and Machine Learning literature.

Recurring concepts is a form of concept drift, where a previously observed pattern recurs in the data after some time. Understanding and utilizing such patterns are beneficial, especially in forecasting problem. In some spatio-temporal settings such as crowd density in urban environments, recurring periodic patterns can be observed, which have not been considered explicitly in previous works on spatio-temporal forecasting. To address this issue, a novel deep learning based method is proposed in our second study, to accurately capture spatial and temporal correlations in geo-spatial data. The proposed model is able to learn and incorporate explicit periodic representations in different temporal scales, and can be optimized with multi-step ahead prediction. By conducting experiments on two real world taxi datasets, we showed that the proposed prediction model, enhanced with periodic representations, outperforms other linear and non-linear prediction baselines.

Outliers are another type of change in temporal data. Current approaches for outlier detection in spatio-temporal settings use reconstruction-based or representation learning-based approaches to learn the normal pattern of data and detect instances which deviate from the normal

pattern (global anomalies). These approaches fail to detect temporal anomaly instances, which are similar to normal instances, but occur in an unusual time context. To address this problem, a temporal anomaly detection model is proposed in our last study. The proposed model can effectively capture spatial and temporal dependencies in spatio-temporal sequences and detect abnormal crowd density patterns in geo-spatial domain, using temporal meta-data prediction error. The conducted experiments that used real world taxi datasets, shows the effectiveness of the proposed model in detecting abnormal crowd density patterns in geo-spatio-temporal environments, that cannot be detected by the existing reconstruction-based anomaly detection methods.

List of Figures

1.1	Patterns of changes in data over time	2
1.2	Example of recurring periodic patterns in traffic data	3
1.3	Visualization of changes in actor participations and concepts in activity data . .	6
1.4	Overview of the research problems	11
2.1	A schematic diagram of neural network	15
2.2	An unfolded recurrent neural network architecture	18
2.3	Architecture of general ensemble learning model	24
2.4	Matrix Factorization method	26
2.5	Graphical representation of LDA	28
2.6	Graphical representation of dynamic topic models	29
3.1	Visualization of changes in actor participations and concepts in activity data . .	43
3.2	Unimodal vs. multimodal loss functions	44
3.3	Simplex projection	47
3.4	Acceleration time series data on X axis for different activities performed by one participant.	54
3.5	Speedup achieved for different staleness settings	56
4.1	Examples of 32×32 partitioned spatial images	60
4.2	Weighting based fusion model architecture	61
4.3	Comparison of $PCRN_{seq}$ with ARIMA, VAR and ST-ResNet on multi-step prediction errors up to 6 steps ahead.	69
4.4	Multi-step and one step ahead prediction errors with variable interval lengths .	70
5.1	Examples of spatial images from Singapore and Beijing taxi dataset. Brighter pixels represent higher crowd density.	72
5.2	Proposed model architecture	73
5.3	3D visualization of the regression space for temporal metadata prediction . . .	76
5.4	Examples of temporal anomaly heatmaps on Singapore taxi dataset	86
6.1	Overview of the proposed models	88

List of Tables

3.1	Table of key terms and notation.	42
3.2	Synthetic, semi-synthetic and real-world datasets statistics	51
3.3	NMI scores obtained by each method on synthetic and real datasets	52
3.4	Features added to the original time-series motion data.	55
3.5	NMI score for ConTrack and kmeans on activity recognition data	55
3.6	Iterations per second and NMI scores	57
4.1	The characteristic of two benchmark datasets	66
4.2	Next step prediction results	68
4.3	Comparison over different PCRN variants and fusion types	69
4.4	Number of training images for each interval length for TaxiSG dataset	70
5.1	The characteristic of datasets	80
5.2	Annotated anomaly timestamps in TaxiSG and TaxiBJ datasets	82
5.3	AUC scores obtained by different models on TaxiSG and TaxiBJ datasets	83
5.4	The misclassified instances of GSTAD-Cls models on odd test-days dataset	84
5.5	AUC scores obtained by different model architectures	85

Chapter 1

Introduction

1.1 Background and Motivations

Data plays a crucial role in every aspect of human life today. People are increasingly relying on data-oriented services for their daily activities, and businesses are striving to make use of their domain data in order to enhance their performance, generate more revenue and avoid possible risks in the future. Massive amounts of data is generated in real-time by a wide range of devices, from small sensors to large satellites, in structured and unstructured forms such as numbers, texts, images, audios and videos.

In many real world applications, data is generated in streaming format as a continuous sequence of data records. Data generated from various types of sensors (e.g. temperature or pressure sensors, GPS devices, accelerometers etc.), financial transactions, click-stream data from internet users, audio and video signals are just some examples of data streams. When the sequential data records contain time-stamp information that represent a state in time, they are referred to as ‘temporal data’. In a discrete setting, time-dependent (temporal) data X , takes the following form:

$$X = \{(t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)\}, \quad (1.1)$$

where t refers to the time-stamp information and x refers to a data record or a state.

Data generation process is non-stationary in many real world applications. In such environments, the statistical properties of data such as mean, standard deviation and covariance of features are changing over time. Such changes in underlying distribution of temporal data make analyzing them a challenging task. Typically, the data is processed in batch mode in many applications, in which the temporal dimension is collapsed and any trend of change is eliminated. However, the dynamic patterns existing in the temporal dimension of data can provide valuable insights on the trends of change in the past, and prediction of values in the future.

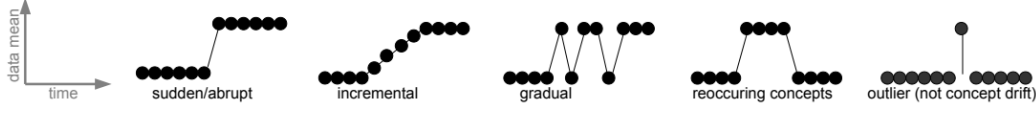


Figure 1.1: Patterns of changes in data over time, obtained from [GZB⁺14]¹. Outliers are not a type of drift.

In their survey on concept drift adaptation methods, Gama et al. outline different forms of changes in temporal data as *sudden drift*, *incremental drift*, *gradual drift*, *recurring concepts* and *outliers*, as illustrated in Figure 1.1 [GZB⁺14]. In this study, we generally refer to the three types of sudden, incremental and gradual drift as concept drift. The recurring concepts is also another type of drift, where a previously observed concept reoccurs after some time. Another type of change in temporal data is outliers, which should not be mistaken for concept drift, as they are temporary deviations from the data distribution, in the form of noise. Understanding each type of change in temporal settings, is important and beneficial in different problems and applications.

Concept Drift: In many industrial applications, a large volume of streaming or temporal data is generated on a regular basis, and there is a strong incentive to mine such data in order to enhance decision making processes. For example, in activity recognition applications, time series motion data from millions of users can be captured using different wearable sensors connected to the internet, and detecting changes in motion patterns is essential for recognizing which physical activity the user is engaging in (e.g. walking, running, sleeping, ...) [LL13]. Another example is internet advertising, where mining user click streams and profile data is the core revenue generator for many firms [MHS⁺13, YGH⁺15]. A key challenge in this setting is that of concept drift, the observation that actor (customers, users, etc.) behavior tends to change over time. In supervised learning settings (e.g. regression, classification) [GMCR04, ZZS08, SAK14, WFYH03, KM⁺03, HSD01], accounting for drift keeps predictive models accurate over long periods of time, while in unsupervised learning (e.g. clustering) [AHE⁺11, HSX11], the goal is to detect the occurrence of such concept drifts, and present the information for easy interpretation. Since the data is unlabeled and high-dimensional in most real world scenarios, visualizing and understanding the changes are challenging and require algorithms that provide the outputs in a way that produces interpretable insight.

Recurring Concepts: Recurring concepts is a type of drift in temporal data, in which a previously observed concept recurs after a period of time. Many spatio-temporal time-series data in geo-spatial domains show recurring concepts in the form of *periodic patterns* over different time scales, such as days or weeks. For instance, the number of cars passing a pre-defined

¹The figure is obtained from Figure 2 of [GZB⁺14]

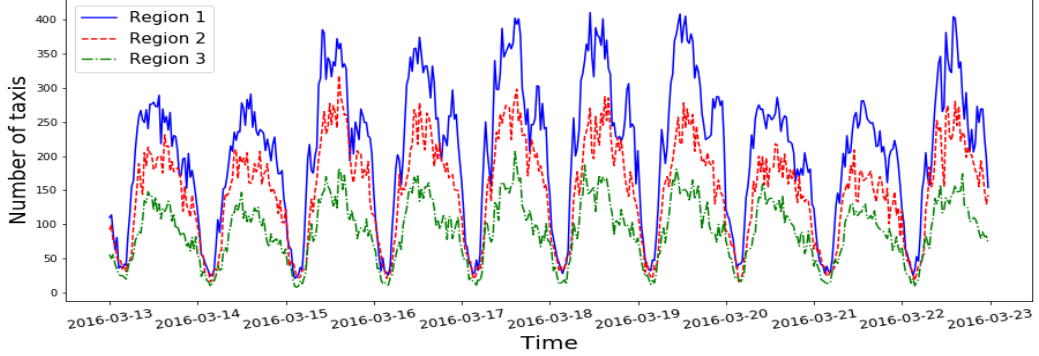


Figure 1.2: Example of recurring periodic patterns in traffic data. Number of taxis flowing into 3 different regions of size $750m * 750m$ are shown over period of 10 days in Beijing city. The time-series data show clear *daily* and *weekly* recurring periodic patterns.

bounded region usually increases in rush hours and decreases over nights, and this pattern repeats everyday. On a weekly scale, the traffic patterns in weekends are similar to each other, compared to the weekday patterns. Some examples of these periodic patterns can be observed in Figure 1.2.

Observing and understanding the recurring concepts in the temporal settings could be extremely beneficial for the prediction problem. In this problem, given a set of temporal data points $\{X^{(1)}, X^{(2)}, \dots, X^{(t)}\}$, the goal is to predict the values of $\{X^{(t+1)}, \dots, X^{(t+\tau)}\}$ for an arbitrary horizon τ , as accurately as possible. From a machine learning point of view, a prediction model can learn from patterns that have occurred in the historical data (recurring concepts in this case), to better predict the future values in the next period.

Extensive amounts of research have been performed on the problem of forecasting in the last decades, and various statistical and machine learning-based approaches (linear and non-linear models) have been proposed [Cha00]. In the past few years, numerous spatio-temporal forecasting techniques and applications have been studied, including road traffic forecast [LBF13, LDK⁺15], human mobility pattern prediction for urban planning [ZCWY14], as well as next frame prediction in videos [SMS15, FGL16]. However, limited work has been done to incorporate the recurring periodical patterns into the prediction models, especially in spatio-temporal settings.

Outliers: Outliers are another type of change in temporal data. Detection of outliers are crucial in many applications such as fraud detection [NHW⁺11], detection of faulty sensors [ZMH10], network intrusion detection [GTDVMFV09] and abnormal event detection in surveillance videos [HCN⁺16]. For instance, in the geo-spatial domain, sudden surge in traffic volume in different regions of a city could be detected in real-time and managed. In the telecommunication industry, sudden increase or decrease in mobile data usage could be monitored for optimized resource allocation.

The supervised approaches are generally not suitable for outlier detection task as they require large amount of annotated anomalous data, which are not easily obtainable in many applications. Moreover, the inherent imbalance between positive and negative instances makes it challenging to learn efficient classifiers [BBK14]. The unsupervised methods generally model the normal distribution of data through probabilistic approaches or clustering methods, and detect instances that significantly deviate from the normal pattern [CBK09]. Such methods work well for low dimensional data but when the dimension of data grows, a small variation in each dimension can result in high deviation from data distribution. Therefore, the classic outlier detection methods are not applicable to high-dimensional data. To address this challenge, many existing unsupervised anomaly detection models on spatio-temporal data such as videos, learn the normal pattern of data using reconstruction-based approaches, and identify the instances that deviate from the normal patterns based on the reconstruction error. In spatio-temporal domain, there has been a lot of research in recent years on detection of abnormal events in videos [KTP18]. However, relatively less work has been done on anomaly detection in geo-spatio-temporal domains.

Reconstruction-based or representation learning based anomaly detection approaches usually learn a single normality pattern from all data instances. However, some anomalies might not deviate from the learned model but may be abnormal with respect to some *temporal context*. In anomaly detection literature, these patterns usually are referred to as ‘contextual anomalies’ [CBK09]. For instance, high traffic in a highway is not abnormal when it is observed during rush hours. However, the same pattern is considered abnormal when it happens at midnight and an anomaly detection model should be able to detect this.

To summarize, considering the aforementioned types of changes in temporal data, we aim to address the following problems for analyzing temporal data in this thesis:

- **Tracking** the *changes of multiple concepts* in temporal data over time.
- **Predicting** the future state, by utilizing the *recurring concepts* in temporal data.
- **Detecting** abnormal patterns in data by identifying *temporal outliers*.

These problems will be introduced in more detail in the following section.

1.2 Problems and Research Scope

In this thesis, we focus on three main problems described as follows:

1.2.1 Unsupervised Concept Tracking in Large-Scale Multidimensional Data

In many industrial applications such as telecommunications, finance and marketing, temporal data is generated in large-scale volumes. From a data analytics point of view, understanding, tracking and visualizing the changes in such massive unlabeled data could provide valuable insight over the existing trends in the data, and could be leveraged for better decision making processes. The existing work on concept drift problem usually focuses on supervised settings [vZ110], where the changes in data distribution makes supervised models inaccurate or obsolete. Comparatively, limited work has been done to understand the changes in temporal data from an unsupervised point of view.

In order to address the above challenge, we aim to design a model for tracking and detecting changes in unsupervised data, and implement it as a distributed algorithm that can run on a cluster of machines, in order to scale to real-world Big Data volumes. For example, in the physical activity recognition domain, we envision that our method will detect patterns that intuitively correspond to different physical motions and gestures: walking, running, standing, sleeping, and so forth. Such motion patterns (the concepts) are shared among many people (the actors), and the degree to which each user participates in each pattern can vary depending on the activity the individual performs. However, user activities may not always be clearly distinguishable — during exercise, one might alternate between walking and running, or jumping and running, so the participation is now a mix of two or more activities. Whereas supervised activity recognition methods usually predict only one activity at a time [CPR11, KLLK10, PEK⁺06], our goal is to design a method which detects and tracks a mix of activities (concepts) performed by the same actor, at the same time. Figure 1.3 provides a simple illustration of these ideas.

1.2.2 Next State Prediction of Spatio-Temporal Data with Recurring Periodic Patterns

In many geo-spatio-temporal domains, there are patterns (concepts) that recur over different periods of time. For instance, monthly precipitation rate in a specific region shows periodical pattern over years. The traffic density in a highway increases every day in the rush hours and decreases over midnights. There are also recurring weekly patterns in traffic data which show similar patterns in the same weekdays.

In this problem, we study the problem of crowd density prediction in geo-spatio-temporal settings, where the data shows recurring periodic patterns over different time-scales such as

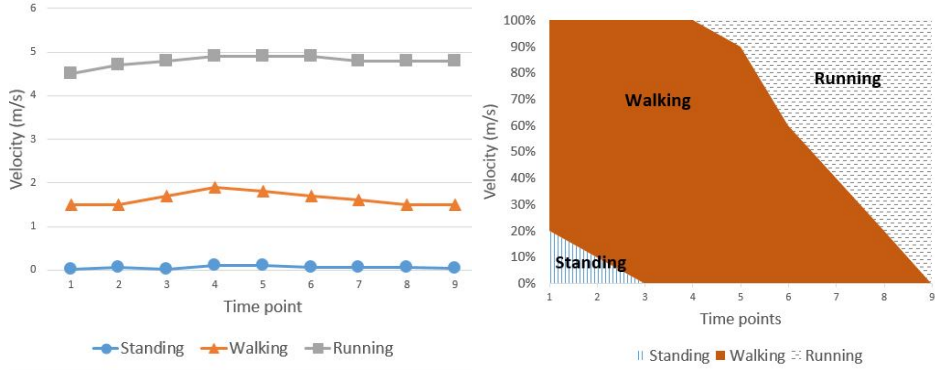


Figure 1.3: Visualization of changes in actor participations and concepts in activity data. **Left:** Tracking the changes in the velocity feature of standing, walking and running concepts over time. **Right:** Tracking the changes in the actor’s participation for standing, walking and running concepts over time.

days, weeks, etc. Crowd density estimation and crowd flow prediction is one of the main problems in the spatial-temporal forecasting domain, which has been the focus of extensive amount of research in the past few years. In a geo-spatial domain such as a city, the spatial area under study can be represented as a high-dimensional image. Using such data, the problem of crowd density prediction can be defined similar to next frame prediction in videos, in order to predict the crowd density values for all of the regions in the next time-interval. Providing such predictions is beneficial for many industries. For instance, 30 minutes ahead prediction of traffic volume in different regions of a city, can provide sufficient time for decision makers to resolve possible issues by rerouting the traffic flow in extreme cases.

There is an extensive amount of research on time-series forecasting and prediction. However, they mostly focus on univariate time-series or low dimensional multi-variate series. In high-dimensional settings, there has been a lot of recent research on problems of next frame prediction in videos. However, these methods do not consider periodic patterns as they are not prevalent in video data. In this problem, based on the intuition that the crowd density data in many regions show periodic patterns over different time-scales, we aim to propose an optimization-based model that can leverage the previous periodic patterns in the data to improve the prediction accuracy. In addition to next state prediction, it is important for many applications to predict multiple steps ahead. Therefore, the prediction model needs to predict a sequence of spatial images corresponding to multiple steps ahead.

1.2.3 Temporal Anomaly Detection for Geo-Spatio-Temporal Data

Large amount of geo-tagged mobility data is collected everyday from mobile devices carried by people or GPS logging devices attached to vehicles. The aggregated number of people or

density of vehicles in different regions may irregularly increase or decrease due to numerous reasons such as accidents, gatherings, events, holidays and festivals, etc. Detection of abnormal patterns in a real-time is an important problem in various domains. For instance, regulating the traffic in the shortest possible time is important for traffic management. Similarly, balancing coverage for a cellular network, when a large number of mobile users gather in one location, is crucial for mobile operators.

Previous work on anomaly detection in spatio-temporal data uses reconstruction loss [KTP18, HCN⁺16], or deviation of low-dimensional representations from normal instances [XRY⁺15] to identify abnormal patterns. They can capture anomalies in the data that cause high reconstruction loss, or high deviation from the normality patterns. We call such anomalies ‘global anomalies’. Global anomalies refer to the rare patterns that are significantly different from normal observed patterns in that space. A sudden unusual rise in the number of people in a highway is an example of global anomaly.

However, we observe that many anomalies in spatio-temporal applications, such as traffic or crowd density monitoring, are similar to normal instances, and thus not global anomalies, but may happen at unusual times. We call such anomalies ‘temporal anomalies’. Observing the traffic pattern from rush hours in a street at midnight is an example of a temporal anomaly. The previous works would not detect temporal anomalies, because they have been observed by the model before. Considering the limitation of the previous works on anomaly detection in geo-spatial domains, we aim to propose an optimization-based model for detecting not only global anomalies but also temporal anomalies in geo-spatial domains.

1.3 Approach and Methodology

In order to understand the changes in temporal data, such as tracking the concepts, leveraging the recurring concepts for prediction and detection of abnormal patterns, a large amount of temporal data is required to build effective machine learning based models. Fortunately, recent advances in hardware technology have made sensors and ubiquitous devices cheaper and more accessible. Therefore, with large amounts of data available, it is possible to train complex models to perform the aforementioned tasks.

In this thesis, we propose innovative machine learning-based approaches to learn accurate models that address the defined problems in Section 1.2. Specifically, for each problem, we define mathematical models in the form of objective functions and optimize them using popular and well-defined methods from the optimization literature. Next, we briefly explain the approach and methodology used to solve the problems defined in section 1.2.

1.3.1 Concept Tracking (problem 1.2.1)

The problem of tracking multiple concepts under unsupervised settings presents several challenges, that although have been addressed separately in the literature, have not been considered together. Thus, in our first study, our objective is to solve the following challenges simultaneously:

- **Multiple-concept, unsupervised problem setting:** This problem setting has received comparatively less attention in existing concept drift and stream mining works. The majority of existing works are either single-concept [QAWZ15], or focus on predicting new labels from existing labeled data [KM⁺03, SAK14]. Our method detects and tracks multiple concepts, which are shared over multiple actors, in an unsupervised fashion.
- **Two-level actor-record data:** In our setting, data is two-level, where each actor can have a different number of records (data samples) per epoch, and where every actor’s concept participation needs to be independently tracked. This two-level structure is not found in unsupervised methods such as Matrix Factorization [GNHS11] and Sparse Coding [OF97], while tensor decomposition approaches [KB09] require every actor to have the same number of records.
- **Scalability:** On larger datasets, effective parallelization strategies are vital to keep execution times reasonable. We apply parallel algorithmic strategies from the optimization literature, such as Stochastic Gradient Descent [Bot10], simplex projections [WCP13] and Adaptive Gradient [DHS11]. We implement these on a bounded-asynchronous distributed Machine Learning framework [XHD⁺15] for faster execution on compute clusters.

To address these challenges, we design an optimization-based method, **ConTrack**, where the desired attributes of the output are modeled as additional terms or constraints — for example, we require that each actor’s concept participations are non-negative and sum up to 1, so that his/her activity patterns can be easily interpreted. ConTrack is designed as a distributed algorithm, and scales to multiple machines to address larger problem settings.

Our method is related to unsupervised scalable algorithms from Machine Learning and Data Mining literature, such as Matrix Factorization [GNHS11] and Topic Models [YGH⁺15], which can decompose input data with N samples into a smaller number of factors or topics respectively. A key difference between our method and Matrix Factorization, is that the latter does not draw a distinction between actors and records, resulting in poor performance in our experiments. While Topic Modeling is able to group multiple records under a single actor, it can only be used on data with discrete-valued features, such as text corpora. Real-world settings often require real-valued features, such as accelerometer data from wearable sensors.

Most concept drift and stream mining algorithms are inapplicable to our setting, because they are supervised or assume only one concept. Conversely, unsupervised learning and clustering methods, such as K-means++ [AV07], Fuzzy Clustering [BEF84], Matrix Factorization [GNHS11], and Sparse Coding [OF97], can be imperfectly applied to our problem setting (given some modifications, in order to cope with multiple users and temporal data). Our evaluation will use the aforementioned methods as baselines.

1.3.2 Next States Prediction (problem 1.2.2)

In this problem, a prediction model needs to accurately predict the next state of a geo-spatial space, given a window of observations. Formally, if $X^{(t)}$ represents the observation at time point t , we would like to learn a function f , to predict the next state $\hat{X}^{(t+1)}$, given a window of size τ of observations up to time point t .

$$\hat{X}^{(t+1)} = f(X^{(t)}, \dots, X^{(t-\tau+1)}; \theta) \quad (1.2)$$

Classic linear time-series forecasting methods such as ARIMA and different variants of this model [BJRL15] are only applicable to stationary time-series where statistical properties of data should remain approximately constant over time. Moreover, finding the right set of parameters for these models is challenging. Seasonal ARIMA (SARIMA) is able to incorporate seasonal parameters into the ARIMA model [WH03], but it becomes ineffective and inapplicable for modeling recurring patterns with long seasonal periods in practice [Hyn10].

Recently, deep learning models have become popular for spatio-temporal prediction problems, due to their inherent ability to model complex non-linearities in data. The emergence of convolutional recurrent networks (CRNs) such as ConvLSTM [XCW⁺15] and ConvGRU [BYPC16] was a breakthrough in spatio-temporal forecasting, due to their ability to capture spatial and temporal correlations in end-to-end trained models. However, these models have not explicitly considered periodic patterns. Previous attempts to do so in deep learning based models for other time-series forecasting applications have been limited; for example, [ZZQ17] additionally used raw input data from previous periods in their model.

In our second study, we propose **Periodic-CRN (PCRN)**, a deep learning based model for crowd density prediction in the spatio-temporal domain. Specifically, our model predicts the next state of a geo-spatial space, given a sequence of previous states. Our model is composed of three main components: 1) Multi-layer pyramidal convolutional recurrent network: to effectively capture spatial and long-term temporal dependencies in the data, 2) Periodic representations: to incorporate recurring periodical patterns into the prediction model and 3) Fusion process: to merge the current representation with periodical representations from previously observed recurring patterns. Our key idea is to use *periodic representations* with tensors, instead of using raw input data from those periods in the learning process, as in previous

work [ZZQ17]. In addition to relieving practitioners from providing redundant input to the model, it lets the model learn effective representations from a *sequence of inputs* in previous periods, as compared to single snapshots from raw inputs.

1.3.3 Temporal Anomaly Detection (problem 1.2.3)

In an anomaly detection problem, given a set of geo-spatial observations $\{X^{(1)}, X^{(2)}, \dots, X^{(T)}\}$ for T discrete time points, the model outputs data points $\{X^{(a_1)}, \dots, X^{(a_n)}\}$, where a_i indicates the anomaly points. In this study, we propose **GSTAD**, a novel end-to-end geo-spatio-temporal anomaly detection model for detecting not only global anomalies, but also temporal anomalies in geo-spatial domains.

The key idea in our approach is detection of temporal anomalies based on recurring periodic patterns in geo-spatio-temporal data. Many time-series data in geo-spatial domains show recurring patterns over different time scales (e.g. daily, weekly). For instance, the patterns of traffic in a specific region in weekday mornings are quite different from the patterns in week-end nights. So, if an observed pattern at a specific time point is drastically different from other historical observations from that time period in the past, but similar to historical observations from other time periods, it is suspected to be a temporal anomaly.

Specifically, given a dataset of crowd density observations (e.g. people, vehicles), we create a sequence of high-dimensional ‘spatial images’, which represent the aggregated crowd density in each region during a time interval of pre-defined length. 3D-convolutional kernels are used as encoders on the sequence of spatial images to capture both temporal and spatial dependencies in the training data. They are also used as decoders to reconstruct the input sequence, and learn representations of normal patterns in the given data, which may help detect global anomalies.

For detecting temporal anomalies, the learned representations are reused in a feed-forward network for prediction of the temporal metadata of the input sequence (e.g. time of day, day of week), where temporal metadata is always available in temporal settings. The feed-forward network are learned together with the 3D-convolutional networks in a single end-to-end model.

Two approaches are proposed for the prediction of temporal metadata: 1) classification-based (GSTAD-Cls) and 2) regression-based (GSTAD-Reg) approaches. The mis-classifications in the classification-based approach and the instances with high mean-squared-error in the regression-based approach might be temporal anomalies.

It is also important for users and practitioners to observe the abnormal patterns in geo-spatial space to get insight on which regions show the most abnormalities at a given moment. To enable this observation, we first collect the mean values of normal patterns in all the regions of the space and, for each abnormal observation, generate a heatmap of most abnormal regions by computing the deviation of observed values from the mean value of normal patterns in historical data.

1.4 Summary of Contributions

In order to better understand and analyze the past and future patterns and events in temporal data, we focus on different types of changes in temporal data and propose three models in this thesis to (1) track multiple concepts based on the changes in distribution of temporal data over time, (2) predict future state by taking leverage of the recurring concepts and (3) detect abnormal patterns in temporal data by identifying the temporal outliers. An overview of the research problems addressed in this dissertation are shown in Figure 1.4. Our contributions in this thesis are summarized as follows:

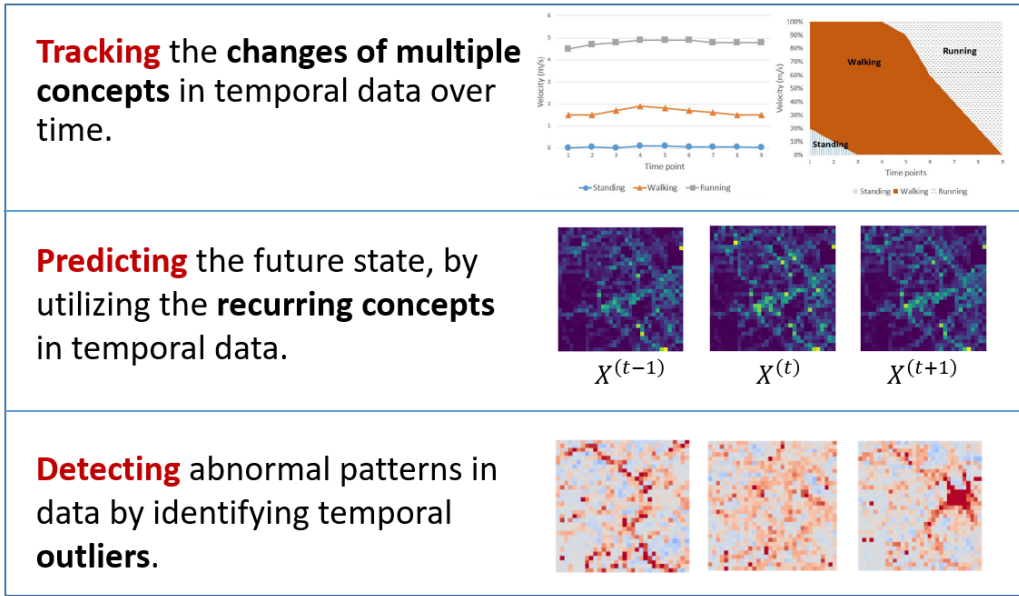


Figure 1.4: Overview of the research problems

Firstly, we propose an optimization based framework called *ConTrack* to track the changes in multiple concepts in unlabeled high-dimensional data. The proposed model, tracks multiple concepts and participation of actors in each concept in multi-dimensional unlabeled data over time. As scalability is a challenge for large-scale data, the optimization engine of *ConTrack* is designed to run in parallel over a cluster of machines and thus is scalable to large-scale multi-dimensional data. Comprehensive experiments over synthetic, semi-synthetic and real data are conducted, and show that *ConTrack* is able to outperform other well-known unsupervised methods in the machine learning and data mining literature.

Secondly, we propose *Periodic-CRN* (PCRN), a deep learning based model for crowd density prediction problem in geo-spatial data. Based on the intuition that many geo-spatio-temporal data show recurring periodic patterns over different time-scales such as days and weeks, we introduce periodic representations, into a spatio-temporal prediction model, and

show their effectiveness in producing more accurate predictions. To the best of our knowledge, this is the first work that makes use of periodic patterns in a convolutional recurrent network architecture, for a spatio-temporal forecasting problem. In addition, we use a pyramidal architecture for multi-layer convolutional recurrent network, for efficient and enhanced capturing of spatial and temporal correlations in spatio-temporal data. Three different methods for updating and reusing periodic representation are proposed and a weighting based fusion mechanism is introduced to weigh the periodic tensors of different scales (e.g. daily, weekly) based on their relevance to the current state. The experiments conducted on two real world taxi trajectory datasets from Singapore and Beijing, show that PCRN outperforms other linear and non-linear methods for one-step and multi-step ahead prediction tasks.

Finally, we propose *GSTAD*, a novel end-to-end deep learning based model for detection of anomaly patterns in geo-spatial domains. By incorporating a temporal meta-data prediction network into a convolutional auto-encoder architecture, the proposed model is able to accurately detect *temporal* anomaly patterns, which could not be captured by the existing reconstruction-based anomaly detection models, as well as *global* anomaly patterns. A novel regression-based approach for temporal metadata prediction is proposed, where unique real valued vectors in Euclidean space are assigned to different time points in weekdays and weekends, in order to recognize recurring patterns over two different time scales (daily, weekly). We show that the regression-based approach captures temporal anomaly patterns better than classification-based approaches. We also show that 3D convolutional networks can capture temporal dependencies as well as spatial dependencies in spatio-temporal data, where the temporal information is represented as the third dimension on top of the 2D of the spatial information, thus improving over 2D convolutional networks that are trained with the same data without the temporal dimension.

Chapter 2

Literature Survey

In this chapter, we review the literature on different problems that we address in this thesis. First, in Section 2.1, some preliminary machine learning related topics that are used throughout the rest of this thesis are briefly introduced. In Section 2.2, the concept drift problem and the methods for handling it in supervised settings are reviewed. In Section 2.3, different unsupervised methods and their extensions to temporal settings are introduced. In Section 2.4, literature survey on predictive models in temporal settings is conducted. Finally in Section 2.5, the existing methods for anomaly detection in temporal settings are introduced.

2.1 Preliminaries

In this section, some preliminary machine learning based methods that are used in the remaining chapters, including optimization using stochastic gradient descent (SGD), neural networks and its subclasses such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are briefly introduced.

2.1.1 Optimization

In most machine learning problems, the historical data is typically used to train models that can perform a certain task. To achieve this goal, an objective function is defined to quantify the goodness of the model. In a supervised learning scenario, an objective function $L(f_\theta(x); y)$ outputs a scalar value that quantifies how good (bad) the function f parameterized by θ is, in approximating target value y , given input x . Different objective functions produce different penalties. Therefore, definition of the objective function is a crucial part of designing a model. In order to find the best set of parameters θ^* for f_θ to approximate y , optimization methods are applied to minimize the objective (loss) function $L(\cdot)$ averaged over the examples as shown in Equation 2.1.

$$\theta^* = \operatorname{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^n L(f_{\theta}(x_i), y_i) \quad (2.1)$$

Mathematical optimization is a vast field of study by itself, which cannot be covered in this review. Here, we focus on *gradient descent* method, as one of the most popular and most common techniques in the optimization literature. Gradient descent is an optimization algorithm to minimize a loss function $L(\theta)$, by iteratively updating model parameters θ in the negative direction of the gradients of the loss function $\nabla_{\theta} L(\theta)$ w.r.t. the model parameters θ . Assuming that loss function $L(\theta)$ is continuous and differentiable at all points, the **batch gradient descent** algorithm [Bot10] is defined as follows:

```

for epochs do
   $\theta_{t+1} = \theta_t - \eta \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} L(f_{\theta}(x_i), y_i)$ 
   $t = t + 1$ 
end for

```

In this algorithm, η is the learning rate, which indicates the step size at each epoch. The above algorithm is referred to as batch gradient descent as the gradients for the whole dataset (batch) needs to be computed at each epoch in order to perform a single update. This makes the algorithm slow and inefficient as the whole dataset needs to be loaded in the memory for gradient computation.

Stochastic gradient descent (SGD) [Bot10] also known as incremental or online gradient descent, is a variation of gradient descent algorithm in which the gradients are calculated for each training example:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(f_{\theta}(x_i), y_i). \quad (2.2)$$

Computationally, stochastic gradient descent is faster than batch gradient descent as they perform gradient update using individual examples. As a single example could be noisy, the updates could be stochastic and fluctuations may occur during convergence, but it has been shown that SGD ultimately converges to the optimal minima if the objective function is convex [GBC16]. **Mini-batch gradient descent** [Bot10] computes the gradient and perform updates on a small batch of examples rather than a single example to diminish the effect of noise. This approach allows the model to have more stable convergence than SGD with single example updates. The update equation of mini-batch gradient descent is shown in Equation 2.3.

$$\theta_{t+1} = \theta_t - \eta \frac{1}{k} \sum_{i=1}^k \nabla_{\theta} L(f_{\theta}(x_i), y_i), \quad (2.3)$$

where, the mini-batch size $k \ll n$, and n is the total number of examples.

2.1.2 Neural Networks

Neural networks [Ros57] are computational systems that their functions are inspired by the biological structure of human brain. These models learn to perform a certain task via observing examples and adjusting a set of internal weights (model parameters) to optimize an objective function. Feed forward neural networks are the most basic type of neural networks which are structured in layered configuration consisting of input, hidden and output layers, where layers are interconnected to previous and next layers. The artificial neurons in different layers can send signals to other connected neurons and apply a non-linear function on their input. General architecture of a simple feed forward neural networks is illustrated in Figure 2.1.

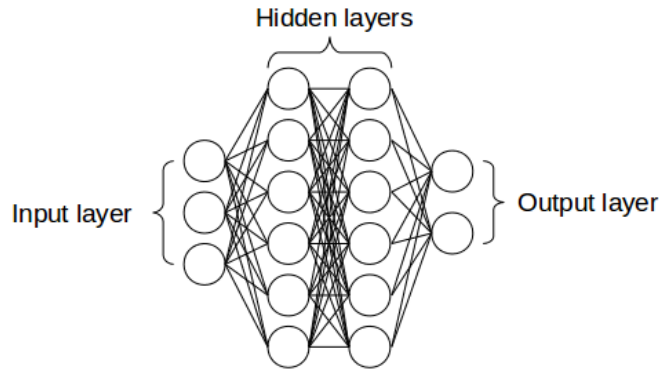


Figure 2.1: A simple feed forward neural network architecture with one input layer, two hidden layers and one output layer.

Formally, for a feed-forward neural network with L layers, $L-1$ weights matrices $\{W^2, \dots, W^L\}$ and $L-1$ bias parameters $\{b^2, \dots, b^L\}$ are defined, which transform input feature vector $x \in \mathbb{R}^{d_i}$ into output vector $a^L \in \mathbb{R}^{d_o}$ in the output layer, using the following procedure.

```

initialize:  $a^1 \leftarrow x$ 
for  $l = 2$  to  $L$  do
   $z^l \leftarrow W^l \cdot a^{l-1} + b_l$ 
   $a^l \leftarrow \sigma(z^l)$ 
end for

```

The activation function σ is a differentiable non-linear function, applied on the weighted sum of the inputs of each neuron. Tangent hyperbolic function $\tanh$, sigmoid logistic function $s(x) = 1/(1 + \exp(-x))$ and rectified linear unit function $\text{relu}(x) = \max(0, x)$ are common choices for activation functions.¹

¹Although relu function is non-differentiable at a single point $x = 0$, sub-gradient descent can be used in the optimization process.

2.1.2.1 Back-Propagation Method

Back-propagation is an optimization algorithm based on gradient descent method, that is widely used for training neural networks. The optimization process in neural networks consists of two stages 1) forward pass and 2) parameter updates.

In the forward pass, the input provided in the first layer is propagated forward using the aforementioned feed forward procedure, until it reaches the output layer. The cost is then calculated using the defined loss function and the gradients of the loss function with respect to the output layer activities are computed. Starting from the last layer in a backward manner, the gradients of the loss function w.r.t. the inputs of next layer, weight parameters between two layers and the activities of the previous layer are computed. In computing the gradients of each layer, the gradients calculated for the next layer are reused in the previous layer, through chain multiplication of derivatives. This property allows the model to compute the gradients more efficiently compared to the approach of calculating gradients separately. The pseudocode of the back-propagation algorithm is shown in Algorithm 1.

Algorithm 1: Back-propagation algorithm

```

1: initialize:  $a^1 \leftarrow x$ 
2: for  $l = 2$  to  $L$  do {FeedForward}
3:    $z^l = W^l \cdot a^{l-1} + b_l$ 
4:    $a^l = \sigma(z^l)$ 
5: end for
6:  $\delta^L = \nabla_a C \circ \sigma'(z^L)$  {Output error}
7: for  $l = L - 1$  to  $2$  do {Back-propagation}
8:    $\delta^l = ((W^{l+1})^T \delta^{l+1}) \circ \sigma'(z^l)$ 
9: end for
10:  $\frac{\partial C}{\partial W_{jk}^l} = a_k^{l-1} \delta_j^l$  {Gradients w.r.t weights}
11:  $\frac{\partial C}{\partial b_j^l} = \delta_j^l$  {Gradients w.r.t biases}

```

In Algorithm 1, δ^l represents the error vector for layer l and the symbol $\circ$ refers to Hadamard (element-wise) product. After computing the gradients of the cost function w.r.t. the weights and biases, the parameters are updated using gradient descent method (Equation 2.2).

2.1.3 Convolutional Neural Networks

Convolutional neural networks are sub-class of neural networks that are widely used for image processing and feature extraction from visual signals in computer vision research. The pioneering work by [LBBH98] laid the foundation for gradient-based learning using convolutional neural networks. Since then, due to their inherent ability in feature extraction and modeling dependencies, they have been widely used in many applications and domains including

image classification [KSH12], natural language processing (NLP) [CW08] and recommender systems [VdODS13]. In this section, we briefly overview the fundamentals of convolutional neural networks. What we are most interested in, is 3D convolutional neural networks which are effective in modeling both spatial and temporal dependencies in spatio-temporal data.

Convolutional neural networks consist of one or multiple convolutional layers followed by a non-linearity function. A convolutional layer consists of one or multiple rectangular-shaped weight matrices, known as convolutional kernels (filters) which convolve over the input matrix and generate feature maps. Assuming we have an input matrix $X \in \mathbb{R}^{N \times N}$ and a convolutional kernel $g \in \mathbb{R}^{M \times M}$, the convolution operation to get the value of feature c at point $[i, j]$ is defined as follows:

$$c[i, j] = g[i, j] * X[i, j] = \sum_{\tau_1} \sum_{\tau_2} g[\tau_1, \tau_2] \cdot X[i - \tau_1, j - \tau_2]. \quad (2.4)$$

In Equation 2.4, $*$ symbol denotes the convolution operation. In 3D convolutional networks, the kernels are 3 dimensional instead of 2 dimensions. This allows the model to extract features along the temporal dimension, when the input is volumetric. We review the literature on 3D convolutional neural networks in more detail in section 2.4.2.

The Equation 2.4 can be extended to 3D convolutional operation using 3D kernel $g \in \mathbb{R}^{M \times M \times M}$ as follows:

$$c[i, j, k] = g[i, j, k] * X[i, j, k] = \sum_{\tau_1} \sum_{\tau_2} \sum_{\tau_3} k[\tau_1, \tau_2, \tau_3] \cdot X[i - \tau_1, j - \tau_2, k - \tau_3]. \quad (2.5)$$

2.1.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a sub-class of artificial neural networks, that are widely used for capturing temporal dependencies in sequences. By retaining a state, these models are able to preserve the previously observed information in a sequence. They have been successfully used in different applications, such as language modeling [MKB⁺10], machine translation [WSC⁺16], speech recognition [GMH13] and caption generation [XBK⁺15]. In this section, we provide a brief overview of RNNs and its variants such as Long Short Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). Finally, the encoder-decoder network architecture and the attention mechanism are briefly introduced.

2.1.4.1 Vanilla RNN Architecture

Vanilla RNN is the basic form of recurrent neural network. The input in RNN is a sequence $x^{(t)}$ where t refers to a discrete time point. RNN has a memory cell referred to as hidden state

unit $h^{(t)}$ which keeps the information that RNN has preserved up to each time point. Hidden state at each time point is calculated by non-linear combination of the input at that time point and the hidden state at the previous time point. The hidden state $h^{(t)}$ and the output $o^{(t)}$ at each time point is calculated as follows:

$$\begin{aligned} h^{(t)} &= \sigma(Ux^{(t)} + Wh^{(t-1)} + b_h), \\ o^{(t)} &= \text{softmax}(Vh^{(t)} + b_o). \end{aligned} \quad (2.6)$$

The function σ in equation 2.6 refers to a non-linear function. U, W and V are the weight parameters and b_h, b_o are the bias parameters that the model learns. A schematic diagram of an unfolded vanilla RNN is illustrated in Figure 2.2.

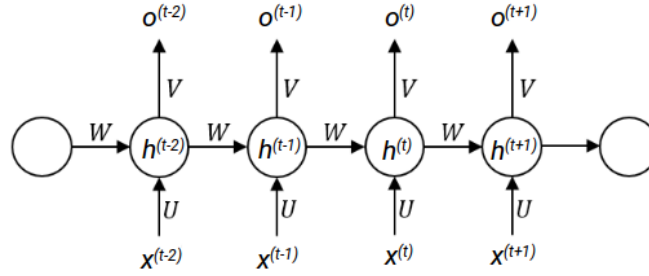


Figure 2.2: An unfolded recurrent neural network architecture.

Despite its success in modeling short sequences, the vanilla RNN architecture suffers from a problem known as ‘vanishing gradient problem’. To update the model parameters, vanilla RNN uses Back-Propagation Through Time (BPTT) method [Wer90], which contains long series of derivatives multiplications for a sequence of RNN nodes. The long multiplication chain, causes fast shrinking of the gradients when the neurons become saturated, which causes little or no change in the value of parameters. This means that, the network will not be able to take the further dependencies into account. To address this issue, different variants of RNN such as Long Short-Term Memory (LSTM) units [HS97] and Gated Recurrent Units (GRU) [CVMG⁺14] are proposed. In the following subsection these architectures are introduced.

2.1.4.2 Long Short-Term Memory (LSTM) Architecture

Long Short-Term Memory (LSTM) unit was originally introduced in [HS97] to address the vanishing gradient problem by introducing multiple switch gates and a cell memory state, which lets the gradients to bypass the units and prevent them from shrinking to zero. Different gates, control what information should be remembered and what information needs to be forgotten. The equations for LSTM gates, cell state and its output are as follows:

$$\begin{aligned}
 f^{(t)} &= \sigma(W_f \cdot h^{(t-1)} + U_f \cdot X^{(t)} + b_f), \\
 i^{(t)} &= \sigma(W_i \cdot h^{(t-1)} + U_i \cdot X^{(t)} + b_i), \\
 o^{(t)} &= \sigma(W_o \cdot h^{(t-1)} + U_o \cdot X^{(t)} + b_o), \\
 \tilde{c}^{(t)} &= \tanh(W_h \cdot h^{(t-1)} + U_h \cdot X^{(t)} + b_h), \\
 c^{(t)} &= f^{(t)} \circ h^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}, \\
 h^{(t)} &= o^{(t)} \circ \tanh(c^{(t)}).
 \end{aligned} \tag{2.7}$$

In Equation 2.7, $f^{(t)}$, $i^{(t)}$ and $o^{(t)}$, represent the forget gate, input gate and output gate respectively. Forget gate $f^{(t)}$, is a switch that controls what information are thrown away at time point t . The input gate $i^{(t)}$, controls what information are read at time point t and $o^{(t)}$ is a switch that controls what information are written in output. $\tilde{c}^{(t)}$ is the candidate state vector and $c^{(t)}$ is the cell state vector. The output vector is represented by $h^{(t)}$.

2.1.4.3 Gated Recurrent Unit (GRU) Architecture

Gated Recurrent Unit (GRU) is another variant of RNN introduced in [CVMG⁺14]. Compared to LSTMs, they have more simplified structures with fewer number of gates (forget gate and update gate). The cell state $c^{(t)}$ is also merged with hidden state $h^{(t)}$. The equations for GRU components are as follows:

$$\begin{aligned}
 z^{(t)} &= \sigma(W_z \cdot h^{(t-1)} + U_z \cdot X^{(t)} + b_z), \\
 r^{(t)} &= \sigma(W_r \cdot h^{(t-1)} + U_r \cdot X^{(t)} + b_r), \\
 \tilde{h}^{(t)} &= \tanh(W_h \cdot (r^{(t)} \circ h^{(t-1)}) + U_h \cdot X^{(t)} + b_h), \\
 h^{(t)} &= (1 - z^{(t)}) \circ h^{(t-1)} + z^{(t)} \circ \tilde{h}^{(t)}.
 \end{aligned} \tag{2.8}$$

In Equation 2.8, $r^{(t)}$ and $z^{(t)}$ are the update gate and forget gate respectively. Moreover, $\tilde{h}^{(t)}$ is the candidate state vector and $h^{(t)}$ is the hidden state vector.

GRU architecture has fewer number of parameters than LSTM which makes it easier to train. However, the empirical studies on comparison of these two architectures show that their performance are not significantly different [CGCB14].

In addition to the above-mentioned variants of RNN, many other recurrent-based neural network architectures such as bidirectional RNNs [SP97], hierarchical RNNs [EHB96], Gated feedback RNNs [CGCB15] and hierarchical multi-scale RNNs [CAB16] have been proposed. For more detailed review of different RNN architectures, interested readers can refer to [LBE15, Sut13].

2.1.4.4 Encoder-Decoder Networks and Attention Mechanism

Deep recurrent neural networks and its variants have shown to be extremely powerful in sequence modeling and temporal applications such as language modeling [MKB⁺10] and speech recognition [GMH13]. Emergence of encoder-decoder networks and sequence-to-sequence models [CVMG⁺14, SVL14] made a breakthrough in many applications such as neural machine translation [WSC⁺16]. These networks are composed of two separate recurrent neural networks, one for encoding a sequence into a dense representation vector (thought vector) and another for decoding the learned representation into an output.

In the encoder-decoder networks, the last hidden state vector of the encoder is copied into the decoder network. Therefore, all the information in the sequence is encoded in the last hidden state vector. This could be problematic in long sequences, as the encoder RNN might lose some information from earlier parts of the sequence. The attention mechanism introduced in [BCB14], addressed this problem by selectively attending different parts of the input sequence during the decoding process. This approach has been used in many problems such as image captioning [XBK⁺15] using visual attention on images, video captioning [VXD⁺14], document classification [YYD⁺16] and speech recognition [CJLV16].

Given a context vector c (hidden state vector at a decoder cell) and a set of hidden state vectors outputted from encoder cells $\{h_1, h_2, \dots, h_n\}$, the goal of the attention model is to output a single vector z , which is the weighted summation of hidden state vectors considering the context c . In the simplest form, the attention mechanism is defined as follows:

$$e_i = \tanh(W.c + U.h_i), \quad (2.9)$$

$$a_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}, \quad (2.10)$$

$$z = \sum_i a_i \cdot h_i. \quad (2.11)$$

In the first step, context vector c is non-linearly combined with each hidden state vector to compute the relevance and similarity of each vector with the current context (Equation 2.9). To combine context vector with hidden state vectors, other approaches such as dot product or linear combination can also be used. In the next step softmax function is applied on the output vectors e_i so that $0 \leq a_i \leq 1$ and $\sum a_i = 1$ (Equation 2.10). Finally, the weighted sum of the hidden state vectors considering the learned weights a_i is assigned to z (Equation 2.11).

2.2 Concept Drift in Temporal Data

After defining the preliminary methods used throughout this thesis, we are now ready to elaborate the related work for the three key problems addressed in this thesis. In this subsection,

we introduce concept drift problem and review different methods for handling this issue in supervised settings.

Today, in many applications and domains, the data is generated in the streaming format. Data generated from different types of sensors in industrial environments, financial transactions generated in the banking domain and videos captured from surveillance cameras are just some example of wide variety of data streams. Dynamic nature of data is an important factor in the data streams that distinguishes them from static data. Unlike the simplistic assumption that data streams are always generated from a stationary data generation process, in the real world scenarios, the underlying distribution of data in temporal domains are changing over time and making them non-stationary.

2.2.1 What is Concept Drift

The term ‘concept’ in concept drift, refers to one or more hidden factors that govern the data generation process. For instance, the concept of seasons in weather data, affects features such as temperature, humidity, precipitation rate, etc. As another example, the concept of user interest would affect the news pages that are visited in a news website by the user. Changes in such concepts lead to a phenomenon known as ‘concept drift’.

Concept drift is a challenging issue especially in supervised settings as the changes in underlying distribution of data can make the trained models inaccurate or completely obsolete over time. Several surveys have focused on the concept drift problem and provided an overview of different techniques to address this issue [GZB⁺14, Tsy04, DRAP15]. [Tsy04] stated that an effective concept drift handling technique should fulfill three criteria: 1) It should adapt to the changes quickly, 2) it should be robust to noises and 3) it should consider recurring contexts. Furthermore, it categorized types of drift into *sudden* and *gradual* concept drift.

Let $p_t(X)$ be the underlying data distribution of data at time point t . In supervised settings, [GZB⁺14] defined concept drift as the change in joint probability distribution of feature vector X and target variable y between two time points t_0 and t_1 :

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y). \quad (2.12)$$

A probabilistic approach to detect such concept drifts, is to model and compare posterior probabilities $p(c | X) = \frac{P(c)p(X|C)}{p(X)}$ at different time points. Any changes in the prior probability of classes $p(c)$, the distribution of any of the classes $p(X | c)$, or changes in posterior probability of classes $p(c | X)$ over a period of time could cause changes in joint probability distribution and result in concept drift [KHA99, GFHP07]. However, calculating these probabilities is not straight-forward in many problems.

[Tsy04] categorized concept drift into *Real drift* and *Virtual drift*. Changes in marginal distribution $p(X)$ is defined as virtual drift while changes in posterior distribution $p(y | X)$ is referred to as real drift. However, both types of drift are viewed as concept drift in practice and the challenge is to learn a model that can handle the concept drift problem effectively.

2.2.2 Concept Drift Handling Methods

The problem of concept drift in supervised settings has been the subject of research over the past two decades in data stream mining field and different methods have been proposed to handle the challenge that concept drift imposes on supervised machine learning models. [vZI10] categorized concept drift handling methods into two main classes of trigger-based and evolving methods. [DRAP15] referred to these two categories as *active* and *passive* methods respectively.

Trigger-based or active methods handle drift by detecting the change in the underlying distribution of data and updating the model afterwards. Evolving or passive methods, update the model continuously using the latest arrived data and adapt the model to the changes. Active methods work better when the drift is abrupt or sudden and passive methods are better at handling gradual or incremental drift [DRAP15]. In the following subsections, we briefly summarize the active and passive methods for handling drifts.

2.2.2.1 Active Methods

Active concept drift detection algorithms work by continuously running a change detector on the data to test whether there is a significant change in the underlying data distribution. This detector can be integrated into a model, so that every time the detector signals change, the algorithm updates the model. Therefore, *change detection* methods can play an important role in this category.

Change Detection Methods: [DKVY06] proposed a non-parametric approach based on *kdt-tree* data structure to estimate the data distributions using bootstrapping method and measuring their difference using Kullback-Leiber (KL) divergence metric. [KF⁺12, QAWZ15] used a PCA based method for density estimation of multi-dimensional data for change detection. [SWJR07] proposed ‘density test’ which is a statistical methods to check whether the current data in the test window are sampled from the same distribution as in reference window, while [dRFMdAPAB16] used incremental Kolmogorov-Smirnov test for the same task. [BM⁺08] paired a reactive learner which predicted based on the recently available data, and a stable online learner which used the whole history of data, and compared their accuracy to detect the change in underlying data distribution.

[BG07] proposed ‘Adaptive Windows’ (ADWIN) to detect the changes in univariate data streams. Their model used sliding window technique over the stream of data. Whenever two large enough sub-windows show significantly large differences in their mean value, a change signal is generated, the old sub-window is discarded and the recent values are kept. In another study, [VB06] used the entropy measure to signal the change.

Monitoring Classifier Accuracy: Another approach to detect the changes in underlying data distribution in classification problems, is to monitor the output of a learner and trigger a change signal when the error-rate of the algorithm passes a certain threshold. Drift Detection Method

(DDM) [GMCR04] and Early Drift Detection Method (EDDM) [BGdCÁF⁺06] followed this strategy. DDM considered the expected accuracy of the learner. After observing the true labels, the prediction outcome would be a series of true and false values which could be defined as Bernoulli trials. By monitoring these values, the probability of error is calculated from the binomial distribution and if it passes a certain threshold, the system signals warning or drift states. In the warning state, the data is stored to re-create the model and in the case of drift, a new model replaces the last model. EDDM improved DDM for gradual drift cases by considering the distance between classification errors. [HKB16] also proposed a semi-supervised approach for change detection by computing the classifier confidence in classifying the data.

The statistical change detection methods are mostly applicable on univariate data, and detection of changes in multivariate data is still an open research problem. Moreover, the trigger-based methods that monitor the classifiers accuracy, rely on immediate availability of the ground-truth for drift detection. However, in practice, the labels are provided with arbitrary amount of delay.

2.2.2.2 Passive Methods

In contrast to active techniques, passive methods or ‘evolving methods’ update their model of the data continuously, in order to adapt to the changes. [Tsy04] classified the passive methods into three main classes: 1) instance selection methods 2) instance weighting methods and 3) ensemble methods.

Instance selection methods, select and use a portion of data to update the model. The FLORA family of algorithms proposed by [WK96] is one of the pioneering works which used a fixed sliding window to learn from the most recent data. Other studies such as [KW95] and [Sal97] also used window-based methods to update their model. There are also tree-based online algorithms that are able to cope with drift problem by incrementally updating their model as data stream arrives [DH00].

In instance weighting methods, instances are weighted based on different criteria such as age and consistency with the current model to reduce the effect of drifting concept. [Kli04] used such approach to handle the drift. SyncStream [SAK14] and IBLStream [SH12] used a prototype based approach. In this approach, nearby points are summarized into ‘prototypes’ using constrained clustering, in order to compactly represent the data distribution. Each prototype is given a weight that can increase or diminish with time, as the data distribution changes. The SyncStream approach tested for concept drift via principal component analysis (PCA) or statistical analysis.

Ensemble methods are one of the most frequently used evolving techniques in handling concept drift. In this technique, instead of one classifier, an ensemble of classifiers decide on the output of a prediction. Each model becomes an expert on a different distribution in the data (using a separate chunk of data as input), and with proper weight management, the combined prediction can produce a more accurate result.

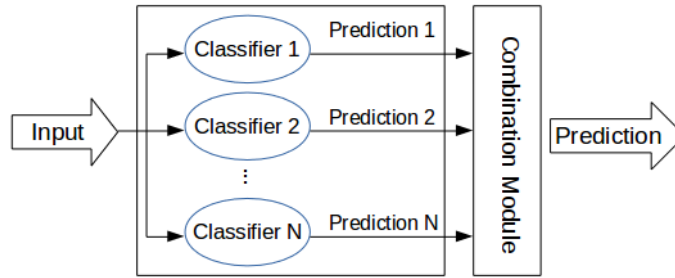


Figure 2.3: Architecture of general ensemble learning model

SEA [SK01] was one of the first ensemble techniques to handle drifts. It builds multiple classifiers on different chunks of data and in each round of prediction, the worst performing classifier is discarded. The output is predicted based on majority voting. ‘Dynamic Weighted Majority’ (DWM) proposed by [KM⁺03] is another ensemble-based technique which dynamically creates and removes classifiers. It has a weighting mechanism which decreases the weight of classifiers with lower accuracy. If the final weighted prediction is wrong, a new classifier gets added to the system. [WFYH03] proposed an Accuracy Weighted Ensemble (AWE) model and also a pruning algorithm in order to use a subset of classifiers for prediction. They also mathematically proved that an ensemble on last K chunks of data have better accuracy than a single classifier built on all K chunks. [ZZS08] categorized drifts into loose concept drifts (LCDs) and rigorous concept drifts (RCDs). For LCDs they proposed an instance weighting mechanism called kernel mean matching (KMM) and for RCD cases they proposed a classifier weighting method called Optimal Weight Adjustment (OWA).

2.3 Unsupervised Methods for Temporal Data

In most studies on concept drift problem in temporal settings, research has been mostly focused on supervised scenarios, where the concept drift causes decline in accuracy of models. Nevertheless, changes in data distribution also occur in unsupervised settings where the labels are not available. In this section, we give an overview of the unsupervised methods that are relevant to the setting that we are investigating for concept tracking problem, i.e. the temporal aspect, concept tracking using decomposition or the methods that consider several actors in their settings.

First, we review different clustering techniques that operate on temporal and time-series data. These studies are mostly categorized under ‘Data Stream Clustering’ category. The ‘Evolutionary Clustering’ methods address the cases where the data distribution is changing over time. Next, we review some other popular unsupervised methods that are related to the setting that we are investigating, such as Matrix Factorization, Sparse Coding and Dynamic

Topic Models. In addition to explaining the original methods, we review extension of these models for addressing the settings with multiple users/records and multiple concepts that are presented in the literature.

2.3.1 Evolutionary Clustering

Clustering data streams is a relatively old field in data stream mining. In the streaming settings, the clustering process has to be performed in an online fashion by scanning data only once [ZCQJ08]. [AHWY03] proposed a clustering algorithm on evolving data streams, called ‘CluStream’. It decomposed the process of clustering into an online and an offline components. The online component kept summary of statistical properties of data in form of microclusters, while the other component used those statistical information to perform clustering. Same authors proposed ‘HPStream’, as an extension to their clustering algorithm framework for high dimensional data [AHWY04].

[ZCQJ08] highlighted an excessive memory usage problem in CluStream when sliding windows is used, as this method needs to store the previous snapshots in their model. Moreover, they identified that the expired records affect CluStream results, while this effect needs to be eliminated in the sliding window based techniques. To address these problems, they presented ‘SWClustering’ framework. In order to better handle the outliers, [CT07] proposed a density based clustering framework called ‘D-Stream’. In this framework an online component mapped data point into grids in real-time and an offline component performed density based clustering on this grid and output the clusters. In a separate study, [CEQZ06] proposed a framework called ‘DenStream’. They introduced a core-micro-cluster structure to summarize the clusters as they evolve.

The evolutionary clustering methods are capable of clustering evolving data streams. However, they are mostly applicable to univariate or low-dimensional multivariate data. Increasing the number of dimensions adversely affects the clustering accuracy. Moreover, these methods do not consider settings with multiple actors, where each actor may generate multiple records. Therefore, they can not be used for tracking participation of actors in different concepts. Scalability to large amount of data is another limitation for evolutionary clustering methods, as distributing the data and parallelizing the clustering process is challenging.

2.3.2 Matrix Factorization based Methods

Matrix Factorization [GNHS11] is an unsupervised mathematical model in linear algebra. In this model input matrix $R_{N \times D}$ is decomposed into two matrices $P_{N \times K}$ and $Q_{D \times K}$ such that $P * Q' \approx R$. Matrix Factorization is a popular method that is used in many applications such as recommender systems [KBV09], where the large and sparse matrix of users and items are factorized into two smaller matrices by finding the latent features that determine how users rate

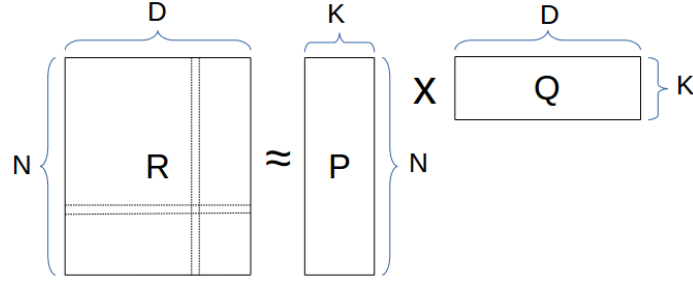


Figure 2.4: Matrix Factorization method

items. Decomposition of the large user-item matrix into two smaller constituents, enables the practitioner to approximate rating of each item for an individual user easier and recommend the high ranked items to the user.

Non-negative matrix factorization (NMF) [LS99] is a Matrix Factorization based method with extra non-negativity constraint, i.e. neither the original matrix nor any of the factorized matrices are allowed to have negative values. In domains such as audio signal processing and computer vision where the non-negativity of elements is an inherent part of the data, NMF is widely used. The decomposition algorithm used in this method is based on multiplicative update rules. Other methods such as alternating least square and projected gradient methods [Lin07] have been proposed for matrix factorization with non-negativity constraints.

Modified versions of MF and NMF have been used in different domain where the data is temporal. In these methods, the goal is to decompose an input matrix on sequential time points and track the changes. [Vir07] and [NLRK⁺10] used NMF based models in audio signal processing domain. [Vir07] proposed a NMF-based approach to separate sound sources in audio signals by decomposing them into basis functions and time-varying gain matrix. Multiplicative update rules were used to find the optimum solution.

In another work [NLRK⁺10] proposed a model based on NMF for sparse representation of music signals and learning time-varying bases. The bases are spectral patterns that have state transition during music composition. NMF with temporal features has also been used in biomedical and neurophysiological domains. In order to derive interpretable results, [CC05] proposed a modified version of NMF by adding constraints to their objective function to guarantee temporal smoothness when the input data is temporal.

2.3.3 Sparse Coding based Methods

Sparse Coding [OF97] is an unsupervised method similar to Matrix Factorization in that it decomposes an input matrix $R_{N \times D}$ into two smaller matrices (dictionary matrix $Q_{D \times K}$, and coefficient matrix $P_{N \times K}$). However, Sparse Coding requires the generated coefficient matrix P

to be sparse. The sparsity is achieved by applying L_1 norm on the coefficient matrix P . The objective of Sparse Coding is to minimize function f such that:

$$f(R, P, Q; \lambda) = \|R - PQ'\|_2^2 + \lambda \|P\|_1. \quad (2.13)$$

[MBPS10] proposed an online optimization algorithm for sparse coding based on stochastic approximation, which can scale up to large datasets with millions of records. Online nature of this algorithm allows the model to process dynamic data such as video sequences one at a time and in mini-batches.

Sparse coding has also been used for sound source separation. [Vir03] presented a sparse coding based approach to separate meaningful sound sources from music signals. The gain of basis functions vary over time and temporal continuity penalty function prevents the gains to show dramatic changes in consecutive time points. [ZFFX11] applied their sparse coding based approach on dynamic video sequences in order to automatically detect unusual events. In their completely unsupervised approach, sparse coding bases are learned from dynamic video signals. Based on the intuition that usual events are easier to reconstruct than unusual ones, the unusual events are detected. By dynamically updating the sparse coding bases, this model also handles the problem of concept drift that may occur in video data.

Matrix Factorization and Sparse Coding based models are able to decompose input data with N samples into a smaller number of factors or topics respectively. Moreover, they can be parallelized to run on a cluster of machines and are suitable for handling large-scale data. However, these methods are not suitable for settings where multiple actors generate multiple records over time, as they do not draw a distinction between actors and records in their model structure.

2.3.4 Dynamic Topic Models

Probabilistic topic models are a large family of statistical methods. These models are mostly used to analyze massive collection of text based documents and discover the hidden themes (known as topics) in documents [Ble12]. Latent Dirichlet Allocation (LDA) [BNJ03] is a generative model that assumes each document is a distribution over multiple topics and topics are distributions over a fixed set of words. The graphical model for LDA is illustrated in Figure 2.5.

In Figure 2.5, the words $w_{d,n}$ are observed. and topics β_k , topic assignments $z_{d,n}$ and topic proportions θ_d are unobserved and unshaded. α and η are proportions parameter and topic parameter respectively, that control the sparsity of each variable. The generative process of LDA is defined as the following joint distribution:

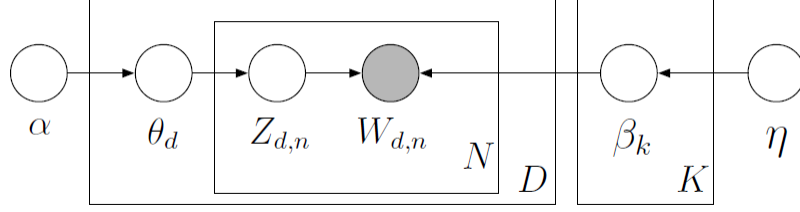


Figure 2.5: Graphical representation of LDA

$$p(\beta_{1:k}, \theta_{1:D}, z_{1:D}, w_{1:D}; \alpha, \eta) = \prod_{i=1}^K p(\beta_i | \eta) \prod_{d=1}^D p(\theta_d | \alpha) \left(\prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right). \quad (2.14)$$

As shown in Figure 2.5 there are a number of dependencies introduced in the joint distribution. For instance the observed word $w_{d,n}$ depends on all of the topics $\beta_{1:K}$ and topic assignments $z_{d,n}$. Topic assignment $z_{d,n}$ is also dependent on topic proportions θ_d .

Although the application of topic models is for text-based documents and the setting that we investigate is non-textual, we realized that the idea behind the topic models which is extracting topics and also topic assignment for observed words is inspiring for our works. More specifically in our setting, each actor (user) has a collection of records at each time point where each of the records could be generated from a different concept just like the words in topic models can be generated from different topics.

More importantly we are interested in an extension of basic topic models, called dynamic topic models [BL06] where temporal data is analyzed to discover the evolution of different topics over time. In this model the topics $\beta_{t,k}$ and proportion parameter α_t evolve over time. As shown in Equation 2.15, word distribution of topic k at time t , $(\beta_{t,k})$ is generated from $\beta_{t-1,k}$ and α_t is generated from α_{t-1} .

$$\begin{aligned} \beta_{t,k} | \beta_{t-1,k} &\sim \mathcal{N}(\beta_{t-1,k}, \sigma^2 I). \\ \alpha_t | \alpha_{t-1} &\sim \mathcal{N}(\alpha_{t-1}, \delta^2 I). \end{aligned} \quad (2.15)$$

While Topic Modeling is able to group multiple records (words) under a single actor (document), it can only be used on data with discrete-valued features, such as text corpora, which is not suitable for settings with real-valued features.

To summarize, most concept drift and stream mining algorithms are inapplicable to our concept tracking problem setting, because they are supervised or assume only one concept.

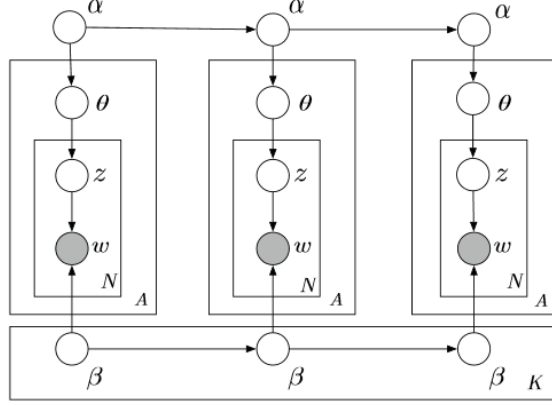


Figure 2.6: Graphical representation of dynamic topic models (for 3 time points), obtained from [BL06]

Conversely, unsupervised learning and clustering methods, such as Matrix Factorization [GNHS11], Sparse Coding [OF97] and some clustering-based methods such as K-means++ [AV07] and Fuzzy Clustering [BEF84], can be imperfectly applied to our problem setting (given some modifications, in order to cope with multiple users and temporal data). We consider using the aforementioned methods as baselines for evaluating our concept tracking model.

2.4 Predictive Models for Temporal Data

Prediction is an important problem in various fields such as machine learning and artificial intelligence research. Given historical data $\{(X_1, y_1), (X_2, y_2), \dots\}$ where X represents a feature vector and y represents a target random variable, the goal of predictive models is to forecast the outcome $\hat{y}$, given feature vector $\hat{X}$. Although prediction term is mostly applicable in temporal settings to forecast future outcomes, but they are not necessarily limited to predicting events in the future. In that sense, regression and classification models are inherently predictive models. However, we focus on reviewing the literature on predictive models for temporal data in this study.

2.4.1 Time-Series Forecasting

Time-series is a sequence of data points, organized based on time order. Time-series data are generated in various fields and domains such as engineering, economics, geology, network communication, etc. Therefore, forecasting future values of a time-series by observing the previous values has tremendous applications in different domains.

The target values for prediction in time-series forecasting is the future values of the time-series itself. Therefore, given a sequence of observed values $\{X^{(1)}, X^{(2)}, \dots, X^{(t)}\}$, the goal of

the forecasting model is to predict values of $\{X^{(t+1)}, \dots, X^{(t+\tau)}\}$ where τ is an arbitrary horizon in the future. Generally, time-series forecasting methods are categorized into two classes of 1) statistical methods and 2) machine learning based methods [LBF13].

2.4.1.1 Statistical Methods

The pioneering work of [BJ76] became the basis for emergence of family of AutoRegressive Moving Average (ARMA) models as the most popular method for statistical linear forecasting. This model is composed of two main components: 1) Autoegressive (AR) part and 2) Moving Average (MA) part. The autoregression process (AR), models the dependency of observations with some number of observations in the past. Formally in $AR(p)$ model, where p is a user-defined number of previous steps that are considered for regression of next value, the value of $X^{(t)}$ is predicted as follows:

$$X^{(t)} = \sum_{i=1}^p \phi_i \cdot X^{(t-i)} + \epsilon^{(t)}. \quad (2.16)$$

where ϕ_i is the coefficient for $X^{(t-i)}$ and $\epsilon^{(t)}$ is the residual at time point t . The Moving Average (MA) process, models the dependency between observation and the residuals in the previous steps. The $MA(q)$ model, where q is a user-defined number of previous steps, predicts $X^{(t)}$ as follows:

$$X^{(t)} = \sum_{i=1}^q \theta_i \cdot \epsilon^{(t-i)} + \epsilon^{(t)}. \quad (2.17)$$

Here, θ_i is the coefficient for the residual $\epsilon^{(t-i)}$ which is learned by the model. Combination of $AR(p)$ and $MA(q)$ would result in a more robust $ARMA(p,q)$ model, which is defined as follows:

$$X^{(t)} = \sum_{i=1}^p \phi_i \cdot X^{(t-i)} + \sum_{i=1}^q \theta_i \cdot \epsilon^{(t-i)} + \epsilon^{(t)}. \quad (2.18)$$

In spite of its popularity in practice, ARMA models has the limitation that they can only be applied on stationary data, where the statistical properties of data is not changing over time. Therefore, techniques such as differencing is used to convert non-stationary time-series into stationary and to remove patterns such as trend and seasonality from the data. Autoregressive Integrated Moving Average (ARIMA) model is a variant of ARMA where differencing is applied on the data. Another limitation of ARIMA models is the need for explicit setting of parameters for the number of lags for autoregressive, moving average and differencing components (p,q,d) . This manual parameter setting requires careful tunings and optimizations.

Over the years, ARIMA has been extended to many other settings. Seasonal ARIMA (SARIMA) is an extension of ARIMA which captures the recurring periodic patterns in the data using additional parameters for more accurate forecasting. SARIMA model has been used in various applications such as traffic and electricity demand forecasting [WH03, Tay03]. However, such methods are applicable on univariate time-series data.

Vector Autoregression (VAR) and Vector ARMA (VARMA) are extensions of autoregressive and ARMA models for multivariate time-series data. In such data, the dependency of observations to multiple dimensions are captured through definition of extra parameters in the model. VARMA has also been used for applications such as traffic flow forecasting [KP03]. However, VARMA is suitable for low dimensional data, and increasing the number of dimensions in this model increases the parameter space exponentially which affects its performance adversely.

2.4.1.2 Machine Learning based Methods

Different machine learning based models have been proposed for time-series prediction problem. [LBF13] refers to artificial neural networks and support vector regression (SVR) models as two main approaches for the machine learning based forecasting task.

Support vector regression (SVR) uses the same principals as SVMs to maximize the margins between support vectors and the separating hyper-plane. [WHL04] adapted support vector regression with RBF kernel for travel time prediction problem. An extension of SVR for on-line settings (OL-SVR) proposed in [CNJJH09] for traffic flow prediction problem. [Lin07] proposed two SVR models with modified RBF and linear kernels in order to capture periodic seasonal patterns in the data. However, in spite of their strength, SVR does not process data sequentially and does not capture temporal dependencies existing in the data.

Another approach for time-series modeling and forecasting is modeling the time-series through non-linear systems. The authors in [MS16a] proposed a curve fitting model using non-linear dynamical systems for co-evolving data streams. The proposed method is able to extract the patterns in the time-series and forecast up to a specific time point ahead. Same authors, proposed a similar method for mining co-evolving activities in the web in [MSF15]. However, this method is not scalable to large amount of data, which is prevalent in most temporal applications.

Traditionally, models based on neural networks were widely used for time-series forecasting problems such as traffic prediction [ZPH98, DC97]. Although neural networks can learn the correlations among the input features and the target variable effectively, they are limited in terms of capturing the temporal correlations as they observe input records independently. In order to better capture the temporal dependencies in sequential data, *RNNs* are widely used in machine learning research. [CPC⁺18] proposed a multivariate time-series prediction model based on GRUs, which could take leverage of the missing data to improve the prediction.

[MTW⁺15] proposed a model based on LSTMs for traffic speed prediction. The model proposed by [LGA17] used LSTMs on top of the features extracted by CNN to learn and predict the trends in time-series data.

Recurrent neural network based models are powerful in capturing long-term temporal dependencies in the data, which has led them to achieve the state-of-the-art results in different fields such as machine translation [SVL14], language modeling [MKB⁺10] and speech recognition [GMH13]. However, these models receive data in one dimensional vector form, which is not suitable for spatio-temporal data, where it is important to capture spatial dependencies in the data in addition to the temporal dependencies.

2.4.2 Spatio-Temporal Prediction

Spatio-temporal prediction problem refers to the prediction of the next state of a spatio-temporal space, by observing the historical data. The main difference with time-series forecasting problem is the existence of spatial information in such data. Therefore, for this type of prediction problem, it is important for the models to jointly capture both spatial and temporal dependencies in the data. Videos are one example of spatio-temporal data, as each frame contains spatial information and the sequence of frames constitute the temporal dimension of the data.

2.4.2.1 Spatio-Temporal Feature Learning

The emergence of Convolutional Neural Networks (CNNs) [LBBH98], revolutionized the feature learning process on visual signals. The ability of CNN models to learn spatial correlations in the data through local filters, and sharing parameters through the convolution process which significantly reduced the number of parameters in the model, made them hugely popular in variety of applications such as image classification, object detection, segmentation, etc. Although CNNs are great at capturing the spatial dependencies in the data, the temporal dimension also has to be considered in spatio-temporal prediction. Two types of temporal dependencies may exist in temporal data: 1) short-term dependencies: corresponding to local motions and 2) long-term dependencies: when a state, object or scene appears again after a while.

Capturing short-term dependencies: In their work on video classification, [KTS⁺14] introduced three approaches for capturing short-term dependencies in temporal data. 1) *early fusion*: By concatenating several consecutive frames in temporal dimension and applying convolution on the multi-channel image to extract features, 2) *late fusion*: By applying two separate single-frame networks on equally distanced frames, and merging the outputs together using a fully connected network and 3) *slow fusion*: By applying 3D convolution on consecutive frames to capture both spatial and temporal dependencies.

3D convolutions are shown to be effective in various spatio-temporal applications. [BMW⁺11, KTS⁺14, JXY13] used 3D convolutional kernels to capture local motions for human action

recognition task. However, [KTS⁺14] mentions that 2D convolutions on single frames is a powerful baseline, and applying 3D convolution on sequence of frames in some problems, can only incrementally improve the model accuracy. [TBF⁺15] extended famous VGG network [SZ14b] using 3D convolution, for feature learning task on videos. Another approach for capturing local motions in videos is to extract and use optical flows. [SZ14a] proposed a two stream convolutional network to learn features from raw data and optical flows separately and merged these features for action classification task.

Capturing long-term dependencies: For capturing long-term temporal dependencies in videos, different approaches for combination of CNNs and RNNs have been proposed. [BMW⁺11] was one of the earliest works to use LSTM on locally learned features using 3D ConvNets on videos, for action recognition task. [DAHG⁺15, YHNHV⁺15] used LSTMs on top of features learned from CNNs on video frames for activity recognition and video description problems. Inspired by sequence-to-sequence learning in natural language processing [SVL14], [SMS15] proposed an unsupervised representation learning method for videos. They used a fully-connected LSTM for encoding a sequence of frames and used two separate decoding streams for reconstructing previous frames and predicting next frames.

In addition to the above-mentioned approaches, the recent works of ConvLSTM [XCW⁺15] and ConvGRU [BYPC16] replaced the full connections of state-state and input-state transitions in GRU and LSTM, respectively, with convolution operations. This replacement had the two following implications: (1) better modeling of spatial regularities and (2) reducing the number of parameters significantly. The equations for ConvGRU gates, candidate state $\tilde{h}^{(t)}$ and hidden state $h^{(t)}$ are as follows:

$$\begin{aligned} z^{(t)} &= \sigma(W_z * [x^{(t)}, h^{(t-1)}]), \\ r^{(t)} &= \sigma(W_r * [x^{(t)}, h^{(t-1)}]), \\ \tilde{h}^{(t)} &= \tanh(W_h * [x^{(t)}, r^{(t)} \circ h^{(t-1)}]), \\ h^{(t)} &= (1 - z^{(t)}) \circ h^{(t-1)} + z^{(t)} \circ \tilde{h}^{(t)}. \end{aligned} \tag{2.19}$$

In Equation 2.19, $(*)$ symbol denotes convolution operation which has replaced the dot operation in original GRU equations (Equation 2.8), and $(\circ)$ denotes element-wise multiplication.

2.4.2.2 Video Prediction

Video prediction problem is defined as the problem of predicting next frames of a video, by observing the previous frames. With recent advances in deep learning research, this problem has been extensively studied in recent years. Capturing both spatial and temporal correlations in videos, and high dimensionality of such data, has made this problem challenging.

Based on the ConvLSTM and ConvGRU architectures, several studies further improved next frame prediction in videos [FGL16, BWSK17, LKC16]. [FGL16] utilized convolutional

LSTMs and proposed a deep learning model to predict physical motions on videos of robotic movements as well as human motions. [LKC16] also used convolutional LSTMs in their proposed model to predict future frames in videos. Their proposed model performed local prediction and forwarded the local errors to subsequent layers. [BWSK17] highlighted the problem of blurry prediction in videos and extended ConvLSTM network to propose ‘Parallel MD-LSTM’ units which were capable of capturing the past context for each pixel at different directions.

[MCL16] attributed the problem of blurry prediction in videos to the Mean Squared Error (MSE) loss function that is commonly used for prediction problems and proposed an improved loss function for video prediction problems. They also adapted adversarial training method introduced by [RMC15] along with a multi-scale architecture in their model. [OGL⁺15] proposed two deep learning based models for video prediction in Atari games, where the state of future frames are conditioned on user input. In a separate study, [VPT16] adapted the generative adversarial networks for video generation problem and proposed a deep learning based model using convolutional networks which was able to disentangle backgrounds and foreground in generating the scenes.

The proposed models for prediction of next frame in videos can capture spatial and temporal correlations in data well. However, utilizing the periodic patterns is not considered in these models, as recurring patterns which are more common in geo-spatial settings, are not prevalent in video data.

2.4.2.3 Geo-Spatio-Temporal Prediction

In geo-spatial prediction domain, [XCW⁺15] proposed the convolutional LSTM architecture to forecast the intensity of rainfalls in a spatial region. The full connections among state-state and input-state transitions were replaced with convolution operation in this model, which resulted in better capturing of spatial and temporal dependencies in the data. [ZDDG17] used latent representations to capture spatio-temporal dynamics and RNNs for decoding states into observations.

In another study in geo-spatio-temporal domain, [ZZQ17] proposed a deep learning model based on residual networks [HZRS16] to predict crowd and traffic flow in urban setting in near future. In their study, the geo-spatial input was formatted as images and the problem was formulated similar to next frame prediction problem in videos. However, only convolutional neural networks were used in this model, which are not suitable for capturing temporal dependencies in different spatial regions, from sequence of observations.

Although the reviewed methods are powerful in predicting the next state of a geo-spatio-temporal space, they mostly do not consider the recurring periodic patterns that may exist in such domains. Previous attempts to do so in deep learning based models for other time-series forecasting applications have been limited; for example, [ZZQ17] additionally used only raw input data from previous periods in their model.

2.5 Anomaly Detection in Temporal Data

Anomalies (or outliers) can be classified into three classes of: 1) *point outliers*, 2) *contextual outliers* and 3) *collective outliers* [CBK09]. Point outliers refer to the observations that are abnormally distant from other records in some space. Contextual outliers are the instances that are abnormal with respect to a given context such as time and location. Collective outliers refer to a series of outlier values that form a new pattern in the data. There is a rich literature on point anomaly detection using supervised methods, probabilistic and statistical methods and unsupervised methods such as clustering-based and density-based approaches [Agg15, CBK09]. However, such methods are mostly applicable on univariate or low-dimensional data. Moreover, in geo-spatial settings such as crowd density data, many abnormal instances are not distant from other observations, but occur at unusual times. Those anomalies in geo-spatio-temporal setting are contextual anomalies.

Outlier detection methods are also categorized with respect to other aspects such as dimensionality, supervision, output type (labeling versus scoring), data type (continuous, discrete), detection of global outliers versus local outliers and so on. In their survey on outlier detection on temporal data, [GGAH14] gave an overview of different outlier detection methods for time-series, data streams, network data, distributed data and spatio-temporal data. [Agg15] highlighted the difference between detection of anomalies in time-series data and in high-dimensional data. Due to immediate temporal continuity that exists in time-series data, the anomalies mostly occur in the form of abrupt changes. However, in high-dimensional data settings such as texts, the temporal continuity is much weaker as the points are independent from each other. In such settings the outliers may form novelties. However, this definition, may not be applicable to high-dimensional data such as videos where temporal continuity exists in the pixel level.

2.5.1 Contextual Anomaly Detection

Contextual anomalies are type of anomalies that may not deviate from other observations in the dataset but are abnormal with respect to a specific context such as time, space, etc. Comparatively fewer studies have been conducted on detection of contextual outliers.

One approach for handling contextual anomalies is to reduce the problem to point anomaly detection by considering only the data points that fall into a context at a time [CBK09]. Therefore, the availability of the contextual information is vital in such methods. [HDX02] used class labels as semantic information associated with instances, and detected outliers in each class using clustering-based methods. Spatial information is another type of contextual attribute. Different methods used graph structures to model spatial structure of data along with statistical methods to detect anomalous instances which were outlier with respect to their surrounding regions [SLZ02, KLC06, CS06]. Given an observation, [BM07] used a threshold-based approach by considering the median of the nearby points in time-series to detect outliers.

However, this method does not consider other temporal information such as time of day and day of week.

The contextual information is not always explicitly provided. Therefore, another approach for detection of contextual anomalies, is to fit a model on the normal instances and ask the model to predict or reconstruct the expected output. The data points that significantly deviate from the predicted output are identified as anomalies. Different forecasting models have been used for this purpose, which are explained in more detail in subsection 2.5.2. However, these methods are only applicable on low dimensional time-series data. Modeling the normal patterns from the structure of the data and detection of anomalies based on deviation from the model prediction are also widely used in abnormal pattern detection in high dimensional data such as videos. These methods are explained in detail in subsection 2.5.3.2.

2.5.2 Anomaly Detection in Time-Series Data

For unsupervised time-series outlier detection, linear models such as *ARIMA*, *VARMA* and statistical methods such as Cumulative Sum (CUSUM) are commonly used. In the prediction-based approaches the contextual anomalies are detected based on deviation from predicted values from the fitted regression model. Different forecasting models such as autoregressive models [HM10] and support vector regression models (SVR) [MP03] have been used for this purpose. [TPP00] studied outlier detection problem for multi-variate time-series data using Vector ARIMA model. In another study on multivariate time-series anomaly detection problem, [GPT06] proposed a model that detects multiple outliers by applying univariate outlier detection techniques on the projections of multivariate data. They showed that univariate outlier detection on useful projections of multi-variate data are more effective than VARMA based models that require manual model specification.

In this section we briefly reviewed the literature on outlier detection problem in time-series data. The proposed methods for time-series data are mostly statistical based methods and are applicable to univariate time-series data or low-dimensional multi-variate series. However, in spatio-temporal problems, we usually face with a large number of time-series data that are generated from multiple locations all at once, which might have correlations with each other. Anomaly detection models in such problems need to consider high-dimensionality of the data as well as the spatial dependencies that might exist between the nearby regions. In the following subsection, we review the literature on spatio-temporal anomaly detection problem.

2.5.3 Anomaly Detection in Spatio-Temporal Data

The goal of spatio-temporal anomaly detection methods is to identify spatio-temporal objects in the data, which show significantly different behavior from their temporal and spatial neighbors. [KB06] used a modified clustering algorithm based on DB-SCAN to create clusters from both

spatial and temporal neighbors of normal data points. Consequently, if a data point is not included in any of the clusters, it would be identified as a potential outlier. [WLC10] proposed a method for detection of spatio-temporal anomaly patterns in precipitation data. Their method applied a spatial scan statistic on different possible rectangular regions in the data to identify the regions with highest discrepancy over time.

[DP09] studied the problem of spatial and temporal outlier detection in global temperature and precipitation data. Distance-based outlier detection methods were applied on the data normalized over all regions and all time-stamps separately, to identify outliers. [YKX⁺05] used a distance based function, comparing fixed regions in a climate dataset, to detect outliers. In another study on vegetation data, [Las06] showed that transformation of data using PCA and mapping them into multiple principal components that decreasingly capture the dataset variance can be helpful in detection of anomalies.

The above-mentioned methods mostly detect spatio-temporal outlier objects using density-based or distance-based approaches. However, such methods can only detect point anomalies which are significantly different from all observations in the dataset. In some spatio-temporal problems, the observations can be anomalous with respect to some context such as temporal information. For detection of such anomalies, the distance-based or density-based approaches are not suitable.

In some spatio-temporal problems, the input data is formatted as high-dimensional matrices or tensors to better represent spatial and temporal features in the data. When the data dimension grows, the classic anomaly detection approaches on low dimensional data is not applicable anymore. Estimation of probability distribution of data is challenging in high dimensional space and different metrics such as Euclidean and Manhattan distance, which are used as the distance metric in many outlier detection methods, suffer from curse of dimensionality in high dimensional space. Considering these issues, the outlier detection methods to handle high dimensional data, are grouped into two main categories, which are reviewed here: 1) *Representation learning based methods* and 2) *Reconstruction based methods*.

2.5.3.1 Representation Learning based Methods

In representation learning based approach, the goal is to learn a lower dimensional representation of original data and apply outlier detection algorithms on that subspace. Deep neural networks such as Deep Belief Networks (DBNs) [Hin09], and autoencoders have been extensively used in the past few years for representation learning. In [ERKL16], DBNs were applied on high-dimensional data to learn representations in a lower dimension. One-class SVM trained on the learned representations detected the anomalies. Inspired by robust PCA [CLMW11], the authors in [ZP17] proposed an extension to deep autoencoders to remove outliers and noise data in an unsupervised fashion. The authors in [ZSM⁺18] proposed a model that utilized deep autoencoders to learn lower-dimensional features and estimated the density of the embedded features using Gaussian Mixture Model (GMM).

Videos are a special case of high-dimensional spatio-temporal data and many research studies have focused on abnormal event detection in data such as surveillance camera videos. In one study, stacked denoising autoencoders were used to learn feature representations from appearance (raw data) and motion (optical flows). Given the learned representations, one-class SVM classifiers were used to detect outliers [XYRS17]. Since the video data contain large amount of dense information, the compression process usually leads to loss of information in the bottleneck layer, making them unsuitable as representations. For such high volume data, reconstruction based approaches are mostly used in the literature.

2.5.3.2 Reconstruction based Methods

Representation learning based method have been shown to be effective for many application. However, when the dimensionality of the input data grows, too much compression would cause loss of information in the learned representations. On the other hand, if the dimensionality of the extracted features are not low enough, the common distance based or clustering based outlier detection methods are not effective. Hence, many anomaly detection methods rely on reconstruction error metric, especially for images and videos.

The intuition behind reconstruction based approach is that, the model learns to reconstruct or predict the normal pattern of data provided in the training set. Subsequently, the model would be able to reconstruct the normal patterns with low reconstruction error in the test time, while it would produce high reconstruction error on the input data that has not been observed before. Formally, the model learns to reconstruct the data in the training set by learning the parameter θ^* as follows [KTP18]:

$$\theta^* = \arg \min_{\theta} \sum_{x_i \in X_{train}} \| f_{\theta}(x_i) - x_i \|_2^2, \quad (2.20)$$

where f_{θ} is the approximation function for reconstruction of the data x_i . Due to their inherent ability to model complex non-linearities in high dimensions, deep learning based architectures such as deep autoencoders have been extensively used in recent years as reconstruction functions. [KTP18] reviewed the deep learning based methods for detection of abnormal patterns and anomalies in videos. Convolutional autoencoders (CAEs) are a special type of autoencoders where convolution layers are used as the encoder and decoder components of autoencoders. [HCN⁺16] used convolutional autoencoders with 2D kernels, on surveillance camera datasets, to learn regularity pattern in videos. The reconstruction error is used to compute regularity score and output anomaly frames and pixels.

In another work, [ZDS⁺17] utilized 3D convolutional layers in their autoencoder structure, to effectively capture temporal dependencies in the data. Moreover, two decoder branches were used for reconstruction of the current frames and prediction of future frames, to capture the

motion patterns more effectively. In [CYL11], an over-complete dictionary of normal patches observed during the training stage was created and sparse reconstruction cost was calculated given a new test frame. High reconstruction cost would signal abnormality at the frame level. As this method has high testing cost due to the comparisons with dictionary of normal patches, [LSJ13] further extended this method from algorithmic perspective to reduce the testing time.

Prediction models have also been utilized for capturing the normal pattern of data by predicting what frames would appear in future. Large prediction error would suggest that abnormal patterns that are unknown to the model have appeared. Convolutional LSTMs were used in few studies as the spatio-temporal prediction model [LLG17, MS16b]. [SSW⁺17] proposed an unsupervised detection method for marker discovery in medical images, using generative adversarial networks. After learning the manifold of normal images using a deep convolutional network, they used an optimization based approach to map test images into the latent space and compute the anomaly score. In a separate study, generative adversarial nets were used to learn the representation of normal instances using both motion and appearance information [RNS⁺17]. Since the model would learn to generate only normal instances provided in the training set, reconstruction of abnormal images would result in high error, which can be used to detect anomalies.

Reconstruction based approaches for anomaly detection, learn normal pattern of data generation process, and use reconstruction loss as the metric for detection of anomalies. Such methods can detect point anomalies (global anomalies), where the anomaly patterns have not been observed by the model in the training stage. However, they fail to capture contextual anomalies such as temporal anomaly instances, which has been observed by the model before, but have occurred in an unusual time context.

Chapter 3

Unsupervised Concept Tracking In Large Scale Multidimensional Data

In many real-world applications, a large volume of unlabeled temporal data is generated on a regular basis. From data analytics point of view, understanding and tracking the changes in such temporal data, can provide valuable information and interpretable insight for decision making processes. In this chapter, a model for tracking multiple concepts in unsupervised multi-dimensional data is proposed. This chapter is organized as follows. In Section 3.1, the problem setting and problem statement are introduced. In Section 3.2, the concept tracking model ConTrack is proposed. In Section 3.3 the algorithm implementations details are explained. The experimental setting and results are reported in Section 3.4. Finally, Section 3.5 summarizes the chapter¹.

3.1 Problem Statement

Our design objective is a model and algorithm that tracks pattern (i.e. concept) changes in large temporal datasets, where *multiple actors produce multiple records over time*. For example, in activity recognition, the attached sensor to a person (actor) generates motion records like acceleration signal on each dimensional axis, and we are interested in (1) tracking changes that occur across a number of actors simultaneously, such as an increase in walking speed (which might depend on time of day, weather, etc.) or an increase in running speed; (2) tracking the extent to which these changes occur for each actor individually in order to recognize the activity or combination of activities that an actor is performing. This leads to the following problem statement:

¹The work presented in this chapter has been published in [ZHKC16]

As **input**, we are given D -dimensional real-valued² temporal data $x_{ij}^{(t)}$, where $i \in \{1, \dots, N\}$ indexes actors, $t \in \{1, \dots, T\}$ indexes discrete time epochs, and $j \in \{1, \dots, N_i^{(t)}\}$ indexes the records associated with actor i at time t (of which there are $N_i^{(t)}$ in total).

The goal is to **output** (1) D -dimensional “concept vectors” (henceforth referred to as “concepts”) $B_k^{(t)}$. There are K concept vectors per time epoch, for a total of KT concept vectors. The concept vector $B_k^{(t)}$ represents the average value of concept k at epoch t (analogous to a cluster centroid in unsupervised clustering). (2) K -dimensional “concept participation vectors” $\theta_i^{(t)}$ (henceforth referred to as “participations”). There are N participation vectors per time epoch, for a total of NT participation vectors. The participation vector $\theta_i^{(t)}$ indicates what concepts actor i participates in at epoch t .

In order to design an efficient algorithm, we also introduce a K -dimensional, simplex-constrained “auxiliary” variable $Z_{ij}^{(t)}$ for each datum $x_{ij}^{(t)}$, that can be thought of as a concept participation vector specific to each record $x_{ij}^{(t)}$. This auxiliary variable $Z_{ij}^{(t)}$ is used to make the optimization tractable, and is not part of the model’s output.

To ensure interpretability, we require that each participation $\theta_i^{(t)}$ and $Z_{ij}^{(t)}$ must lie in the K -dimensional probability simplex; i.e. $\sum_k \theta_{ik}^{(t)} = 1$, $\theta_{ik}^{(t)} \geq 0$ and $\sum_k Z_{ijk}^{(t)} = 1$, $Z_{ijk}^{(t)} \geq 0$. For example, $\theta_i^{(t)} = [0.8, 0.2, \dots]$ means that “actor i participates 80% in concept 1 and 20% in concept 2 at time t ”. Negative values of $\theta_i^{(t)}$ are disallowed, as they are difficult to interpret. Table 3.1 summarizes the key terms used in this chapter.

Activity example: Let us consider an intuitive activity example: the D dimensions in each datum $x_{ij}^{(t)}$ correspond to motion features in a short movement window such as “average acceleration magnitude”, “velocity”, “maximum acceleration magnitude”. The number of concepts K corresponds to the number of physical activity patterns we would like to track (such as walking, running, etc.), and the series $B_k^{(1)}, \dots, B_k^{(t)}$ represents the evolution of concept k over time. Each $B_k^{(t)}$ represents the average value of concept k at time t ; for example, if average walking speed is increasing, we expect the concept $B_k^{(t)}$ associated with walking to increase in “velocity” as t increases.

The K -dimensional participations $\theta_i^{(t)}$ reveal how much each actor i participates in the K concepts — e.g. suppose an actor who is walking at time t slowly increases his speed, we might expect his/her participation in the “walking” concept to decrease at time $t + 1$, while participation in “running” could increase. The changes in $B_k^{(t)}$ and $\theta_i^{(t)}$ can be visualized to better identify areas for actionable insight (such as advertising running shoes to users who run many times a week, as might be detected by a smartwatch accelerometer); Figure 3.1 illustrates this idea.

²Categorical-valued dimensions or features can also be used, by transforming them into multiple binary-valued features using the well-known “one-hot” transform.

Notation	Type	Description
N	<i>Input</i>	Number of actors, indexed by i
T	<i>Input</i>	Number of time epochs, indexed by t
D	<i>Input</i>	Number of features/dimensions in actor records, indexed by d
$N_i^{(t)}$	<i>Input</i>	Number of records belonging to actor i at time t , indexed by j
$x_{ij}^{(t)}$	<i>Input</i>	The j -th record of actor i at time t . A D -dimensional vector.
K	<i>Parameter</i>	Desired number of concepts, indexed by k
$B_k^{(t)}$	<i>Output</i>	Average value of concept k at time t . A D -dimensional vector.
$\theta_i^{(t)}$	<i>Output</i>	Concept participation of actor i at time t . A K -dimensional simplex-constrained vector.
$Z_{i,j}^{(t)}$	<i>Auxiliary Variable</i>	Concept participation for record $x_{i,j}^{(t)}$. A K -dimensional simplex-constrained vector.

Table 3.1: Table of key terms and notation.

3.2 The Proposed Model

To solve the problem statement across multiple domains (e.g. activity, retail or sensor data), we propose an optimization model, **ConTrack**, that can (1) track K concepts $B_k^{(t)}$ that are shared amongst the N actors; (2) track each of the N actors' participation $\theta_i^{(t)}$ in the K concepts. At a high level, this optimization takes the form:

$$\min_{\theta, B, Z} L(x, \theta, B, Z) + g(\theta) + h(B) + f(Z), \quad (3.1)$$

such that $\theta, Z \in \Delta^{K-1}$ (the probability simplex). $L(\theta, B, Z)$ is a loss function that ensures B, θ closely match the data x , while $g(\theta)$, $h(B)$, $f(Z)$ and $f(\theta)$ are penalty functions to make θ, B, Z robust to noise. The constraint $\theta \in \Delta^{K-1}$ and $Z \in \Delta^{K-1}$ ensures the participation vectors θ, Z lie in the probability simplex (non-negative, elements sums to 1), aiding interpretability.

More concretely, ConTrack takes the following form:

$$\begin{aligned} \min_{\theta, B, Z} M(x, \theta, B, Z; \eta, \gamma, \beta, \alpha) = & \quad (3.2) \\ & \sum_{t=1}^T \sum_{i=1}^N \frac{1}{TNK} \|\theta_i^{(t)}\|_2 - \frac{1}{N_i^{(t)}} \sum_{j=1}^{N_i^{(t)}} Z_{ij}^{(t)} \|^2_2 + \sum_{t=1}^T \sum_{i=1}^N \sum_{j=1}^{N_i^{(t)}} \frac{1}{TN N_i^{(t)} D} \|Z_{ij}^{(t)} B^{(t)} - x_{ij}^{(t)}\|_2^2 \\ & + g_{t-robust}(\theta; \eta) + h_{t-robust}(B; \gamma) + f_{sparse}(Z; \beta) + f_{sparse}(\theta; \alpha), \\ s.t. \theta \in \Delta^{K-1}, Z \in \Delta^{K-1}. \end{aligned}$$

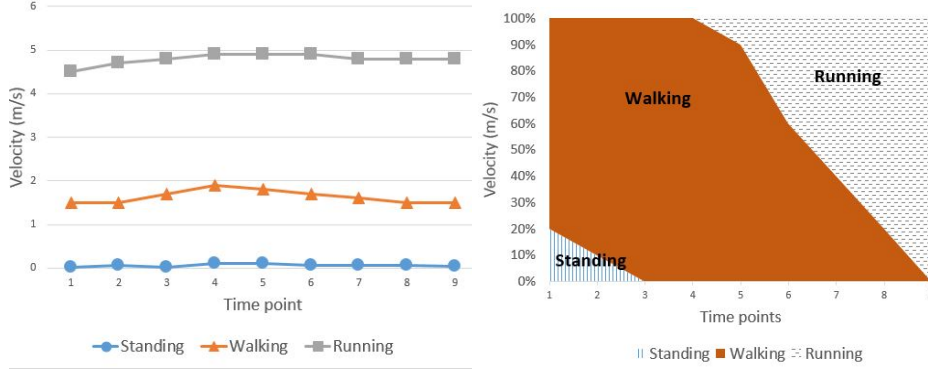


Figure 3.1: Visualization of changes in actor participations and concepts in activity data. **Left:** Tracking the changes in velocity feature of standing, walking and running concepts over time. **Right:** Tracking the changes in actor’s participation for standing, walking and running concepts over time.

The above form contains two least square terms, which we refer to as $L_1(\theta, Z)$ and $L_2(x, B, Z)$ respectively. By minimizing these terms, we find the most suitable values of $B^{(t)}$, $\theta_i^{(t)}$ and $Z_{i,j}^{(t)}$ to match the input data $x_{i,j}^{(t)}$.

In particular, each of actor i ’s records has its own participation vector $Z_{i,j}^{(t)}$, so it can pick the most appropriate concept; observe that the product $Z_{i,j}^{(t)} B^{(t)}$ represents the concept for data point $x_{i,j}^{(t)}$. Following this understanding, L_2 encourages data $x_{i,j}^{(t)}$ to be close to the chosen concept $Z_{i,j}^{(t)} B^{(t)}$. Moreover, L_1 ensures actor i ’s overall participation $\theta_i^{(t)}$ is close to the average of its record participations $(N_i^{(t)})^{-1} \sum_j Z_{i,j}^{(t)}$. The normalizers $\frac{1}{TNK}$ in L_1 and $\frac{1}{TNN_i^{(t)}D}$ in L_2 ensures that time epochs and number of concepts have proportional influence on the model, and all actors carry equal weight in the optimization (even though their number of records $N_i^{(t)}$ may differ, e.g. in data with power-law behavior).

The ConTrack model is *multimodal*, and suitable for two-level data where multiple actors have different number of records at each time point. By introducing the auxiliary variable Z for each record, the model is able to better represent the data by choosing a separate concept for each data point. We contrast this with the simpler loss function found in matrix factorization methods [GNHS11], $\|\theta_i^{(t)} B^{(t)} - x_{i,j}^{(t)}\|_2^2$, where all actor i ’s records are modeled using one *average* concept $\theta_i^{(t)} B^{(t)}$; this can result in very high loss function values when the concepts in $\theta_i^{(t)}$ are far apart. The difference between the MF-based, unimodal model and the ConTrack multimodal model is illustrated in Figure 3.2. Two-level data is also found in topic models such as LDA and Dynamic Topic Models [BL06, BNJ03], but we note that the settings are different: ConTrack takes real-valued data as input and is optimization-based, while topic models operate on text data and are typically probabilistic models. Furthermore, ConTrack is able to robustly track concepts and participations over time, as we shall explain next.

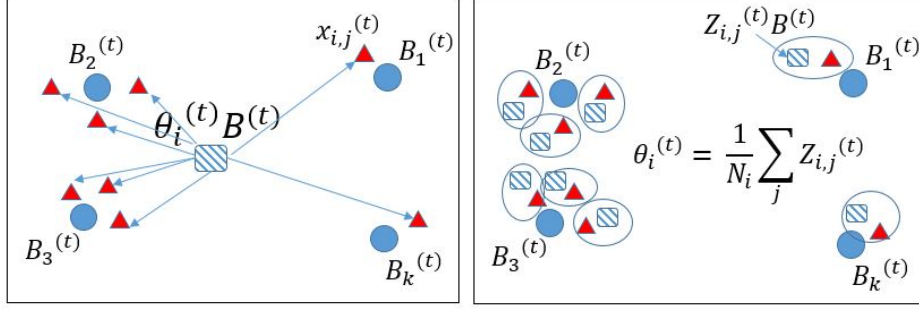


Figure 3.2: Unimodal loss function (**Left**, seen in matrix factorization) vs. Multimodal loss function (**Right**, seen in ConTrack). The unimodal loss function is not ideal for actors whose records participate in multiple well-separated concepts, because the actor-level average concept $\theta_i^{(t)} B^{(t)}$ is not near any of them — this in turn causes poor model fit. In contrast, ConTrack provides a better fit by allowing each record to have its own concept participation Z ; the actor-level concept is therefore a *mixture* over the individual record participations, and not an average.

In Equations 3.1, 3.2, we added penalty functions $g(\theta)$, $h(B)$, $f(\theta)$ and $f(Z)$ to improve the robustness of ConTrack in three respective ways: temporal robustness in participations θ , temporal robustness in concepts B , “one concept per record” (achieved through sparsity on Z), and “few concepts per actor” (achieved through sparsity on θ). We explain each type of robustness below:

Temporal robustness in participations: As time t passes, actors may increase or decrease their participation $\theta_i^{(t)}$ in different concepts, or may even disengage from existing concepts or engage to new concepts. However, the change should not be sensitive to outliers, i.e. $\theta_i^{(t)}$ cannot be too different from $\theta_i^{(t+1)}$ — for example, consider an environmental sensor that tracks the weather; we do not expect multiple rapid weather changes (such as “clear weather” to “raining” and back) over a short duration of a few minutes. We can encourage temporal robustness in participations $\theta_i^{(t)}$ through a least-squares penalty function $g_{t-robust}(\theta; \eta)$:

$$g_{t-robust}(\theta; \eta) = \sum_{t=1}^T \sum_{i=1}^N \frac{\eta}{TNK} \|\theta_i^{(t)} - \theta_i^{(t+1)}\|_2^2, \quad (3.3)$$

where η is a tuning parameter, and the normalizer $\frac{1}{TNK}$ ensures that actors, time epochs and concepts have proportional influence on the model.

Temporal robustness in concepts: Depending on the time scale of the data, we may want the concepts $B_k^{(t)}$ to be temporally robust to outliers at the next epoch $t+1$. For example, in activity recognition, we would not expect a user’s walking gait to change dramatically over a few seconds. This is accomplished through another least-squares penalty function $h_{t-robust}(B; \gamma)$:

$$h_{t-robust}(B; \gamma) = \sum_{t=1}^T \sum_{k=1}^K \frac{\gamma}{TKD} \|B_k^{(t)} - B_k^{(t+1)}\|_2^2, \quad (3.4)$$

where γ is a tuning parameter.

One concept per record: To ensure each record participation Z is associated with exactly one concept, we impose a strong sparsity constraint on Z that ensures most elements become zero. This reduces the complexity of the learned model, and provides the following benefits: (1) it reduces model identifiability issues, because allowing Z to take on any mixture of concepts unnecessarily increases the number of possible solutions to $B_k^{(t)}$; (2) it makes interpretation easier, since every record is clearly associated with one concept. We accomplish this via a penalty function $f_{sparse}(Z; \beta)$ (where β is a tuning parameter) that is inspired by the Dirichlet prior³ in statistics:

$$f_{sparse}(Z; \beta) = \sum_{t=1}^T \sum_{i=1}^N \sum_{j=1}^{N_i^{(t)}} \frac{1}{TNN_i^{(t)}K} \left(\sum_{k=1}^K -(\beta - 1) \ln(Z_{i,j,k}^{(t)}) \right). \quad (3.5)$$

The normalizer $\frac{1}{TNN_i^{(t)}K}$ ensures that actors, number of records for each actor, time epochs and concepts have proportional influence on the model⁴.

Few concepts per actor: We apply a similar sparsity constraint to θ , to ensure that actor participations are limited to a few (or only one) concepts. The penalty function $f_{sparse}(\theta; \alpha)$ is defined as follows, with similar normalizer to Equation 3.5:

$$f_{sparse}(\theta; \alpha) = \sum_{t=1}^T \sum_{i=1}^N \frac{1}{TNK} \left(\sum_{k=1}^K -(\alpha - 1) \ln(\theta_{i,k}^{(t)}) \right). \quad (3.6)$$

3.3 Implementation and Optimization

To solve for an optimal solution θ^*, B^* to the ConTrack model in Equation 3.2, we derive and implement parallel Stochastic Gradient Descent (SGD) equations and simplex projection (which maintains the simplex constraint on θ, Z). We also apply the Adaptive Gradient (AdaGrad) technique [DHS11] to automatically select an optimal step-size at each iteration, which relieves the practitioner from having to manually tune step sizes (which is necessary for first-order optimization algorithms such as SGD), while ensuring convergence in as few iterations as possible. Our parallel algorithm is implemented on the Petuum distributed Machine Learning framework [XHD⁺15], which allows for additional speedups via bounded-asynchronous communication, compared to MapReduce or Message-Passing implementations.

³The Dirichlet prior is defined as $Dir(x) = \prod_{i=1}^K x_i^{\alpha_i - 1}$. We use the negative logarithm of this function as our sparse penalty function $f_{sparse}(Z; \beta)$, in line with how the least-squares loss $L_2(x, B, Z)$ is the negative logarithm of a normal distribution.

⁴In general, we normalize every term $L(), g(), h(), f()$ in this fashion. This balances their effects on the model, and makes parameter tuning easier. For example, if the number of actors N is increased, the effect contributed by each individual actor i decreases proportionally, and we do not have to re-tune parameters related to concepts and time epochs.

3.3.1 Stochastic Gradient Descent Equations

Stochastic gradient descent (SGD) [Bot10] is a popular, large-scale iterative approach for minimizing the objective function. SGD computes the first derivative of the function on a subset of the input data, and takes a (weighted) step in the negative direction of the derivative. We show the partial derivatives of the objective function M in Equation 3.2 with respect to $\theta_{i,j}^{(t)}$, $B_{p,q}^{(t)}$ and $Z_{i,u,j}^{(t)}$, in Equations 3.7, 3.8 and 3.9 respectively.

$$\frac{\partial M}{\partial \theta_{i,j}^{(t)}} = \frac{2}{TNK} \left(\theta_{i,j}^{(t)} - \frac{1}{N_i^{(t)}} \left(\sum_{u=1}^{N_i^{(t)}} Z_{i,u,j}^{(t)} \right) \right) + 2 \frac{\eta}{TNK} \left(2\theta_{i,j}^{(t)} - \theta_{i,j}^{(t+1)} - \theta_{i,j}^{(t-1)} \right) + \frac{1 - \alpha}{TNK \theta_{i,j}^{(t)}} \quad (3.7)$$

$$\frac{\partial M}{\partial B_{p,q}^{(t)}} = \sum_{i=1}^N \sum_{j=1}^{N_i^{(t)}} 2 \frac{Z_{i,j,p}^{(t)}}{TN N_i^{(t)} D} \left(\left(\sum_{k=1}^K Z_{i,j,k}^{(t)} B_{k,q}^{(t)} \right) - x_{i,j,q}^{(t)} \right) + 2 \frac{\gamma}{TKD} (2B_{p,q}^{(t)} - B_{p,q}^{(t+1)} - B_{p,q}^{(t-1)}) \quad (3.8)$$

$$\begin{aligned} \frac{\partial M}{\partial Z_{i,u,j}^{(t)}} &= \frac{2}{TNK} \left(\frac{1}{N_i^{(t)^2} \sum_{p=1}^{N_i^{(t)}} Z_{i,p,j}^{(t)}} - \frac{\theta_{i,j}^{(t)}}{N_i^{(t)}} \right) + \sum_{d=1}^D 2 \frac{B_{j,d}^{(t)}}{TN N_i^{(t)} D} \left(\left(\sum_{k=1}^K Z_{i,u,k}^{(t)} B_{k,d}^{(t)} \right) - x_{i,u,d}^{(t)} \right) \\ &+ \frac{1 - \beta}{TN N_i^{(t)} K} \cdot \frac{1}{Z_{i,u,j}^{(t)}} \end{aligned} \quad (3.9)$$

3.3.2 Adaptive Gradient

Instead of using a constant step size during the stochastic gradient descent, we use the Adaptive subgradient method (Adagrad) [DHS11] to automatically compute the most effective step size, based on historical gradient information. Adagrad greatly reduces the number of iterations needed to converge to an optimal solution.

$$\gamma_{t,i} = \frac{\gamma}{\sqrt{G_{t,ii}}} \quad \text{where} \quad G_{t,ii} = \sum_{t'=1}^T g_{t',i}^2 \quad [\text{DHS11}] \quad (3.10)$$

γ is the initial step size in above formula and G is a diagonal matrix that its element (i, i) is the square summation of historical gradients for element i . Therefore, the update for element i is defined with the following equation.

$$x_{t+1,i} = x_{t,i} - \frac{\gamma}{\sqrt{G_{t,ii}}} g_{t,i} \quad [\text{DHS11}] \quad (3.11)$$

3.3.3 Simplex projection

As explained in Section 3.1, the K -dimensional actor participation vectors $\theta_i^{(t)}$ must lie in the $(K - 1)$ -dimensional probability simplex (Fig 3.3), so that they are interpretable. To enforce this, after each gradient step, we project the participation vectors $\theta_i^{(t)}$ so that $\sum_d \theta_{id}^{(t)} = 1$ and $\theta_{id}^{(t)} \geq 0$. We use the projection algorithm of [WCP13] (Algorithm 2) for this purpose.

Algorithm 2: Projection into probability simplex [WCP13]

Input: $y \in \mathbb{R}^D$

- 1: Sort y into u $u_1 \geq u_2 \geq \dots \geq u_D$
- 2: Find $\rho = \max\{1 \leq j \leq D : u_j + \frac{1}{j}(1 - \sum_{i=1}^j u_i) > 0\}$
- 3: Define $\lambda = \frac{1}{\rho}(1 - \sum_{i=1}^\rho u_i)$

Output: x s.t. $x_i = \max\{y_i + \lambda, 0\}$

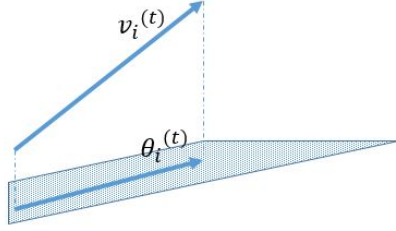


Figure 3.3: After each stochastic gradient step, the updated actor participation vectors $\nu_i^{(t)}$ may lie outside the K -dimensional probability simplex. We project them back into the simplex to obtain legitimate participation vectors $\theta_i^{(t)}$.

3.3.4 Distributed-Parallel Algorithm

Algorithm 3 shows the distributed implementation of our optimization model. In addition to the inputs from Table 3.1, our algorithm also takes as input the regularization parameters η, γ (for temporal robustness in θ, B) and β (for sparsity in record participations Z). Except for the places marked [single thread], our algorithm runs in parallel over worker threads in a distributed cluster, and the Petuum framework automatically handles communication among them (via the `clock()` function).

Our algorithm performs updates epoch-by-epoch; we found that this improves accuracy when temporal robustness is turned on. Based on Equations 3.7, 3.8, 3.9, our algorithm first computes the gradients for B, θ, Z , and then uses Adagrad to apply these gradients with a suitable step size (Equations 3.10, 3.11). After the Adagrad update, we finally project $\theta_i^{(t)}$ and $Z_{i,j}^{(t)}$ back into the probability simplex using Algorithm 2.

Algorithm 3: Distributed optimization algorithm

Input: $X_{R \times D}, T, K, N, D, \eta, \gamma, \beta$

- 1: Initialize B ; [single thread]
- 2: Initialize θ, Z ;
- 3: clock(); // Inter-machine communication
- 4: read B ;
- 5: **for** $m = 0$ to $iterations$ **do**
- 6: compute objective value;
- 7: **for** $t = 0$ to T **do**
- 8: Compute $B^{(t)}, \theta^{(t)}$ gradients;
- 9: clock(); // Inter-machine communication
- 10: **for** $k = 0$ to K **do**
- 11: Compute $B_{k,d}^{(t)}$ Adagrad step size; [single thread]
- 12: Update $B_k^{(t)}$; [single thread]
- 13: **end for**
- 14: **for** $i = 0$ to N **do**
- 15: Compute $\theta_{i,k}^{(t)}$ Adagrad step size;
- 16: Project $\theta_i^{(t)}$ on simplex;
- 17: Update $\theta_i^{(t)}$;
- 18: **end for**
- 19: clock(); // Inter-machine communication
- 20: read $B^{(t)}$;
- 21: Compute $Z^{(t)}$ gradients;
- 22: **for** $i = 0$ to N **do**
- 23: **for** $j = 0$ to $N_i^{(t)}$ **do**
- 24: Compute $Z_{i,j,k}^{(t)}$ Adagrad step size;
- 25: Project $Z_{i,j}^{(t)}$ on simplex;
- 26: Update $Z_{i,j}^{(t)}$;
- 27: **end for**
- 28: **end for**
- 29: **end for**
- 30: **end for**

Output: $\theta_{N \times K}, B_{K \times D}$

3.4 Experiments

We conduct two sets of experiments, one to evaluate the performance of our method at tracking multiple changing concepts in multi-user unsupervised and semi-supervised data, the other to evaluate the scalability of our method with increasing parallelization (more worker threads and more machines).

3.4.1 Experimental Setting

Baselines: Because our setting involves multiple concepts, multiple users and unsupervised data, none of the concept drift and stream mining algorithms mentioned in our related work section can be directly applied to our problem. Instead, we shall compare ConTrack against four well-known baselines from the unsupervised learning literature: (1) Kmeans++ (2) Matrix Factorization (3) Sparse Coding and (4) Fuzzy C-Means (FCM).

The baselines used in this study are well-known unsupervised methods that are modified and used for the task of concept tracking. In particular, the clustering based methods (Kmeans++ and FCM) can identify the concepts by clustering the data at individual time epochs and link the concepts between two consecutive time epochs to track the changes. In addition, soft clustering methods such as Fuzzy C-Means are capable of assigning probability of records belonging to multiple clusters, similar to our participation vectors. In addition to clustering methods, decomposition methods (MF and Sparse Coding) can decompose the large data matrix into two smaller matrices corresponding to our concepts and participation vectors.

Kmeans++ [AV07] takes the well-known Kmeans algorithm for clustering unsupervised data, and augments it with an initialization strategy that effectively avoids poor local minima. By setting the number of clusters K to the number of concepts to be extracted, Kmeans++ can attempt to learn the concepts B and participations θ in our problem setting. Matrix factorization [GNHS11] decomposes an input matrix $R_{N \times D}$ into two matrices $P_{N \times K}$ and $Q_{D \times K}$ where $P * Q' \approx R$. As with Kmeans++, by setting the number of latent features K to the number of concepts to be extracted, MF can learn B, θ . Sparse coding (SPC) [OF97] is similar to MF, in that it decomposes an input matrix into two smaller matrices (dictionary matrix, corresponding to concepts B , and coefficient matrix, corresponding to participations θ). However, SPC requires the generated coefficient matrix to be sparse (in the same way our model supports sparse θ through g_{sparse}). Finally Fuzzy C-Means [BEF84] is a clustering algorithm where data points can belong to more than one cluster by assigning membership levels. We map the clusters to the concepts and membership levels to the concept participation in order to generate B and θ .

Because all the baselines do not have an explicit notion of actors/records or time epochs, we must “transform” their inputs/outputs to and from our problem setting, in order to ensure a fair evaluation. This transformation procedure will be explained in detail later.

Datasets: In the multi-concept, multi-user and unsupervised problem setting, ground-truth data is not widely available, and we therefore use a mix of synthetic and real datasets for our evaluation. Following the strategy employed in recent concept drift research [QAWZ15], in

Section 3.4.3, we begin with (1) fully synthetic data (Synthetic), and (2) “partly synthetic” data (Stock, Activity) that is simulated using real-world multivariate time-series data as a starting point. In either case, we insert drifts into the concepts B and actor participations θ , which the baselines and methods are required to track. Table 3.2 summarizes the basic statistics of each dataset.

In the Synthetic dataset, we simulated daily retail banking transactions with 100 concepts $B_k^{(t)}$ (corresponding to spending categories such as transportation, food, luxury goods, etc.), and 1000 users with varied participation $\theta_i^{(t)}$ in these concepts. The data consists of features engineered from basic information such as “merchant ID”, “date/time of transaction” and “dollar amount”; each actor’s records $x_{ij}^{(t)}$ were generated by drawing random vectors in a radius around one of the concepts $B_k^{(t)}$, where the concepts $B_k^{(t)}$ are selected in proportion to the actor’s concept participation $\theta_i^{(t)}$.

The Stock and Activity datasets were partly synthesized from real-world multivariate temporal datasets⁵. We first manually identified well-separated groups of data samples (different stock symbols, or different human participants), and used each group as the basis for one concept $B_k^{(t)}$. We then simulated actors by generating random vectors around the concept, similar to the Synthetic dataset.

In our second experiment in real world data, (Section 3.4.4), we use an activity recognition dataset with ground truth [CPR11] to evaluate the performance of our method. This dataset contains acceleration time series data from several participants, who performed pre-defined activities such as walking, standing, going up/down stairs, etc.. The data was captured using a single accelerometer sensor placed on each participant’s chest. From this dataset, we selected 5 activities (concepts) for evaluation; no synthetic data is created from this dataset. Detailed statistics are shown in Table 3.2.

Drift Contexts: In our problem setting, there are two dimensions to drift classification — whether the drift occurs in concepts $B_k^{(t)}$ (Concept Drift) or in actor participations $\theta_i^{(t)}$ (User/Actor Drift) or both. We refer to this distinction as “Drift Context”. Our experiments consider three Drift Contexts: (1) drift in concepts only (Concept Drift) (2) drift in actors/users only (User Drift) and (3) drift in both actors and concepts (User-Concept Drift). In User-only Drift, actor participations $\theta_i^{(t)}$ gradually change across several epochs, while the concepts remain fixed. Conversely, in Concept-only Drift, the actor participations remain fixed while the concepts are moving across the feature space. Finally, in User-Concept Drift, we alternate between concept drift and actor drift in different epochs. Considering three types of drift contexts for the synthetic and partly-synthetic datasets, we evaluate performance of the ConTrack model and compare it with the baseline methods.

⁵We used the Dow Jones stock dataset [BPD13], which provides weekly changes in stock symbols over 6 months, and a physical activity monitoring dataset [RS12], in which the physical activity of several human subjects was monitored via a group of sensors, over 3-5 minute periods.

Dataset	Context	Size	Concepts	Users	Dim	Epochs
synthetic	User Drift	62MB	100	1000	22	30
	Concept Drift	61MB	50	1000	22	30
	User-Concept Drift	25MB	50	1000	22	30
Stock (semi-synthetic)	User Drift	110MB	20	1000	13	30
	Concept Drift	89MB	20	1000	13	24
	User-Concept Drift	111MB	20	1000	13	30
Activity (semi-synthetic)	User Drift	503MB	16	1600	39	30
	Concept Drift	506MB	16	1600	39	30
	User-Concept Drift	307MB	16	1600	39	30
Activity Recognition (Real)		48MB	5	14	3	90

Table 3.2: Datasets statistics

3.4.2 Evaluation Procedure

Transformation of inputs and outputs for baselines: Our problem setting and ConTrack model take temporal, actor-organized records $x_{ij}^{(t)}$ as input, and output concepts $B_k^{(t)}$ and participations $\theta_i^{(t)}$. In order to evaluate all baselines in a consistent and fair manner using NMI, we must transform our data $x_{ij}^{(t)}$ into a format that they can use, and also transform their outputs into participations $\theta_i^{(t)}$.

Kmeans++, MF, SPC and FCM are unsupervised methods that do not consider temporal and actor-related information, so we remove these aspects of the data before inputting it to those methods. Specifically, we “flatten” all records $x_{ij}^{(t)}$ into a simple list of records x_j (where j is the record index), and we take note of each record’s original actor i and time epoch t . We then take the output result from each method, and use the original actor and time epoch labels to transform the output into participations $\theta_i^{(t)}$, similar to those generated by ConTrack. For example, in Kmeans++, we create a participation vector $v_i^{(t)}$ for each actor, where $v_{ik}^{(t)}$ counts the number of actor i ’s records that were assigned to cluster k at time t . The vector $v_{ik}^{(t)}$ is then normalized such that $0 \leq v_{ik}^{(t)} \leq 1$ and $\sum_k v_{ik}^{(t)} = 1$, resulting in a simplex-constrained participation vector $\theta_i^{(t)}$.

Initialization procedure: ConTrack solves a non-convex optimization problem, and therefore sensitive to the choice of initialization. We use a probabilistic seeding strategy (similar to Kmeans++’s seeding method) to initialize concepts and participations — this is not only (1) cheap to compute, but also (2) greatly improves our accuracy over naive random initialization.

Normalized Mutual Information (NMI) scoring [Ho14]: In order to evaluate the results

Dataset	Drift Context	Kmeans++		FCM		SPC		MF		ConTrack	
		$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.1$	$\sigma=0.2$
Synthetic	User Drift	0.97	0.96	Fail*	Fail	0.45	0.48	0.34	Fail	0.97	0.96
	Concept Drift	0.57	0.57	0.27	0.48	0.34	0.40	0.11	Fail	0.97	0.95
	User-Concept Drift	0.52	0.52	Fail	Fail	0.33	0.35	0.31	0.31	0.58	0.57
Stock	User Drift	0.99	0.99	0.99	0.99	0.59	0.64	0.41	0.39	0.99	0.99
	Concept Drift	0.27	0.27	0.3	0.3	0.13	0.12	0.14	0.16	0.74	0.74
	User-Concept Drift	0.31	0.31	0.31	0.31	0.16	0.17	0.16	0.19	0.72	0.72
Activity	User Drift	0.89	0.87	Fail	Fail	0.67	0.63	0.34	0.49	0.9	0.88
	Concept Drift	0.53	0.53	Fail	Fail	0.32	0.32	0.16	0.19	0.91	0.91
	User-Concept Drift	0.39	0.39	0.18	0.35	0.19	0.2	0.16	0.15	0.48	0.48

Table 3.3: NMI scores obtained by each method on synthetic and real datasets. “Fail” denotes cases where the procedure did not output any concepts for actors (because the output fails to meet the threshold σ).

generated by our models and the baselines, we choose to measure how well the recovered actor participations $\theta_i^{(t)}$ match the ground truth. To do so, we first make the following definitions: “actor i subscribes to concept k at time t if $(\theta_i^{(t)})_k \geq \sigma$ ”, where σ is a threshold. We then define “concept-set” for concept i at time t to be “the set of all actors subscribed to concept i at time t ”.

Our evaluation objective is to measure how well the concept-sets output by each algorithm match the ground truth concept-sets. To measure this, we use the Normalized Mutual Information (NMI) scoring function, as described in [Ho14]. NMI takes two concept-sets as input, and outputs a score between 0 and 1, where 1 represents a perfect match.

Machine Specifications: Except for scalability experiments, most of our evaluations were conducted on two machines: one with 16GB RAM and 4 CPU cores (used for Kmeans++, FCM and ConTrack), the other with 112GB RAM and 16 CPU cores (used for MF and SPC, because they require more memory). For the scalability experiments involving only our algorithm, we

used a 4-machine cluster with 112GB RAM, 16 cores per machine, connected via Gigabit Ethernet.

3.4.3 Synthetic and Semi-Synthetic Experiments

Using the synthetic and partly synthetic datasets described in Table 3.2, we evaluated the performance of the ConTrack model and compared its performance with other baselines in Table 3.3. Each entry in the table indicates the NMI score obtained by a particular method on a particular dataset and Drift Context, at a particular subscription threshold σ (we report results for two thresholds, $\sigma=0.1$ and 0.2).

To find robustness parameters $(\alpha, \beta, \gamma, \eta)$ suitable across all synthetic and semi-synthetic datasets, we performed a gridsearch and found that $\alpha = 1$, $\beta = 0.9$, $\gamma = 1.0$ and $\eta = 1.0$ yielded the best average results. We therefore used these values for the experiment.

As indicated in Table 3.3, the ConTrack model outperforms all other baselines in accuracy. In User-only Drift datasets (concepts are not moving), clustering algorithms like Kmeans++ and FCM show good performance, while ConTrack is still able to get equal or better results. However, once we shift to settings where the concepts may drift (Concept-only Drift and User-Concept Drift), ConTrack substantially outperforms other baselines. In terms of running time, ConTrack and Kmeans++ required less than 5 minutes to run on the synthetic dataset with 100 concepts, while FCM required near 1 hour, and Matrix Factorization as well as Sparse Coding required 30 minutes (when using a higher performance machine than ConTrack and Kmeans++).

The NMI scores obtained by setting $\sigma = 0.2$ on MF and SPC are generally higher than those obtained by $\sigma = 0.1$. For other baselines, the difference is not significant. Finding a suitable subscription threshold is a trade-off that we intend to investigate thoroughly in future work.

3.4.4 Real-World Activity Recognition Experiments

To illustrate how ConTrack can be applied to a real-world scenario, we conducted an experiment on human activity data captured from a single chest-mounted accelerometer, which provides three-dimensional time series data. In this dataset [CPR11], several participants are asked to perform activities such as working at a computer, standing, walking, going up/down stairs, and talking while standing; details are in table 3.2. For example, Figure 3.4 shows the X -axis acceleration for each activity performed by one of the participants.

Treating each activity as one underlying concept, we used ConTrack to track users' participations in concepts. The concept output from ConTrack shows how each activity is drifting over time, while the user participation output shows how users change their activity composition as time passes — note that ConTrack not only performs activity recognition, but also tracks evolution in the shared concepts. We evaluated the accuracy of ConTrack by comparing the user participation output to the ground truth activity labels in the original data.

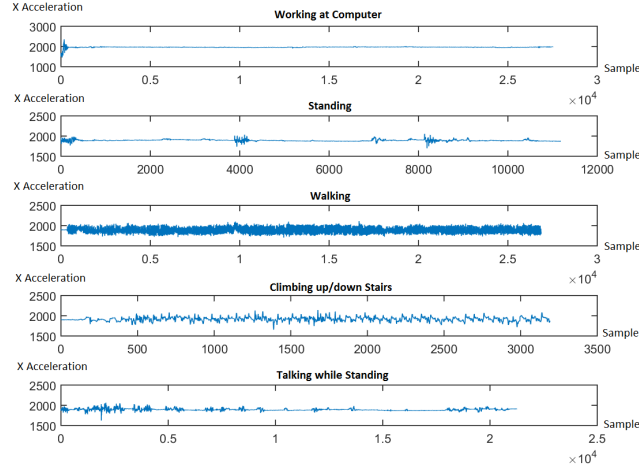


Figure 3.4: Acceleration time series data on X axis for different activities performed by one participant.

We prepared the data for ConTrack by extracting 90000 records per user, from the accelerometer time series data. Each group of 1000 records was merged into one epoch, for a total of 90 epochs. While the original data contains acceleration time series on each axis A_x , A_y , A_z , we choose to follow the feature engineering guidelines in [CPR11], and thus added extra features to improve the discriminative power of ConTrack. First, we added the magnitude of acceleration $A_m = \sqrt{A_x^2 + A_y^2 + A_z^2}$, as well as low-pass and high-pass filters at the cutoff frequency of 2.6 Hz. This resulted in 12 time-series vectors, to which we added even more features by applying various window-based functions (window length of 50 samples, adjacent windows overlap by 50%). The feature details are shown in table 3.4; in total, 117 features were generated. To tune the ConTrack parameters — sparsity on participations and records (α, β), temporal robustness on participations (η) and temporal robustness on concepts (γ) — grid search was used to find the best parameter configuration, which was found to be $\eta = 1000$, $\gamma = 1000$, $\alpha = 0.95$ and $\beta = 0.95$.

In activity recognition applications, a small number of ground truth labels are usually available (e.g. users may occasionally self-declare their activities on a smartphone app). To improve its tracking performance, ConTrack can exploit labels in a semi-supervised fashion, by using a fraction of the labels as seeds to the initialization procedure. Furthermore, the use of labels allows us to compare ConTrack’s performance with supervised activity recognition classifiers.

Table 3.5 shows the NMI scores obtained by ConTrack and Kmeans, using different percentages of data labels as seeds to initialization. ConTrack enjoys substantially better performance than Kmeans, confirming that a temporal model is required to track the many concept drifts that occur in the dataset, especially over a long duration. When using additional concept labels (more supervision) in the initialization process, the NMI score of ConTrack increases as expected, whereas Kmeans is unable to make use of the additional labels.

Feature	Explanation
Mean	Mean of values in the window
Std	Standard deviation of values in the window
Skewness	Skewness of values in the window
Kurtosis	Kurtosis of values in the window
Correlation	Pair-wise correlation between vectors
RMS Velocity	RMS of integration of acceleration in window
Energy	Coefficients energy of 7 level wavelet decomposition
Min-Max	Sum of all differences between pair-wise min-max peaks
Peaks-Num	Total number of high peaks in the window
Peak-Dist	average distances between sequential peaks

Table 3.4: Features added to the original time-series motion data.

To compare with previous results in the literature, we also evaluate ConTrack using a standard accuracy metric, in which we simply assign the dominant concept in the concept participation vector θ as the output activity class (i.e. every user can only participate in a single concept at any given time). Under this metric, ConTrack achieves 0.91 accuracy with 20% supervised labels, which is comparable to state-of-the-art activity classification methods; e.g., the classifiers in [CPR11] achieve accuracy scores between 0.9 to 0.94 on the same dataset using 20% supervised labels. However, unlike classifier methods, ConTrack is also able to track evolving concepts and participations; such changes are to be expected in long time-series data. Furthermore, our experiments also rule out classical unsupervised methods (e.g. Kmeans), which are simply unable to track evolving concepts in real-world data.

% labels used as initialization seeds	ConTrack	Kmeans
5%	0.20	0.003
10%	0.47	0.029
20%	0.70	0.006
33%	0.82	0.006
50%	0.91	0.028

Table 3.5: NMI score for ConTrack and kmeans on activity recognition data. The original data is fully-labeled, so 5% labels means that 5% of the time points are initialized with labels as seeds.

3.4.5 Scalability Experiments

In order to verify that our ConTrack implementation scales to larger input data, we performed a cluster experiment on 4 machines (specifications in Section 3.4.2), where we generated a larger 5GB version of the Synthetic Concept Drift dataset (containing 50 concepts and 10000 users).

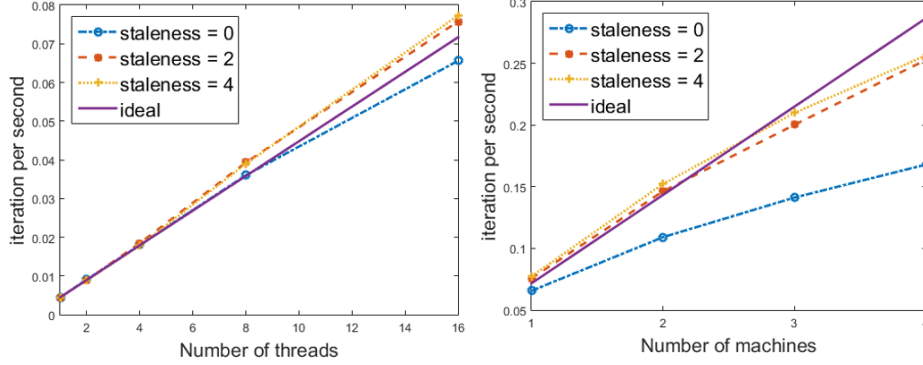


Figure 3.5: Speedup achieved for different staleness settings on (a) 1-16 threads on 1 machine, (b) 1-4 machines each one running with 16 threads. The solid line represents ideal speedup with more threads/machines.

Our algorithm is parallelized by dividing the actors and their records over separate threads. The threads perform optimization on their own data chunks and do synchronization whenever needed. The practitioner is able to set the number of threads and number of machines to control the level of parallelization.

In our experiments, we ran our algorithm for 30 iterations, using different numbers of threads (1 to 16) on one machine, as well as different numbers of machines (1 to 4) with the maximum number of threads per machine (16). For each experiment, we report the iterations executed per second (i.e. the speed of the algorithm), as well as the NMI score after 30 iterations in Table 3.6. Figure 3.5 plots the speedup (iterations/sec) achieved with more threads, and shows close to linear scalability when using modest “staleness” settings between 2-4.⁶ Moreover, the model’s accuracy is not adversely affected by the choice of parallel setting.

3.5 Summary

In this chapter, we proposed ConTrack, a scalable method to track multiple concepts and actors (who participate in these concepts) in unsupervised data, in order to provide interpretable insights for data analysts. ConTrack handles the unsupervised, multi-actor, multi-concept problem setting that has received little attention in the concept drift and stream mining literature, and is formulated as an optimization problem in order to exploit established parallelization techniques and distributed programming systems. Our evaluation on synthetic and real datasets confirms that ConTrack is superior to other unsupervised methods in tracking multiple evolving concepts.

⁶Staleness is a tunable parameter in the Petuum system [XHD⁺15]. Increasing staleness reduces communication costs, and allows the algorithm’s per-iteration rate to approach ideal speedup Pr iterations/second, where P is the total number of worker threads in the cluster, and r is the number of iterations executed per second when the algorithm is run with a single thread.

Machines	Threads	iter/sec			NMI		
		s=0	s=2	s=4	s=0	s=2	s=4
1	1	0.004	0.004	0.004	0.95	0.95	0.95
	2	0.009	0.008	0.008	0.95	0.95	0.95
	4	0.018	0.018	0.017	0.95	0.95	0.95
	8	0.036	0.039	0.038	0.95	0.95	0.95
	16	0.065	0.075	0.077	0.95	0.95	0.95
2	32	0.109	0.146	0.152	0.85	0.90	0.95
3	48	0.141	0.200	0.209	0.92	0.90	0.91
4	64	0.168	0.252	0.256	0.94	0.88	0.95

Table 3.6: Iterations per second and NMI scores on our 5GB dataset, running on 1 Machine (1-16) threads, as well as 1-4 machines with 16 threads each, using staleness factors 0,2,4

Chapter 4

Next State Prediction of Spatio-Temporal Data with Recurring Periodic Patterns

Prediction of future state of a dynamic environment, is crucial for many different domains and applications. One typical application is future crowd density estimation in geo-spatial settings. Today, with the large amount of location reporting sensors available in mobile devices, vehicles and other moving objects, it is possible to build accurate models for a different applications such as estimating traffic flow and congestion, predicting taxi demand, and forecasting people crowd density in public transportation stations.

In this chapter, we introduce our proposed model for prediction of next state in spatio-temporal settings, by taking leverage of the recurring periodic patterns which are observable in geo-spatial data. The chapter is organized as follows. In Section 4.1, the problem is defined and the problem setting is introduced. In Section 4.2, the proposed Periodic Convolutional Recurrent Network (PCRN) model is introduced. In Section 4.3, the implementation and optimization details are explained and in Section 4.4, the experimental studies are reported. Section 4.5 summarizes the chapter¹.

4.1 Problem Statement

We define the geo-spatio-temporal prediction problem similar to next frame prediction problem in a video sequence. The observations of crowd density or any other feature of interest for each pre-defined geographic region during a fixed time interval are aggregated and mapped into a 2-D dimensional multi-channel image $X^{(t)} \in \mathbb{R}^{M \times W \times H}$ for discrete time point t , referred to as ‘spatial image’ hereafter. Here W, H, M refer to the width, height and number of channels of spatial images, respectively. Therefore, $\{X^{(1)}, X^{(2)}, \dots, X^{(T)}\}$ represents the set of all spatial images used for model creation.

As the input, a sequence of spatial images $\{X^{(t-\tau+1)}, \dots, X^{(t)}\}$ is given to the model, where τ is a pre-defined sequence length. As the outcome of our model, we are interested in predicting

¹The work presented in this chapter has been published in [ZKLC18]

a spatial image for the next time interval $X^{(t+1)}$. Since the temporal information of the spatial images is implicated in spatio-temporal setting, we also treat this metadata information as an additional input to the model. Altogether, PCRN is proposed as an optimization model to learn an approximation function f , based on a deep neural network architecture and formulated as:

$$\hat{X}^{(t+1)} = f(X^{(t)}, \dots, X^{(t-\tau+1)}, X_{meta}^{(t+1)}; \theta). \quad (4.1)$$

In Equation 4.1, $X_{meta}^{(t+1)}$ denotes the temporal metadata information for the predicted frame, and θ are the parameters that the model learns. $\hat{X}^{(t+1)}$ is generated by an approximation function over previously observed spatial images defined by a deep neural network architecture, which will be explained later.

In Equation 4.2, we define the objective function in least-square form, to minimize the Euclidean distance between the the predicted image at the next time point and the ground truth.

$$\mathcal{L}(\theta) = \frac{1}{T - \tau} \sum_{t=\tau}^{T-1} \| X^{(t+1)} - \hat{X}^{(t+1)} \|_2^2. \quad (4.2)$$

Spatial Images: We create spatial images from the records generated by independent actors, which contain both temporal (time-stamp) information and geo-spatial coordinates.

Formally each record $r^{(t',l)} \in \mathbb{R}^D$ is a D dimensional record generated at time t' , and at location $l = \langle lat, lon \rangle$ indicates latitude and longitude coordinates of the generated record respectively. $r_d^{(t',l)}$ indicates the d -th dimensional value of the record $r^{(t',l)}$.

To create spatial images from records, we partition the area to be predicted into a $W \times H$ grid map, and define mapping function $g(l)$, to map records located inside the partitioned area into their corresponding grids in the grid map. Finally, for a discrete time point t , representing a fixed interval, we create spatial image $X^{(t)}$ as follows:

$$X_{(d,i,j)}^{(t)} = |\{r^{(t',l)} | g(l) = \langle i, j \rangle, t \leq t' < t + v, \mathcal{I}(r_d^{(t',l)})\}|. \quad (4.3)$$

The number of records satisfying an arbitrary requirement based on feature $r_d^{(t',l)}$, generated during interval of length v from discrete time point t , and located at grid (i, j) are aggregated together, and set as the pixel value of $X_{(d,i,j)}^{(t)}$. Therefore, the multi-channel image $X^{(t)}$ will represent the crowd density in discrete time interval t . Figure 4.1, shows examples of spatial images used as input of the model.

4.2 The Proposed Method

In this section, we introduce the proposed PCRN model and its components: (1) Pyramidal convolutional recurrent network, (2) Periodic representations and (3) Fusion process in detail. Finally we explain the implementation and design decisions related to our model.

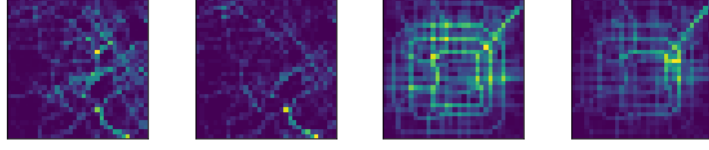


Figure 4.1: Examples of 32×32 partitioned spatial images. Lighter colors represent higher density. Left to right: volume of all taxis and taxis with passengers on board in downtown Singapore, in-flow and outflow of taxis in Beijing.

Periodic-CRN (PCRN) model proposed in this chapter consists of three main components: (1) Convolutional recurrent network (CRN) to capture spatio-temporal representation of an input sequence, (2) Periodic representation dictionaries to dynamically reuse and update representations from multiple periodic patterns, and (3) Fusion process to merge periodic representations with current input sequence representation and temporal metadata. The overall model architecture is illustrated in Figure 4.2.

4.2.1 Pyramidal CRN Model

Capturing spatial and temporal correlations is crucial for building accurate spatio-temporal prediction models. As in many geo-spatial problems such as crowd density or traffic congestion prediction, the state of traffic in one region inevitably influences nearby regions.

Specifically, we consider using Convolutional GRU [BYPC16] as our recurrent neural network architecture, as it has shown good performance, but with fewer parameters to be optimized. The equations for ConvGRU gates, candidate state $\tilde{h}^{(t)}$ and hidden state $h^{(t)}$ are shown in Equation 4.4 as follows:

$$\begin{aligned} z^{(t)} &= \sigma(W_z * [x^{(t)}, h^{(t-1)}]), \\ r^{(t)} &= \sigma(W_r * [x^{(t)}, h^{(t-1)}]), \\ \tilde{h}^{(t)} &= \tanh(W_h * [x^{(t)}, r^{(t)} \circ h^{(t-1)}]), \\ h^{(t)} &= (1 - z^{(t)}) \circ h^{(t-1)} + z^{(t)} \circ \tilde{h}^{(t)}. \end{aligned} \tag{4.4}$$

In Equation 4.4, $(*)$ symbol denotes convolution operation and $(\circ)$ denotes element-wise multiplication. Here $h^{(t)}$ is the hidden state tensor preserving the sequential information up to time point t .

Stacking convolutional RNN layers would result in a wider receptive field over nearby spatial regions and take their influence into account in upper layers, which leads to better representations and more accurate predictions. At the same time, stacking more layers with the same number of time steps would increase computational complexity and memory consumption significantly, resulting in very slow convergence.

Inspired by the RNN structure for speech recognition proposed in [CJLV16], we adapt a pyramidal structure for our multi-layer Convolutional GRUs, where the consecutive outputs

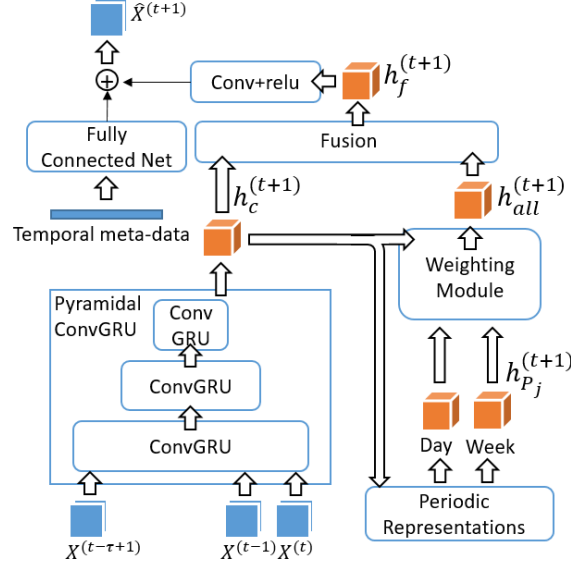


Figure 4.2: Weighting based fusion model architecture

from lower layers are concatenated before being passed to upper layer. This modification reduces the sequence length of the upper layer by a factor of 2, which results in significant speedup without negatively affecting performance. Formally the input and hidden state in layer $i > 0$ are derived as follows:

$$X_i^{(t)} = [h_{i-1}^{(2t)}, h_{i-1}^{(2t+1)}], \quad (4.5)$$

$$h_i^{(t)} = f(X_i^{(t)}, h_i^{(t-1)}). \quad (4.6)$$

The inputs at layer $i = 0$ ($X_0^{(t)}$) are the raw spatial images and the function $f(\cdot)$ refers to the set of operations in Equation 4.4.

4.2.2 Periodic Representations

Recurrent neural networks generate a hidden state vector at each cell, forget unnecessary information and update them accordingly by recurrently processing their inputs and previous states (Equation 4.6). Consequently, at the last cell, the hidden vector (tensor) preserves the sequential information of input in a dense representation. We aim to reuse these representation tensors for each period by introducing a loop-back mechanism in our learning procedure.

The key idea in this model is to dynamically maintain memory-based dictionary of periodic representations learned by the stacked pyramidal ConvGRUs, by implementing the two following procedures: (1) Loading periodic representations for current time point t from all available dictionaries, and (2) saving representation of the current time point in the dictionaries after the

forward pass through stacked ConvGRUs. We denote the current representation tensor at the last ConvGRU layer as $h_c^{(t)}$, where t refers to the data point as in $X^{(t)}$.

Definition 1: For a spatio-temporal time-series data X , let P_j be a periodic pattern, i.e. a set of periodic representations with the same scale, e.g. P_1 :daily pattern, P_2 :weekly pattern, etc. We define l_{P_j} as the size of the set P_j i.e. the number of discrete time intervals within the period scale of P_j . For instance, $l_{P_{daily}} = 48$ when the time interval duration is 30 minutes.

We build a dictionary of periodic representations for each P_j . When predicting $X^{(t+1)}$, the pyramidal ConvGRU outputs a current representation tensor $h_c^{(t+1)}$, given the input sequence $(X^{(t-\tau+1)}, \dots, X^{(t)})$. This representation tensor is stored in a key-value dictionary data structure $\mathcal{D}_j = (\mathcal{K}_j, \mathcal{V}_j)$, where $\mathcal{K}_j$ is the set of keys, and $\mathcal{V}_j$ is the set of periodic representation tensors h_{P_j} , for period P_j . For each periodic pattern P_j , the keys $\mathcal{K}_j$ of $\mathcal{D}_j$ are generated using a bucketing function defined as follows:

Definition 2: We define bucketing function $b_j(t)$ to map the time point t into bucket k , such that $\{k \in \mathbb{N} | k \leq l_{P_j}\}$.

For instance, assuming the interval is given in minutes, and 60 is divisible by the interval duration, we define bucketing function $b_{daily}(t)$ as follows:

$$b_{daily}(t) = t.hour * (60/interval) + \lfloor t.minute/interval \rfloor + 1. \quad (4.7)$$

We propose 3 different approaches for retrieving and updating periodic representations stored in $\mathcal{V}_j$. (1) Using sequential periodic representation, (2) Using estimated average of periodic representations and (3) Using temporally ordered representations. The representations are initialized to zero, to represent lack of knowledge from periodic patterns at early stages.

4.2.2.1 Sequentially Ordered Representation

Assuming $X^{(t+1)}$ is the data point to be predicted, representation $h_{P_j}^{(t+1)}$ is loaded from $\mathcal{D}_j$ as shown in Equation 4.8. Here, the bucketing function b_j , outputs the lookup key for dictionary $\mathcal{D}_j$. In the saving procedure, the current representation output $h_c^{(t+1)}$ generated from ConvGRU, replaces the previously stored representation in $\mathcal{D}_j$ as shown in Equation 4.9. This procedure is performed sequentially on training and testing data, however since the data might be randomly shuffled prior to training, the periodic patterns are not temporally ordered necessarily.

$$h_{P_j}^{(t+1)} \leftarrow \mathcal{D}_j[b_j(t+1)] \quad (4.8)$$

$$\mathcal{D}_j[b_j(t+1)] \leftarrow h_c^{(t+1)} \quad (4.9)$$

4.2.2.2 Estimated Average Representations

In this approach, instead of the last periodic representation, a set of last n representations are stored in a FIFO queue. Here $\mathcal{D}_j[b_j(t+1)]$ retrieves the queue of representation tensors for period P_j and the average of all tensors is provided to the model as shown in Equation 4.10.

In saving phase, the current representation $h_c^{(t+1)}$ replaces the oldest representation tensor in $\mathcal{D}_j[b_j(t+1)]$, as shown in Equation 4.11.

$$h_{P_j}^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \mathcal{D}_j[b_j(t+1)]_i \quad (4.10)$$

$$\mathcal{D}_j[b_j(t+1)].enqueue(h_c^{(t+1)}) \quad (4.11)$$

4.2.2.3 Temporally Ordered Representations

In this approach, we keep a list of representation tensors in the exact temporal order for each periodic pattern P_j . Using the lists, we can identify the representation tensor of the previous period for the time point t ; for instance, the representation tensors of $(t - l_{P_1})$ and $(t - l_{P_2})$ indicate the daily and weekly representations for t , respectively.

For this purpose, we use exact time-stamps as the keys for dictionary $\mathcal{D}_j$ instead of using the bucketing functions b_j . The retrieving and saving procedures are shown in Equation 4.12 and 4.13 respectively.

$$h_{P_j}^{(t+1)} \leftarrow \mathcal{D}_j[(t+1) - l_{P_j}] \quad (4.12)$$

$$\mathcal{D}_j[(t+1) - l_{P_j}] \leftarrow h_c^{(t+1)} \quad (4.13)$$

4.2.3 Fusion Process

In order to effectively make use of periodic representations in the model, they should be merged with current representation tensor $h_c^{(t+1)}$ generated by ConvGRU.

Element-Wise Fusion: Following the procedure outlined in [ZZQ17], we first introduce element-wise fusion where current representation tensor and periodic representations are merged in the following way:

$$h_f^{(t+1)} = \text{relu}(W_c \circ h_c^{(t+1)} + \sum_j (W_{P_j} \circ h_{P_j}^{(t+1)})). \quad (4.14)$$

Where W_c and W_{P_j} are learnable weight tensors and $(\circ)$ denotes element-wise multiplication that adjusts the importance of current and periodic representations.

The element-wise fusion method requires learning separate weight tensors with large number of parameters for each periodic pattern. Moreover, they consider all periods equally important which is not the case in practice. In order to estimate relevance of each periodic representation to the current state, we propose a weighting based fusion method inspired by attention mechanism in encoder-decoder networks [BCB14].

Weighting Based Fusion: In this method, we consider current representation tensor h_c as the context, and during training, we learn scalar weights a_j for each representation tensor h_{P_j} ,

which indicates importance and relevance of that periodic representation with respect to current context h_c . Through softmax operation, we require that $a_j \in [0, 1]$ and $\sum_j a_j = 1$. The final context tensor is derived by the weighted sum of all representation tensors by their importance factor. This procedure is shown in Equations 4.15-4.18 as follows:

$$e_j^{(t+1)} = W_f.flatten(W_e * [h_c^{(t+1)}, h_{P_j}^{(t+1)}]) + b_f, \quad (4.15)$$

$$a_j^{(t+1)} = \frac{\exp(e_j^{(t+1)})}{\sum_{j'} \exp(e_{j'}^{(t+1)})}, \quad (4.16)$$

$$h_{all}^{(t+1)} = \sum_j a_j^{(t+1)} h_{P_j}^{(t+1)}, \quad (4.17)$$

$$h_f^{(t+1)} = \text{relu}(W_c \circ h_c^{(t+1)} + W_P \circ h_{all}^{(t+1)}). \quad (4.18)$$

In Equation 4.15, W_e is a $(1 * 1)$ convolutional kernel that acts as a feature pooling mechanism in filter space [LCY14]. A single tensor h_{all} represents all periodic tensors weighted by their importance.

After the fusion process, a convolution layer transforms h_f into the same shape as input spatial image i.e. $\mathbb{R}^{M*W*H}$. In Equation 4.19, W_f is a $W \times H$ convolutional kernel with M number of filters.

$$h_f^{(t+1)} = \text{relu}(h_f^{(t+1)} * W_f) \quad (4.19)$$

metadata Fusion: In spatio-temporal applications, time-stamps of records are available and could be used to provide additional information to the model. For this purpose, temporal information, such as time of day and day of week are one-hot encoded, concatenated and passed to two fully connected layers. After reshaping the output, they are merged with $h_f^{(t+1)}$ to create spatial image $\hat{X}^{(t+1)}$, as shown in Equations 4.20, 4.21.

$$h_{meta}^{(t+1)} = \text{relu}(\text{relu}(X_{meta}^{(t+1)} \cdot W_{fc1} + b_{fc1}) \cdot W_{fc2} + b_{fc2}) \quad (4.20)$$

$$\hat{X}^{(t+1)} = \text{relu}(h_f^{(t+1)} + h_{meta}^{(t+1)}) \quad (4.21)$$

4.3 Implementation and Optimization

Stochastic gradient descent (SGD) method, using back propagation through time (BPTT) was used to optimize the model. The least-square error term introduced in Equation 4.2 was used as the objective function to be minimized. To ensure proportional influence of all pixels, spatial images were down-scaled using min-max normalization, mapping all records into range $[0, 1]$.

Using grid search method, three ConvGRU layers were used in hierarchical architecture with 32, 64 and 96 number of filters respectively. The input sequence length in the lowest

ConvGRU layer was set to 28 time steps. Tensorflow library was used as the optimization framework, and the model was trained on a server with 16GB GPU memory. The learning rate was set to $1e^{-4}$ and the model was trained for 500 epochs, with early stopping mechanism.

The algorithms for training *PCRN* model is shown in Algorithm 4. In this Algorithm, notation $\mathcal{M}$ refers to mini-batch sets and *optimize()* function refers to a gradient descent step using back-propagation.

Algorithm 4: *PCRN* training algorithm

Input: Normalized spatial images $X : \{X^{(0)}, \dots, X^{(T)}\}$
 Timestamps $ts : \{ts^{(0)}, \dots, ts^{(T)}\}$, sequence length: τ
 Dictionary $\mathcal{D}_j \leftarrow \emptyset$ for each periodic pattern P_j

Output: Learned *PCRN* parameters

- 1: $X_{img}, X_p, X_{ts} \leftarrow \emptyset$
- 2: **for** $t = \tau$ to T **do**
- 3: add $[X^{(t-\tau)}, \dots, X^{(t-1)}]$ to X_{img}
 add $X^{(t)}$ to X_p , add $ts^{(t)}$ to X_{ts}
- 4: **end for**
- 5: $X_{train}, X_{test} \leftarrow split_test_train(X_{img}, X_p, X_{ts})$
- 6: initialize model parameters
- 7: **for** $iter = 1$ to N **do**
- 8: select random mini-batch $\mathcal{M}_{\mathcal{X}}$ from X_{train}
- 9: **for** each periodic pattern P_j **do**
- 10: $\mathcal{M}_{h_{P_j}} \leftarrow$ Load Periodic representation for $\mathcal{M}_{\mathcal{X}}$ from $\mathcal{D}_j$
- 11: **end for**
- 12: $loss, \mathcal{M}_{h_c} \leftarrow optimize(\mathcal{M}_{\mathcal{X}}, \mathcal{M}_{h_P})$
- 13: **for** each periodic pattern P_j **do**
- 14: $\mathcal{D}_j \leftarrow$ Save current periodic representation $\mathcal{M}_{h_c}$
- 15: **end for**
- 16: **end for**

4.4 Experiments

4.4.1 Experimental Settings

In this section, the datasets, various baselines and the evaluation metric used in the experiments are introduced.

4.4.1.1 Datasets

In this problem we are interested in predicting the next states of a geo-spatial domain and detecting abnormal pattern considering the recurring periodic patterns that occur in such data.

Dataset	Type	Period (months)	Interval (mins)	Total Images	Test days
TaxiBJ	Taxi Flow	18	30	21360	28
TaxiSG	Taxi Density	4	5	34560	12

Table 4.1: The characteristic of two benchmark datasets

The following two datasets were used for evaluation.

Singapore taxi trajectories (TaxiSG): This dataset contains trajectory log records generated from taxis in Singapore in period of January to April 2015. The CBD area in downtown Singapore were partitioned into a 32×32 grid map and taxi log records were aggregated in each cell for intervals of 5 minutes. Density of all taxis and taxis with passengers on board were used to create set of $X^{(t)} \in \mathbb{R}^{2 \times 32 \times 32}$ spatial images according to Equation 4.3. Last 12 day's data were used for validation and testing.

Beijing taxi trajectories (TaxiBJ): The dataset contains in-flow and out-flow of taxis in different regions in Beijing in four separate time intervals: July-Oct. 2013, March-June 2014, March-June 2015 and Nov. 2015-April 2016. Beijing is partitioned into 32×32 grids, and for each grid cell, in-flow and out-flow of vehicles in intervals of 30 minutes are calculated using the procedure explained in [ZZQ17], resulting in set of spatial images $X^{(t)} \in \mathbb{R}^{2 \times 32 \times 32}$.

The two datasets span over several months and show recurring periodic patterns in crowd density data in different regions, which make them suitable for evaluation of the proposed models. However, they are different in terms of spatial and temporal granularity. The TaxiBJ dataset covers a larger area (Beijing city), while TaxiSG dataset covers downtown Singapore. Therefore, it is important for the models to capture spatial correlations in both settings effectively. The time period between consecutive spatial images in TaxiBJ dataset is 30 minutes while in TaxiSG this period is 5 minutes. Therefore, in both cases, the model needs to capture temporal correlation between the sequence of spatial images well, through the recurrent neural network architecture.

4.4.1.2 Baseline Methods

The following 6 baselines are used in our experiments:

- Snaive: Seasonal naive model, which sets the predicted value equal to the last observed seasonal pattern, where the season duration is set as week.
- HA: Historical average over observed weekly patterns.
- ARIMA: ARIMA is a powerful linear model to fit and forecast time-series data.
- VAR: Vector Autoregressive model that captures linear interdependencies between multiple univariate time-series for modeling and forecasting multivariate time-series data [KP03].

- ConvGRU: Convolutional GRU architecture introduced in [BYPC16] modified to have pyramidal architecture but without fusion with periodic representations.
- ST-ResNet: A deep learning model for crowd flow prediction by learning deep convolutional neural networks using residual learning [ZZQ17]. Learned representations are merged in a fusion process along with external information such as weather condition and holidays. Raw inputs from previous periods need to be provided during runtime.

Snaive and HA are two simple heuristic methods for time-series prediction. In addition, we included powerful linear and non-linear forecasting methods as baselines for the experiments. ARIMA is one of the most powerful linear time-series forecasting models, which is composed of Autoregressive and Moving average components to effectively forecast short-term future values of time-series. VAR is an extension of ARIMA, where the input is a vector instead of a univariate time-series, and dependencies among different vector elements are captured through introduction of extra parameters in the model.

Convolutional GRU [BYPC16] is a recurrent neural network based architecture, which extracts features from both spatial and temporal dimensions of the data. This model is modified to have pyramidal architecture similar to ours and used as a baseline. It is important to note that, this model does not use periodic patterns in the prediction process. Finally, ST-ResNet [ZZQ17] is the state-of-the-art crowd density prediction model, which uses deep convolutional neural network architecture to predict the crowd density state in the next interval. This powerful non-linear forecasting model is used as another baseline in this study to evaluate the performance of the proposed models.

4.4.1.3 Evaluation Metric

We employ the widely used root mean squared error $rmse(X, \hat{X}) = \sqrt{\frac{1}{T} \sum_{t=1}^T (X_t - \hat{X}_t)^2}$ as the evaluation metric for similarity between predictions and ground-truth $\hat{X}, X \in \mathbb{R}^{M*W*H}$.

4.4.2 Next Frame Prediction

First, we compare 6 baselines with 3 different variants of our model, namely (1) sequentially ordered periodic patterns $PCRN_{seq}$, (2) moving average patterns $PCRN_{avg}$, and (3) temporally ordered periodic representations $PCRN_{temp}$, for crowd density prediction at the next interval. The results for prediction performance from baselines and the best performing variant of our model $PCRN_{temp}$ are shown in Table 4.2. Each entry in the Table shows RMSE score obtained by the particular method on the test set of a particular dataset. Note that a lower value of RMSE means a better performance.

For parametric models such as ARIMA and VAR, best set of parameters were found for each time-series using grid search and Akaike Information Criterion (AIC). For ST-ResNet we used their best performing variant, with 12 residual units. Please note that, unlike our model, ST-ResNet uses additional raw input from previous periods on the runtime. As shown in Table

Method	Taxi BJ	Taxi SG
Snaive	44.83	6.946
HA	40.796	8.685
ARIMA	22.44	4.600
VAR	22.05	4.56
ST-ResNet	16.69	4.52
ConvGRU	17.35	4.341
$PCRN_{temp}$	15.85	4.252

Table 4.2: Next step prediction results

4.2, generally non-linear models outperform linear models. We observe that our proposed $PCRN_{temp}$ (indeed all the variants of our model) outperforms 6 baselines consistently.

Results obtained from ConvGRU model, which has the same architecture as our model but without merging with periodic representations, demonstrate that convolutional RNN architecture is effective for modeling spatio-temporal time-series data as it outperforms other linear and non-linear models, though not consistently outperforming ST-ResNet. Comparing results obtained from PCRN models with ConvGRU shows the effectiveness of using periodic representations in our model.

4.4.3 Representation Update Method and Fusion Mechanism

To compare different periodic representation update methods, we compare three different variants of our model, namely, $PCRN_{seq}$, $PCRN_{avg}$ and $PCRN_{temp}$. The results in Table 4.3 show the superiority of $PCRN_{temp}$ over $PCRN_{seq}$ and $PCRN_{avg}$ which can be attributed to leveraging the exact temporal order of time point representations.

We also compare the proposed weighting based fusion mechanism with element-wise fusion method used in previous work [ZZQ17]. The results for next frame prediction in table 4.3 show that our weighting-based fusion method is able to take relevance of each periodic pattern into account, and in fact, shows better performance in almost all cases in our experiments. Moreover, the number of parameters are significantly reduced using this approach, by learning a single weight tensor for aggregated representations.

4.4.4 Multi-Step Ahead Prediction

PCRN can perform multi-step ahead prediction, by appending the output for a time point t to the input for $t + 1$, and running the model for an arbitrary number of steps. Our model does not use raw input from previous periods and this would give us advantage over other baselines that use periodic information in their models. For instance, ST-ResNet [ZZQ17], which is the only model using periodic information, learns its model by providing the raw data of previous

Fusion	params	Method	TaxiBJ	TaxiSG
Weighting-based	198,848	$PCRN_{seq}$	16.06	4.262
		$PCRN_{avg}$	16.74	4.265
		$PCRN_{temp}$	15.85	4.252
Element-wise	294,912	$PCRN_{seq}$	16.16	4.354
		$PCRN_{avg}$	16.48	4.271
		$PCRN_{temp}$	16.01	4.242

Table 4.3: Comparison over different PCRN variants and fusion types

periods as input during training and performs one step prediction. These inputs are required to be provided during test time as well.

We compare various methods for predicting next 6 steps ahead. The results in Figure 4.3, show that RMSE of ARIMA and VAR linearly grows as time step increases. PCRN shows lowest rate of error increase among all, and consistently outperforms deep learning based ST-ResNet, over all steps especially further steps ahead. This can be attributed to better capturing of temporal correlations through ConvGRUs, and using periodic representations instead of raw inputs from previous periods in our model.

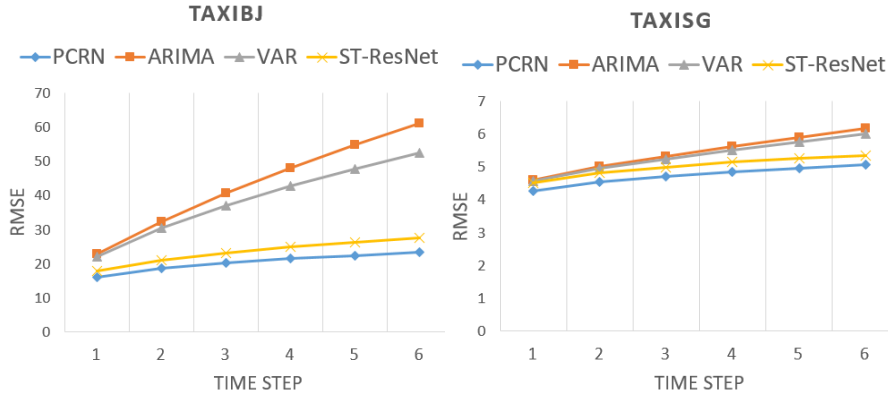


Figure 4.3: Comparison of $PCRN_{seq}$ with ARIMA, VAR and ST-ResNet on multi-step prediction errors up to 6 steps ahead.

We also do an experiment on multi-step ahead prediction with different interval durations. As the prediction interval duration grows, the size of the training set reduces, making it harder for deep models with a large number of parameters to get optimized. To measure the effect of interval length in practice, we perform an experiment on multi-step ahead prediction using variable interval lengths, to predict up to fixed time point ahead on TaxiSG dataset. As illustrated in Figure 4.4, 6 steps ahead prediction with 5-min interval on TaxiSG dataset has lower error than one step prediction with 30-min interval. This can be attributed to lower training instances in 30-min case. This shows the improvement of results when multi-step ahead prediction with

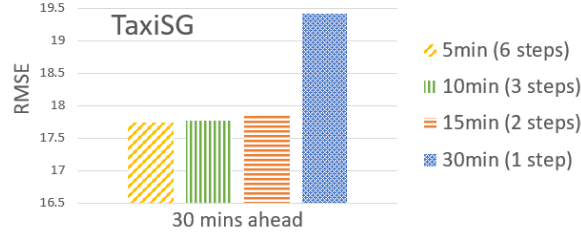


Figure 4.4: Multi-step and one step ahead prediction errors with variable interval lengths

a shorter interval is used. The number of images in each training set for each interval length is shown in Table 4.4.

Interval	5mins	10mins	15mins	30mins
Images	31104	15552	10368	5184

Table 4.4: Number of training images for each interval length for TaxiSG dataset

4.5 Summary

In this chapter, we proposed an optimization based model for crowd density prediction, using pyramidal convolutional recurrent network architecture. By leveraging recurring periodic patterns observed in geo-spatio-temporal domains, and dynamically incorporating them in our prediction model with a loop-back mechanism, our model outperforms other linear and non-linear baselines in the next step and multi-step predictions. Moreover, by introducing a weighting based fusion mechanism, the importance and relevance of periodic representation were taken into account.

Chapter 5

Temporal Anomaly Detection in Geo-Spatial Data

Detection of outliers and anomaly patterns in temporal settings is crucial for a wide range of applications and domains. In geo-spatio-temporal settings, detection of abnormal crowd density patterns in real-time is beneficial for domains such as traffic management, telecommunication industry and urban planning. For instance, detection of abnormal traffic patterns and regulating the traffic in the shortest possible time is important for traffic management. In telecommunication industry, detection of abnormal crowd density patterns of mobile users is crucial for balancing the network coverage in different regions.

In this chapter, a temporal anomaly detection model is proposed to identify abnormal crowd density patterns in geo-spatial settings. The chapter is organized as follows. In Section 5.1, the problem is introduced and the problem setting is defined. In Section 5.2, the anomaly detection model is proposed. In Section 5.3, implementation and optimization procedures are explained and in Section 5.4, the experimental study is reported. Finally, Section 5.5 summarizes the chapter.

5.1 Problem Statement

In geo-spatial anomaly detection problem, we are aiming to detect abnormal crowd density patterns in a pre-defined spatial area. We define disjoint time periods of fixed interval (τ ; e.g. 5 minutes) as instances for this study. We aim at detecting abnormal instances in crowd density data of a pre-defined spatial area, assuming that the majority of the data have normal patterns. We partition the spatial area into a grid structure and, for each time period instance, aggregate the observations within each partition of the area and produce 2 dimensional aggregated data (referred to as ‘spatial image’). The number of observations within a region is set as the pixel density value of the 2-D spatial image. Some examples of the spatial images are illustrated in Figure 5.1. Furthermore, to capture temporal dynamics in the data, M consecutive spatial images are concatenated together to create $X^{(t)} \in \mathbb{R}^{M*W*H}$, referred to as ‘spatial cuboid’, where W and H indicate the width and height of the grid structure. A set of spatial cuboids

$\{X^{(1)}, X^{(2)}, \dots, X^{(T)}\}$ are given as input to our model.

We assume that the temporal information of each spatial image is available and consider the time-stamp of the last spatial image in a spatial cuboid $X^{(t)}$ as the time-stamp of the cuboid, $ts^{(t)}$. A spatial cuboid is generated for each spatial image, by concatenating its $M - 1$ previous spatial images, except the first $M - 1$ images. The goal of the study is to identify whether the time-stamp of each cuboid is an abnormal instance or not. We have collected ground-truth anomalies for two datasets, designated $\{ts^{(a_1)}, \dots, ts^{(a_n)}\}$ where a_i indexes an anomaly. We compare the results of our methods against the ground-truths for evaluation.

The temporal information of spatial images are normalized for the prediction of the temporal metadata of spatial cuboids in two ways as follows: 1) Divide a day into $\frac{60*24}{\tau}$ number of intervals and let $n_{ts}^{(t)}$ indicate the index of the interval corresponding to $ts^{(t)}$ ($0 \leq n_{ts}^{(t)} < \frac{60*24}{\tau}$) and 2) let $w_{ts}^{(t)}$ represent whether $ts^{(t)}$ occurred during a weekend or not. These two normalizations support 1) identification of a cuboid into one of the $\frac{60*24}{\tau}$ intervals and 2) identifying whether the time-stamp of a cuboid belongs to a weekend or not. The tuple $\langle n_{ts}^{(t)}, w_{ts}^{(t)} \rangle$ is used for formulating the temporal anomaly detection model.

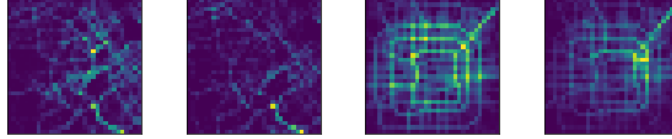


Figure 5.1: Examples of spatial images from Singapore and Beijing taxi dataset. Brighter pixels represent higher crowd density.

5.2 Geo-Spatio-Temporal Anomaly Detection (GSTAD) Model

Our proposed model consists of three components: Encoder network, decoder network and temporal metadata prediction network. The encoder and decoder networks form an auto-encoder, which reconstructs a spatial cuboid from its dense representation, in order to learn representative features from spatial cuboids. The dense representation generated from the encoder network is used to predict the temporal metadata of the spatial cuboid, which may help capture temporal anomalies. Figure 5.2 depicts the overall architecture of the proposed model.

We combine the approximation functions of the reconstruction network (encoder and the decoder) with the prediction network, sharing the parameters of the encoder network, as follows:

$$\hat{X}^{(t)}, \hat{X}_{meta}^{(t)} = f(X^{(t)}; \theta), \quad (5.1)$$

where f is the combined approximation function, θ indicates the parameters of the networks, $\hat{X}^{(t)}$ represents the reconstruction of $X^{(t)}$, and $\hat{X}_{meta}^{(t)}$ represents the predicted metadata of $X^{(t)}$. The approximation functions of the two networks (f_{rec} , f_{pred}) are as follows:

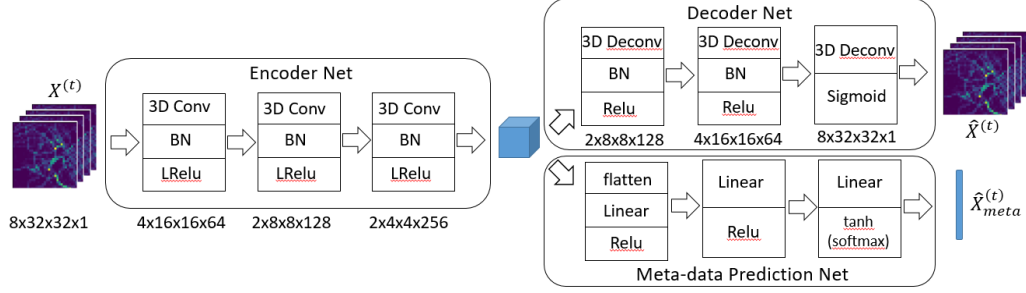


Figure 5.2: Proposed model architecture

$$\hat{X}^{(t)} = f_{rec}(X^{(t)}; \theta_{rec}), \quad (5.2)$$

$$\hat{X}_{meta}^{(t)} = f_{pred}(X^{(t)}; \theta_{pred}), \quad (5.3)$$

where θ_{rec} and θ_{pred} indicate the parameters of the two networks, respectively.

To optimize the model parameters, the reconstruction and the prediction loss functions are jointly minimized as follows:

$$\min_{\theta} \mathcal{L}_{rec} + \mathcal{L}_{pred} + \lambda \|\theta_{pred}\|_2^2, \quad (5.4)$$

where $\mathcal{L}_{rec}$ and $\mathcal{L}_{pred}$ are the loss functions of reconstruction and temporal metadata prediction, and λ is the coefficient that controls the regularization term for parameters of the prediction network. Adding the regularization term helps prevent over-fitting to the training data.

5.2.1 3D Convolutional Autoencoder

Autoencoders are unsupervised methods for representations learning in which the model reconstructs the input data in the output layer, while the data passes through a neural bottleneck layer. This allows the model to learn useful representations of the data in the bottleneck layer. In the first step, we use this neural network architecture to learn representative features from spatial cuboids, which are used for temporal metadata prediction later. Mean Squared Error (MSE) is used as the reconstruction loss function:

$$\mathcal{L}_{rec} = \frac{1}{T} \sum_{t=1}^T \|X^{(t)} - \hat{X}^{(t)}\|_2^2. \quad (5.5)$$

To minimize the reconstruction loss, convolutional autoencoder network is used, as they have shown to be effective in modeling dependencies across dimensions. In particular, we use 3D ConvNets, which are special type of convolutional networks, where the kernels are 3D instead of 2D. This allows the model to extract features along one extra dimension, where the input is volumetric. They are thus suitable for modeling both spatial and temporal dependencies in our datasets, where the 2D spatial images are concatenated across the time dimension. Formally, for 3D input tensor f and convolutional kernel g , 3D convolution in discrete settings is defined as follows:

$$(f * g)[x, y, t] = \sum_{\tau_x=1}^{n_x} \sum_{\tau_y=1}^{n_y} \sum_{\tau_t=1}^{n_t} f[\tau_x, \tau_y, \tau_t] \cdot g[x - \tau_x, y - \tau_y, t - \tau_t]. \quad (5.6)$$

We loosely follow the architecture proposed in [ZDS⁺17], to construct the 3D convolutional autoencoder. Specifically, we use a single channel for each spatial image, and 8 consecutive spatial images ($M=8$) of size $32*32$ ($W=H=32$) are concatenated together. Convolutional kernels of size $3*3*3$ are used based on the recommended architecture for learning spatio-temporal features [TBF⁺15].

Strided convolutions in both spatial and temporal dimensions are used in the encoder and fractional-strided convolutions (transposed convolution) are used in the decoder, following the proposed guidelines in [RMC15]. Batch normalization (BN) is used in both encoder and decoder (except the last layer of decoder) to stabilize the learning process. Leaky relu (LRelu) is used as non-linearity function in the encoder layers and relu is used in the decoder layers. Sigmoid function is applied to the output of the last layer of decoder to squash the output values between range [0 1].

5.2.2 Temporal metadata prediction

The time-series data in some geo-spatio-temporal settings such as traffic or crowd density in urban areas show recurring periodical patterns over different time scales. Since the temporal metadata (time of day, day of week, etc.) are available for each data point in temporal settings, we use an optimization-based approach to predict the temporal metadata of a spatial cuboid. If an observation conforms to the historical patterns previously observed at a time point, the model will be able to predict its temporal metadata accurately; otherwise, it would be identified as an anomaly with respect to the other patterns observed at that time point.

The metadata prediction model is a feed-forward neural network, consisting of two hidden layers with relu nonlinearity and an output layer to predict the metadata vector $\hat{X}_{meta}^{(t)}$. Hyperbolic tangent (tanh) or softmax function is used in the final layer, depending on whether classification or regression is performed. We introduce two approaches for the temporal metadata prediction using classification based approach (GSTAD-Cls) and regression based approach (GSTAD-Reg).

5.2.2.1 Classification-based Method (GSTAD-Cls)

In the classification-based approach, the prediction model aims to classify a spatial cuboid into a discrete time interval. Assuming L is the interval length variable (in hours, divisible by 24) for partitioning the time of day into separate time slots, $(24/L)$ non-overlapping time-slots would exist during 24 hours of a day. Note that the interval length L is used to define the classes in classification-based approach, and is different from the interval length τ between consecutive spatial images $X^{(t)}$ and $X^{(t+1)}$.

The traffic pattern at different times of a weekday are quite different from those of a weekend. Therefore, separate classes for time-slots in weekdays and weekends need to be specified. The temporal metadata for $X^{(t)}$ is defined as a one-hot encoded vector $X_{meta}^{(t)} \in \mathbb{R}^{2*(24/L)}$, which represents every non-overlapping time slot of length L during weekdays and weekend ($24/L$ classes for weekdays and $24/L$ classes for weekends). Therefore, in classification-based approach, $X_{meta}^{(t)}$ is defined as follows:

$$\begin{aligned} X_{meta}^{(t)} &\leftarrow \vec{0} \in \mathbb{Z}^{2*(24/L)}, \\ u &= \frac{w_{ts}^{(t)} * 24}{L} + \lfloor \frac{n_{ts}^{(t)} * \tau}{L * 60} \rfloor, \\ X_{meta}^{(t)}[u] &= 1. \end{aligned} \tag{5.7}$$

In Equation 5.7, u identifies a unique integer corresponding to the time-slot class of $X^{(t)}$. $w_{ts}^{(t)}$ is the weekend binary variable, and $n_{ts}^{(t)}$ is the discrete interval number as defined in section 5.1. After defining the one-hot encoded vectors for $X_{meta}^{(t)}$, the model classifies the input $X^{(t)}$, by minimizing the cross-entropy loss function.

$$\mathcal{L}_{pred} = H(X_{meta}^{(t)}, \hat{X}_{meta}^{(t)}) = - \sum_i (X_{meta}^{(t)})_i \log(\sigma(\hat{X}_{meta}^{(t)})_i), \tag{5.8}$$

where σ is the softmax function which transforms the output vector $\hat{X}_{meta}^{(t)}$ into a probability distribution. Finally, assuming that c is the index of the ground-truth class, and p is a probability threshold in range $[0,1]$, the observation $X^{(t')}$ is flagged as temporal anomaly if $(\hat{X}_{meta}^{(t')})_c < p$.

5.2.2.2 Regression-based Method (GSTAD-Reg)

The classification-based approach for temporal metadata prediction partitions a day into discrete time intervals. The problem with this approach is that the temporal continuity is not considered between the time slots. For instance, if the partitioning is done for one hour intervals, a spatial cuboid generated at 3:55 pm can possibly get easily mis-classified into [4 pm, 5 pm] interval instead of [3 pm, 4 pm]. This would increase the false alarm rate for temporal anomalies.

To address the issue, a regression based approach is proposed which considers temporal continuity by assigning real-valued data points in the Euclidean space to each time point. A geometric mapping function is defined such that data points that are closer to each other in temporal domain get mapped closer to each other in the Euclidean space as well. Moreover, data points that are generated during weekdays and weekends should get mapped separately from each other.

To achieve this, temporal metadata of a data point $X^{(t)}$ is defined as a 3-dimensional vector, $X_{meta}^{(t)} \in \mathbb{R}^3$. The coordinates for this vector are assigned as follows:

$$\begin{aligned} X_{meta}^{(t)} &= \langle x^{(t)}, y^{(t)}, z^{(t)} \rangle, \\ x^{(t)} &= \cos(n_{ts}^{(t)} * \frac{2\pi * \tau}{60 * 24}), \\ y^{(t)} &= \sin(n_{ts}^{(t)} * \frac{2\pi * \tau}{60 * 24}), \\ z^{(t)} &= w_{ts}^{(t)}. \end{aligned} \tag{5.9}$$

In this mapping process, the coordinates x, y represent the time information and map the points into a 2-dimensional circle with radius 1. Each point on the circle represents a time of day value similar to analog clock mechanism of 24 hours. The z value is a discrete variable representing weekday/weekend information for the data point in the third dimension. In Equation 5.9, τ is the interval length for spatial images, and n_{ts} and w_{ts} are the time interval identifier and binary weekday/weekend variables respectively as defined in section 5.1. A 3 dimensional representation of the regression space is illustrated in Figure 5.3.

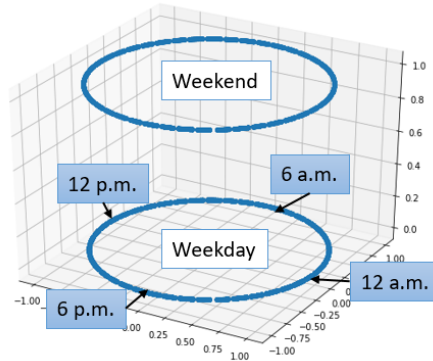


Figure 5.3: 3D visualization of the regression space for temporal metadata prediction. The 2 dimensional circles in x,y axis represent the continuous variable for time of the day and the binary value in z-axis represents weekday/weekend variable.

After defining the coordinates for $X_{meta}^{(t)}$, the prediction task is reduced to regressing the coordinate values, given the spatial cuboid as input. We define the loss function $\mathcal{L}_{pred}$ for

the regression problem as the mean square loss between the ground-truth and predicted values. Since the coordinates are defined on the circumference of the circles as illustrated in Figure 5.3, a least square penalty term is added to the prediction loss function $\mathcal{L}_{pred}$, to penalize the points which are regressed far from the circumference of circles with unit radius in x, y dimensions. The prediction loss function for the regression-based approach is defined as follows:

$$\mathcal{L}_{pred} = \frac{1}{T} \left(\sum_{t=1}^T \| \hat{X}_{meta}^{(t)} - X_{meta}^{(t)} \|_2^2 + \gamma \sum_{t=1}^T \left((\hat{X}_{meta,1}^{(t)})^2 + (\hat{X}_{meta,2}^{(t)})^2 - 1 \right)^2 \right). \quad (5.10)$$

In Equation 5.10, γ is a user-defined coefficient to control the penalization term. Moreover, $\hat{X}_{meta,1}^{(t)}$ and $\hat{X}_{meta,2}^{(t)}$ represent x and y dimensions in Equation 5.9 respectively.

A threshold-based approach is used to detect anomaly observations. The threshold value is set as the Euclidean distance between two points on the circumference of the 2D circles in the regression space with their difference set as the time interval between the two points on the circle. Let q be the discrete number of time intervals (with length τ) between two points. Formally, the threshold value d_q is calculated in the following way:

$$d_q = \sin(q * \frac{2\pi * \tau}{60 * 24})^2 + (1 - \cos(q * \frac{2\pi * \tau}{60 * 24}))^2 = 2(1 - \cos(q * \frac{2\pi * \tau}{60 * 24})). \quad (5.11)$$

Based on the defined threshold d_q , the test observation $X^{(t')}$ is flagged as temporal anomaly if $\| \hat{X}_{meta}^{(t')} - X_{meta}^{(t')} \|_2^2 > d_q$.

5.2.3 Identification of Abnormal Regions

It is important for a spatio-temporal anomaly detection model to identify the timing of an abnormal event, as well as the spatial areas which are causing the abnormalities to occur. In the case of global anomaly detection, the pixel-wise reconstruction error could be visualized to illustrate the most irregular areas that the model is unable to reconstruct. However, in the temporal anomaly detection case, the pixel-wise error is not available. Therefore, another approach should be used to identify the abnormal regions.

For this purpose, we maintain a dictionary of statistical properties (mean and standard deviation) of time-series data for every region from the training data.

$$\mathcal{D}_\mu[i, j] = \frac{1}{T} \sum_{t=1}^T X_{i,j}^{(t)}, \quad (5.12)$$

$$\mathcal{D}_\sigma[i, j] = \sqrt{\frac{\sum_{i=1}^T (X_{i,j}^{(t)} - \mathcal{D}_\mu[i, j])^2}{T}}. \quad (5.13)$$

During test time, for an anomalous observation $\hat{X}^{(t)}$, each pixel value $\hat{X}_{i,j}^{(t)}$ is z-normalized using the stored statistical values for that specific region. After normalizing all the pixel values to zero mean and unit standard deviation, a heatmap of the regions based on their distance from the mean values is created to visualize the abnormal regions.

$$z_{i,j} = \frac{\hat{X}^{(t)} - \mathcal{D}_\mu[i, j]}{\mathcal{D}_\sigma[i, j]}. \quad (5.14)$$

5.3 Implementation and Optimization

In order to optimize the objective function defined in Equation 5.4, back-propagation algorithm using stochastic gradient descent (SGD) method and Adam optimization algorithm [KB14] for controlling the learning rate were used. The learning rate was set to 10^{-3} and mini-batch size was set to 32. Tensorflow library was used as the optimization framework and the models were trained on a GPU server with 16GB GPU memory for 500 epoch with early stopping mechanism.

To ensure proportional influence of all pixels, spatial images were down-scaled using min-max normalization, mapping all records into range $[0, 1]$. For the classification/regression feed forward network, two hidden layers with 1000 neurons and relu non-linearity were used. In classification network, *softmax* function was applied to the output of the last layer to transform the output to a probability distribution. *Tanh* non-linearity was applied in the regression layer to squash the output in range $[-1, 1]$. The penalization parameters λ and γ were set to $1e-2$ and 1 using the grid search method.

5.4 Experiments

In the experimental studies, we evaluate the performance of the proposed models in detection of abnormal patterns in geo-spatial settings. In this section, after defining the experimental settings, the performance of the classification-based and regression-based models are compared with the baselines including some state-of-the-art spatio-temporal anomaly detection methods. In the next experiment, a model architecture analysis is performed. Finally, some examples of visualization of temporal anomalies are shown.

5.4.1 Experimental Settings

In this section, we describe the baselines and the datasets used in the experimental studies, the annotation process and the evaluation procedure.

5.4.1.1 Baselines

In the past few years, many studies have focused on anomaly detection problem in spatio-temporal settings such as videos. Such models mostly use reconstruction-based approach to detect abnormal patterns. We compare our proposed methods with the following baselines, including two state-of-the-art spatio-temporal anomaly detection models:

- **ARIMA:** Linear time-series forecasting model, which identifies anomalies by computing distance between predictions of actual values and groundtruth.
- **Temporal Linear Regression (TLR):** Linear regression model which predicts the temporal metadata of each region independently.
- **STAE [ZDS⁺17]:** A deep learning-based spatio-temporal anomaly detection model, which uses 3D convolutional autoencoder to learn normal pattern of data by reconstructing previous frames and predicting future frames. The model is re-implemented with slight modifications to accommodate the spatial cuboids used in this study.
- **2D ConvAE [HCN⁺16]:** 2-dimensional convolutional autoencoder model to detect irregularities in spatio-temporal data. The model performs 2D convolution on multi-channel input images which are created by concatenating consecutive frames along the feature axis. The model is re-implemented also with slight modifications to accommodate the spatial cuboids used in this study.
- **3D ConvAE:** 3-dimensional convolutional autoencoder architecture introduced in this study, without the metadata prediction network. The reconstruction loss was used to detect anomalies.

The ARIMA model is a powerful linear time-series forecasting model which predicts the expected values of a time-series based on the previous observed sequence in a region. If the next observed value is distant from the prediction output, the time point is suspected to be anomaly. The Temporal Linear Regression model (TLR) predict the temporal metadata of a given time-series data for each region similar to our GSTAD-Reg model mechanism, and detects anomalies based on the prediction error.

STAE [ZDS⁺17] and 2D ConvAE [HCN⁺16] are two state-of-the-art spatio-temporal anomaly detection models, that detect anomalies based on the reconstruction error. We use these baselines to evaluate performance of our models in detection of temporal anomalies in comparison with these methods. In addition, to better understand the importance of the metadata prediction network in temporal anomaly detection, we use 3D convolutional autoencode architecture introduced in this study (Figure 5.2), without the metadata prediction network as a baseline.

Dataset	Period (months)	Interval (mins)	Training images	Test images
TaxiBJ	18	30	15264	7220
TaxiSG	4	5	17280	17280

Table 5.1: The characteristic of datasets

5.4.1.2 Datasets

Since there has been limited studies on anomaly detection problem in crowd density data, to the best of our knowledge, there are no public anomaly detection benchmarks available for this task. Therefore, in order to evaluate our method, we use two taxi trajectories datasets from Singapore and Beijing, and create the spatial images representing crowd density for intervals of τ minutes for each dataset. The real world crowd density data captured from these two regions over a period of several months contain both global anomalies and temporal anomalies, which can be used for evaluation of the proposed models. Using publicly available prior knowledge from the time periods of the datasets (e.g. news articles), the time-stamps of real abnormal traffic conditions in those periods were identified and annotated as ground-truth for anomalies.

- **TaxiSG Dataset:** Taxi trajectory records from January to April 2015 were extracted from a 5km*5km area in downtown Singapore. The area was partitioned into a 32*32 grid map, and the aggregated numbers of taxis for intervals of 5 minutes ($\tau = 5$) were set as pixel values to create a sequence of *spatial images*. Two subsets of the dataset were created by separating even days from odd days, where even days dataset was used for training the models while the odd days dataset was used for evaluation.
- **TaxiBJ Dataset:** The dataset contains inflow of taxis into different regions of Beijing city in four non-overlapping intervals: July-Oct. 2013, March-June 2014, March-June 2015 and Nov. 2015-April 2016. The Beijing city was partitioned into a 32*32 grid map to create spatial images with $\tau = 30$ minutes. For this dataset, the period of Nov. 2015 to April 2016 was used for testing, and the rest of data were used for training.

The Two datasets defined above cover different sized regions with different interval value between spatial images ($\tau = 5$ for TaxiSG and $\tau = 30$ for TaxiBJ). Therefore, the proposed models need to capture spatial and temporal dependencies in both settings with different configurations effectively. In both datasets, 8 consecutive spatial images were concatenated together to create a spatial cuboid $X^{(t)} \in \mathbb{R}^{8 \times 32 \times 32}$ which were used as the input for the models. The detailed information about the datasets is given in Table 5.1

5.4.1.3 Ground-truth Annotation

Definitions of anomalies are generally subjective, and there is no clear guideline for identifying anomalies. In traffic domain, accidents, breakdown of public transportation systems,

special events such as parades and festivals or public holidays might cause anomaly patterns in traffic data. In our datasets, we identified the events and occasions which clearly affected the traffic patterns in the time periods of the test sets, and annotated the affected time-stamps as anomalies. The following categories were considered as anomalies in our taxi datasets:

- **Public holidays:** Studies have shown that the traffic density in public holidays are significantly different than regular days [CMW07]. Based on these observations, the time-stamps in public holidays were included in the anomaly set. In addition to public holidays, governments may issue warnings in some days and advise citizens to stay indoors due to issues such as air pollution or extreme weather conditions, which will affect the traffic condition. We identified the time-stamps for those time periods by referring to the official news sources, and included them in the ground-truth set ^{1,2}.
- **Special events:** Through examining the news sources during the periods of study, some special events that affected the traffic patterns were extracted and annotated as anomalies. Public funeral of Mr. Lee Kuan Yew (the first prime minister and founding father of Singapore) was held in March 2015³, which took place in downtown Singapore and caused major closure of roads. In addition, the rehearsal for state funeral procession⁴ and public tribute ceremony⁵ took place two days before the funeral. The time-stamps of these events are included in the anomaly set.
- **Public transportation breakdowns:** Breakdown or disruption in operations of public transportation systems may affect the traffic condition, especially the mobility pattern of taxis nearby the affected regions. We identified major breakdowns and disruptions with more than 30 minutes delays in Singapore mass rapid transit (MRT) system during the period of study^{6,7}, and included the affected time-stamps into the anomaly set.
- **Point anomalies** Using exploratory data analysis on the dataset, we noticed a few unusual surges in traffic in all regions of TaxiSG dataset, probably due to some errors in data generation process. As these types of outliers are prevalent in real-world industrial settings, they were also included in the anomaly dataset.

The detailed information about the annotated anomaly times-tamps during the period of study are provided in Table 5.2.

Evaluation Procedure: AUC metric is used as the evaluation metric for measuring performance of different models in detecting the annotated anomalies. We compute the False Positive

¹<https://www.theguardian.com/world/2015/nov/28/beijing-residents-told-to-stay-inside-as-smog-levels-soar>

²<https://www.aljazeera.com/indepth/inpictures/2015/11/winter-smog-pollution-beijing-china-15113006333537.html>

³<https://www.bbc.com/news/world-asia-32102686>

⁴<https://www.straitstimes.com/singapore/rehearsal-for-state-funeral-procession-early-friday-morning>

⁵<https://graphics.straitstimes.com/STI/STIMEDIA/2015/lky-8day-coverage/docs/28.pdf>

⁶<https://www.todayonline.com/singapore/train-service-disrupted-east-west-line-2nd-disruption-day>

⁷<https://www.straitstimes.com/singapore/transport/power-shutdown-at-north-south-east-west-lines-past-major-train-disruptions>

Anomaly Type	Date	From	To	#	Dataset	Remarks
Public Holidays	1 Jan. 2015	00:00	23:59	288	TaxiSG	New Year's Day
	19 Feb. 2015	00:00	23:59	288	TaxiSG	Chinese New Year
	20 Feb. 2015	00:00	23:59	288	TaxiSG	Chinese New Year
	3 Apr. 2015	00:00	23:59	288	TaxiSG	Good Friday
	1 Jan. 2016	00:00	23:59	48	TaxiBJ	New Year's Day
	7 Feb. 2016- 13 Feb. 2016	00:00	23:59	336	TaxiBJ	Chinese New Year
	4 Apr. 2016	00:00	23:59	48	TaxiBJ	Ching Ming Festival
Off days	28 Nov. 2015	00:00	23:59	48	TaxiBJ	Pollution Warning
	30 Nov. 2015	00:00	23:59	48	TaxiBJ	Pollution Warning
	8 Dec. 2015- 10 Dec.2015	00:00	12:00	120	TaxiBJ	Pollution Warning
	19 Dec. 2015- 21 Dec.2015	00:00	23:59	144	TaxiBJ	Pollution Warning
	25 Dec. 2015	00:00	23:59	48	TaxiBJ	Pollution Warning
Special Events	29 Mar. 2015	12:00	17:00	100	TaxiSG	State Funeral of Mr. LKY
	27 Mar. 2015	00:00	5:00	100	TaxiSG	State Funeral Rehearsal
	27 Mar. 2015	18:00	23:59	100	TaxiSG	Public Tribute Ceremony
MRT Disruptions	30 Jan. 2015	9:50	10:40	10	TaxiSG	
	3 Feb. 2015	7:10	8:00	10	TaxiSG	
	23 Feb. 2015	15:07	19:33	54	TaxiSG	
	24 Feb. 2015	12:18	13:35	16	TaxiSG	
	27 Feb. 2015	21:10	22:50	20	TaxiSG	
	3 Mar. 2015	7:28	8:11	10	TaxiSG	
	3 Mar. 2015	17:35	19:36	18	TaxiSG	
	9 Mar. 2015	17:35	23:59	62	TaxiSG	
	10 Mar. 2015	00:00	16:40	200	TaxiSG	
	23 Mar. 2015	19:30	21:20	22	TaxiSG	
	27 Mar. 2015	18:00	18:40	8	TaxiSG	
	30 Mar. 2015	19:20	20:20	12	TaxiSG	
	13 Apr. 2015	16:20	17:10	10	TaxiSG	
	28 Apr. 2015	19:30	20:40	14	TaxiSG	
Point anomalies	15 Apr. 2015	18:40	19:15	8	TaxiSG	
	18 Apr. 2015	19:30	20:05	8	TaxiSG	

Table 5.2: Annotated anomaly timestamps in TaxiSG and TaxiBJ datasets

Model	TaxiSG	TaxiBJ
ARIMA	0.406	0.266
TLR	0.459	0.716
STAE [ZDS ⁺ 17]	0.429	0.315
2D ConvAE [HCN ⁺ 16]	0.437	0.341
3D ConvAE	0.445	0.344
GSTAD-Cls ($L = 6$)	0.734	0.758
GSTAD-Cls ($L = 4$)	0.736	0.761
GSTAD-Cls ($L = 3$)	0.739	0.789
GSTAD-Cls ($L = 2$)	0.729	0.789
GSTAD-Cls ($L = 1$)	0.719	0.786
GSTAD-Cls ($L = 0.5$)	0.707	0.786
GSTAD-Reg	0.776	0.804

Table 5.3: AUC scores obtained by different models on TaxiSG and TaxiBJ datasets. In GSTAD-Cls models, L refers to the interval length in hours for each class.

Rate ($\frac{FP}{FP+TN}$) and True Positive Rate ($\frac{TP}{TP+FN}$) based on each value of the thresholds of the models, and plot the ROC curve. For the regression-based models, considering that the interval values are $\tau = 5$ or $\tau = 30$ in the experiments, we change the value of q in Equation 5.11 to set the time different between the groundtruth and the prediction from 0 to near 4 hours. For the classification-based models, the threshold is set as the class probability threshold p as defined in the Methods section. Finally, for the reconstruction-based models, the threshold value is changed in the range of percentile values of the reconstruction error (Equation 5.5) on the training set.

5.4.2 Anomaly Detection Performance

In the first experiment we compare the performance of our proposed temporal anomaly detection model using classification-based approach (GSTAD-Cls) and regression-based approach (GSTAD-Reg) with some of the state-of-the-art spatio-temporal anomaly detection baselines. We implemented the baselines according to their proposed architecture and loss functions, with minor modifications to adapt to our setting. The classification-based and regression-based models were trained by optimizing the objective function introduced in Equation (5.4). The $\mathcal{L}_{pred}$ term is defined exclusively for the classification and regression based methods. For the classification based approach, 6 models with non-overlapping interval lengths $L = \{6, 4, 3, 2, 1, 0.5\}$ hours were trained. The AUC scores obtained by each model on TaxiSG and TaxiBJ datasets are reported in Table 5.3.

The results obtained from different models indicate that both temporal anomaly detection models (GSTAD-Reg and GSTAD-Cls) outperform other baselines by a large margin. The reason for the difference in performance is that STAE, 2D ConvAE and 3D ConvAE models

detect anomalies solely based on the reconstruction error. Although in many spatio-temporal problems such as anomaly detection in videos, the appearance of anomalies can be detected using reconstruction-based approaches [KTP18, HCN⁺16], in geo-spatial domain most of anomaly patterns have been previously observed. Therefore, such temporal anomaly patterns would not be captured by the reconstruction based models well.

In addition, we compared our models with ARIMA and a temporal linear regression model (TLR), which use time-series prediction mechanism and temporal metadata prediction (similar to ours), respectively, to detect anomalies. The two baselines of ARIMA and TLR are thus comparable with reconstruction-based methods (e.g. STAE, ConvAE) and with our proposed methods of GSTAD, respectively. ARIMA does not show good performance compared to other models, even to reconstruction-based methods. However, TLR outperforms the reconstruction-based models, but not GSTAD models. We discuss it in detail with different types of anomalies in the next section.

The regression-based model (GSTAD-Reg) outperforms all classification-based models (GSTAD-CIs) on both datasets, which can be explained by better modeling of temporal continuity in the regression-based approach. The time-slots in the classification models are discrete, and temporal continuity between adjacent time-slots is not captured. Therefore, as the class interval length L decreases, more instances are misclassified into adjacent time-slots as shown in Table 5.4.

Model	All misclassified	Classified to adjacent	ratio
GSTAD-CIs (L=6)	345	183	0.530
GSTAD-CIs (L=4)	573	372	0.649
GSTAD-CIs (L=3)	694	446	0.642
GSTAD-CIs (L=2)	948	716	0.755
GSTAD-CIs (L=1)	2010	1713	0.852
GSTAD-CIs (L=0.5)	3753	3042	0.810

Table 5.4: The misclassified instances of GSTAD-CIs models on odd test-days dataset. As the interval length decreases, higher percentage of instance are mis-classified into adjacent time-slots.

5.4.3 Model Architecture Analysis

The proposed GSTAD model is trained by jointly minimizing reconstruction and prediction losses, as shown in Equation (5.4). To evaluate the influence of the reconstruction term, the performance of the best performing model GSTAD-Reg is compared with and without the reconstruction term in its objective function.

ConvNet Architecture	Loss term	TaxiSG	TaxiBJ
Conv3D (8 images)	Rec+Reg	0.776	0.804
	Reg Only	0.734	0.783
Conv2D (8 images)	Rec+Reg	0.688	0.791
	Reg Only	0.713	0.789
Conv2D (1 image)	Rec+Reg	0.744	0.765
	Reg Only	0.757	0.766
FC-Net (1 image)	Rec+Reg	0.736	0.731
	Reg Only	0.711	0.700

Table 5.5: AUC scores obtained by different model architectures on TaxiSG and TaxiBJ datasets (Rec and Reg stand for reconstruction and regression respectively).

Moreover, 3D convolution is used in the encoder and decoder networks, where the input is a sequence of 8 consecutive spatial images (as illustrated in Figure 5.2). To compare the performance of different architectures in capturing spatial and temporal dependencies in the data and learning discriminative features, we evaluated and compared the performance of the regression-based model (GSTAD-Reg) using 3D convolution and 2D convolution on 8 concatenated spatial images, 2D convolution on a single spatial image and fully connected network on a single spatial image. The results are summarized in Table 5.5

The results in Table 5.5 show that the models which are trained by joint optimization of reconstruction and regression errors generally get higher scores than the ones which only perform regression. This shows that reconstruction process is helpful in learning representative and discriminative features that are used by the regression network.

Moreover, the models with 3D convolutions outperform models with 2D convolutions and fully connected networks. This shows that 3D convolution is effective in capturing spatial and temporal dependencies in the data to learn useful features. On TaxiSG dataset, 2D convolution on 8 spatial images performs even worse than 2D convolution and fully connected network on a single spatial image. This shows that on the one hand, 2D convolutional networks are not reliable in capturing temporal dependencies in the spatio-temporal data, as they perform convolutions spatially, causing the temporal information to collapse after the first layer. On the other hand, it shows that single frames which contain large amount of spatial data, are strong baselines even in temporal domains, as observed and explained in some other studies on spatio-temporal data in the literature [KTS⁺14].

We also tried higher number of depth for spatial images (32 consecutive images instead of 8), but the results obtained for GSTAD models did not change significantly, which may show that 8 consecutive images provide sufficient temporal information for prediction of temporal metadata in our tasks.

5.4.4 Visualization of Temporal Anomalies

In a real-time anomaly detection system, it is important to visualize the regions that show the most abnormal patterns. In this experiment, we use the dictionary-based approach explained in the Methods section, to visualize the most affected regions at each time point using heatmap images. Some examples of the generated heatmaps are illustrated in Figure 5.4.

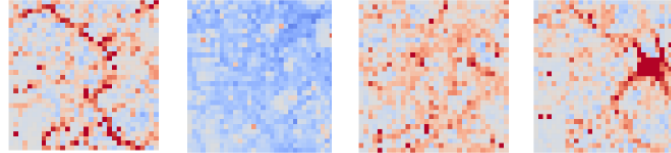


Figure 5.4: Examples of temporal anomaly heatmaps on Singapore taxi dataset. Red and blue colors indicate abnormal high and low crowd density w.r.t. to mean crowd density value in each region respectively. Time-stamps from left to right: 1 January 2015 5:00 a.m., 1 January 2015 8:15 a.m., 27 March 2015 23:30 p.m., 27 March 2015 2:20 a.m.

The examples in Figure 5.4 show regions that have abnormally high or low crowd density with respect to the mean density value in the same region at normal time-stamps. For instance, as illustrated in Figure 5.4, the number of taxis at midnight of the New Year day surges rapidly, as taxis try to pick up passengers who were participating in new year celebration events in downtown, while in the morning of the same holiday, the number of taxis drops rapidly compared to other working days, which usually experience rush hours at the same time.

As another example, on the night of 27 March 2015, two days before public funeral of Mr. Lee Kuan Yew (First prime minister of Singapore), massive crowd gathered in downtown Singapore to pay the last tribute to Mr. Lee (Figure 5.5 left). At the midnight of the same day, the rehearsal for the funeral procession was held in streets of Singapore, which could be observed by the public (Figure 5.5 right). In both cases the presence of huge crowd in streets and high demands for taxis created abnormal traffic patterns which can be seen in Figure 5.4.



Figure 5.5: (Left) 27 March 2015 23:30 p.m., massive crowd in downtown Singapore gathered to pay tribute to Mr. Lee Kuan Yew. (Right) 27 March 2015 midnight, rehearsal of funeral procession ceremony in downtown Singapore. Both events caused abnormal traffic patterns which were captured by the proposed GSTAD model.

5.5 Summary

In this chapter, we proposed a novel geo-spatio-temporal anomaly detection model for geo-spatio-temporal domains. By jointly optimizing reconstruction loss and temporal metadata prediction loss, the proposed model is able to detect anomaly patterns in crowd density data, which could not be captured by the existing reconstruction based spatio-temporal anomaly detection models. A regression-based approach was proposed for temporal metadata prediction model, in which different time points in a week were assigned unique vectors in 3-dimensional Euclidean space, such that the temporal continuity among the time points could be preserved. Through experimental studies, we showed that the regression-based approach outperforms the classification-based method in detection of temporal anomalies. In addition, by using 3D convolutions in the encoder and decoder architecture, the proposed model is able to capture spatial and temporal dependencies in the data and learn representative features. The experiments conducted on two real world annotated taxi trajectory datasets in Singapore and Beijing, showed the effectiveness of the *GSTAD* model in capturing geo-spatial anomaly patterns.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, considering different types of changes that occur in temporal data, three models were developed to 1) **track** the *changes of multiple concepts* over time, 2) **predict** the future state of a geo-spatio-temporal space, by utilizing the *recurring concepts* and 3) **detect outliers**, by identifying temporal anomalies. The developed models are applicable to real world environments, where the data are mostly unlabeled and in large-scale, and can help practitioners and decision makers to better understand the changes, and gain useful insights by analyzing the temporal data in such settings. An overview of the proposed models based on each type of change in temporal data is illustrated in Figure 6.1.

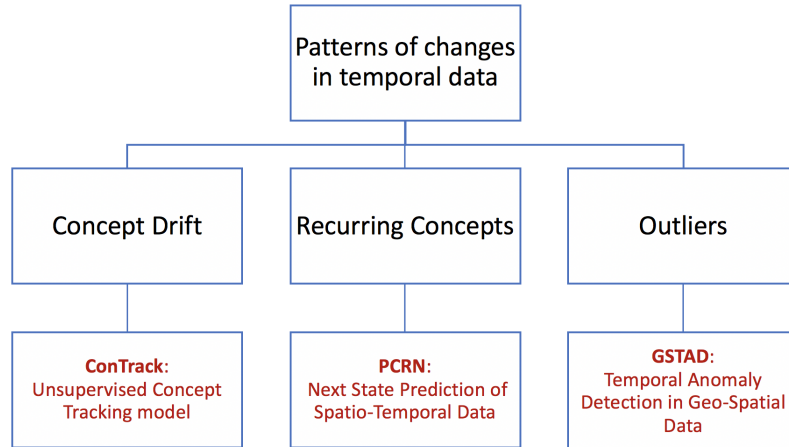


Figure 6.1: Overview of the proposed models

In the first study, we proposed ‘ConTrack’, a scalable model to track multiple concepts and actors participation in each concept in unlabeled data over time, to provide interpretable insight to practitioners and data analysts. The proposed model is applicable to large-scale

unsupervised settings, with multiple concepts and multiple actors, which has received little attention in the concept drift and stream mining literature. Moreover, the proposed model is formulated as an optimization problem, which can be parallelized over a cluster of machines to handle large-scale data in a distributed fashion. The results of empirical studies on synthetic, semi-synthetic and real world data showed the effectiveness of ConTrack in tracking multiple concepts, over different popular unsupervised methods.

In the second study, an optimization based model named ‘PCRN’ for problem of crowd density prediction in geo-spatial environments was proposed. By utilizing a pyramidal convolutional recurrent network architecture, the proposed model can capture long-term spatial and temporal dependencies in the data. Moreover, by leveraging the recurring concepts existing in geo-spatio-temporal data such as periodical patterns over different time scales (days, weeks, etc.), and by incorporating such *periodic representations* through a weighted fusion mechanism into the prediction model, the proposed PCRN model outperforms other linear and non-linear prediction baselines in one-step and multi-step ahead prediction tasks.

Finally, a geo-spatio-temporal anomaly detection model ‘GSTAD’ was proposed, to identify abnormal patterns in geo-spatio-temporal domains. Using a convolutional autoencoder architecture with 3D convolutional layers, the proposed model captures spatial and temporal dependencies in the data effectively, in order to learn representative features. Moreover, using a temporal metadata prediction model, the model is able to capture temporal anomaly patterns in crowd density data, which could not be detected by the existing reconstruction-based spatio-temporal anomaly detection models. For metadata prediction model, a novel regression-based approach is proposed by assigning unique real-valued vectors in 3-dimensional Euclidean space to different time points in a weekends and weekdays, which preserves temporal continuity in prediction. The experiments show superiority of the regression-based approach as compared to the classification-based approach. Through experiments on real world taxi data from Singapore and Beijing, and annotating the anomaly time-stamps in those datasets, we showed the effectiveness of the proposed model in detection of the temporal anomaly patterns.

Altogether, the three aforementioned models (concept tracking, next state prediction and anomaly detection) can provide valuable insights for practitioners and decision makers, to analyse real world temporal data in different applications. The proposed *concept tracking* model is applicable to many real world settings with large scale unlabelled data, where multiple records are generated by multiple actors in sequential time frames. The *next state prediction* and *anomaly detection* models are applicable to geo-spatio-temporal settings, and leverage the recurring periodic patterns in such data for accurate prediction of the next state and detection of temporal anomalies.

6.2 Future Directions

Streaming ConTrack: In Chapter 3, we presented our concept tracking algorithm ‘ConTrack’. Our current algorithm outputs the concepts and participation vectors for each time point. However, it requires temporal data to be presented in batch format. In many real world applications,

there is a need for online algorithms that can process data in streaming format. Some steps have been taken to develop ‘Streaming ConTrack’ model, which can operate in an online fashion. The challenge in developing the streaming concept tracking algorithm is the optimization procedure, which requires the whole dataset to find the best set of concepts and participation vectors at discrete time points. We envision an extension to our algorithm where the data epochs are provided as input in a streaming fashion, and the new variables corresponding to the new data are created, initialized, and optimized as needed, possibly via a sliding window strategy.

Incremental Learning Approaches for Prediction/ Anomaly Detection Problems: In Chapters 4 and 5, prediction and anomaly detection problems were addressed in geo-spatial settings using optimization-based approaches. In some real world applications, incremental concept drift might occur in a long period of time, causing the models to become inaccurate in prediction and anomaly detection tasks. For instance, incremental increase in production of cars, can increase the density of vehicles moving in the streets over the years, which causes drift in crowd density data. Retraining these large models from scratch is not an efficient approach. Therefore, we envision extensions of these models that can incrementally update their parameters after receiving new data mini-batches and forget the old concepts, without adversely affecting the performance of the models.

Representation Learning for Temporal Data: Large amount of the generated data in real world applications are unstructured and unlabeled. Representation learning or feature learning is the process of extracting or learning useful features from such data that can be subsequently used for other machine learning tasks such as classification, prediction, anomaly detection and similarity matching. Different unsupervised methods such as PCA, Matrix Factorization and autoencoders have been used for representation learning task.

Most existing methods learn representations from the raw data without considering the temporal information. This representation learning approach is suitable for non-temporal data such as texts or images, but in many temporal settings, embedding the temporal information into the learned representations is crucial. For instance in geo-spatial settings, learning representations for trajectories of vehicles is useful for different tasks such as trajectory clustering, anomaly detection and finding vehicles with similar trajectory patterns. However, in this problem, it is also important to take the temporal information of the trajectories into account to learn better representative features from the trajectory data. To this end, using effective fusion mechanism to combine raw data with temporal metadata is essential in representation learning process.

Moreover, in recent years, the adversarial mechanism originally used in generative adversarial networks [GPAM⁺14], have been used in representation learning problem to match the aggregated posterior of the representation vectors with an arbitrary prior distribution defined by the practitioner [MSJ⁺15]. We would like to study the feasibility of using such approach for temporal settings; possibly by defining different set of prior distributions for different time contexts (days, hours, etc.), and using adversarial mechanism to push the distribution of the learned representations from each time context to their corresponding prior distribution. Using

such approach in representation learning can be helpful in detection of anomalies or finding similar instances with respect to a specific temporal context.

Appendix A

List of Publications

- **Zonoozi, Ali**, Qirong Ho, Shonali Krishnaswamy, and Gao Cong. "ConTrack: A Scalable Method for Tracking Multiple Concepts in Large Scale Multidimensional Data." In Data Mining (ICDM), 2016 IEEE 16th International Conference on, pp. 759-768. IEEE, 2016.
- **Zonoozi, Ali**, Jung-jae Kim, Xiao-Li Li, and Gao Cong. "Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns." In IJCAI, pp. 3732-3738. 2018
- **Zonoozi, Ali**, Jung-jae Kim, Xiao-Li Li, and Gao Cong. "GSTAD: A Geo-Spatio-Temporal Anomaly Detection Model." Under review for IJCAI-2019.

References

- [Agg15] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.
- [AHE⁺11] Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*, pages 267–276. ACM, 2011.
- [AHWY03] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.
- [AHWY04] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment, 2004.
- [AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [BBK14] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2014.
- [BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

REFERENCES

- [BEF84] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.
- [BG07] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SDM*, volume 7, page 2007. SIAM, 2007.
- [BGdCÁF⁺06] Manuel Baena-Garcia, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.
- [BJ76] George Edward Pelham Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. 1976.
- [BJRL15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [BL06] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.
- [Ble12] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
- [BM07] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, 2007.
- [BM⁺08] Stephen H Bach, Marcus Maloof, et al. Paired learners for concept drift. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 23–32. IEEE, 2008.
- [BMW⁺11] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

REFERENCES

- [BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, 2010.
- [BPD13] Michael Scott Brown, Michael J Pelosi, and Henry Dirska. Dynamic-radius species-conserving genetic algorithm for the financial forecasting of dow jones index stocks. In *Machine Learning and Data Mining in Pattern Recognition*, pages 27–41. Springer, 2013.
- [BWSK17] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Fully context-aware video prediction. *arXiv preprint arXiv:1710.08518*, 2017.
- [BYPC16] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.
- [CAB16] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multi-scale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.
- [CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.
- [CC05] Zhe Chen and Andrzej Cichocki. Nonnegative matrix factorization with temporal smoothness and/or spatial decorrelation constraints. *Laboratory for Advanced Brain Signal Processing, RIKEN, Tech. Rep*, 68, 2005.
- [CEQZ06] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, volume 6, pages 328–339. SIAM, 2006.
- [CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

REFERENCES

- [CGCB15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.
- [Cha00] Chris Chatfield. *Time-series forecasting*. Chapman and Hall/CRC, 2000.
- [CJLV16] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.
- [CLMW11] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.
- [CMW07] Mario Cools, Elke Moons, and Geert Wets. Investigating effect of holidays on daily traffic counts: time series approach. *Transportation Research Record: Journal of the Transportation Research Board*, (2019):22–31, 2007.
- [CNJJH09] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):6164–6173, 2009.
- [CPC⁺18] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.
- [CPR11] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. In *Pattern Recognition and Image Analysis*, pages 289–296. Springer, 2011.
- [CS06] Sanjay Chawla and Pei Sun. Slom: a new measure for local spatial outliers. *Knowledge and Information Systems*, 9(4):412–429, 2006.
- [CT07] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2007.

REFERENCES

- [CVMG⁺14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [CYL11] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3449–3456. IEEE, 2011.
- [DAHG⁺15] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR2015*, pages 2625–2634, 2015.
- [DC97] Mark S. Dougherty and Mark R. Cobbett. Short-term inter-urban traffic forecasts using neural networks. *International Journal of Forecasting*, 13(1):21–31, 1997.
- [DH00] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000.
- [DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.
- [DKVY06] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

REFERENCES

- [DP09] Mahashweta Das and Srinivasan Parthasarathy. Anomaly detection and spatio-temporal analysis of global climate system. In *Proceedings of the third international workshop on knowledge discovery from sensor data*, pages 142–150. ACM, 2009.
- [DRAP15] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.
- [dRFMdAPAB16] Denis dos Reis, Peter Flach, Stan Matwin, and Gustavo Enrique de Almeida Prado Alves Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1545–1554, 2016.
- [EHB96] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pages 493–499, 1996.
- [ERKL16] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.
- [FGL16] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, pages 64–72, 2016.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [GFHP07] Jing Gao, Wei Fan, Jiawei Han, and S Yu Philip. A general framework for mining concept-drifting data streams with skewed distributions. In *SDM*, pages 3–14. SIAM, 2007.
- [GGAH14] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

REFERENCES

- [GMCR04] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in artificial intelligence–SBIA 2004*, pages 286–295. Springer, 2004.
- [GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.
- [GNHS11] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [GPT06] Pedro Galeano, Daniel Peña, and Ruey S Tsay. Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, 101(474):654–669, 2006.
- [GTDVMFV09] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.
- [GZB⁺14] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.
- [HCN⁺16] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 733–742. IEEE, 2016.
- [HDX02] Zengyou He, Shengchun Deng, and Xiaofei Xu. Outlier detection integrating semantic knowledge. In *International Conference on Web-Age Information Management*, pages 126–131. Springer, 2002.

REFERENCES

- [Hin09] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.
- [HKB16] Ahsanul Haque, Latifur Khan, and Michael Baron. Sand: semi-supervised adaptive novel class detection and classification over data stream. In *AAAI’16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1652–1658, 2016.
- [HM10] David J Hill and Barbara S Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010.
- [Ho14] Qirong Ho. *Modeling Large Social Networks in Context*. PhD thesis, PhD thesis, Carnegie Mellon, 2014.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [HSD01] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.
- [HSX11] Qirong Ho, Le Song, and Eric P Xing. Evolving cluster mixed-membership blockmodel for time-varying networks. 2011.
- [Hyn10] Rob J Hyndman. Forecasting with long seasonal periods. <https://robjhyndman.com/hyndsight/longseasonality/>, 2010. Accessed: 2018-01-10.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [JXY13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.
- [KB06] Alp Kut and Derya Birant. Spatio-temporal outlier detection in large databases. *Journal of computing and information technology*, 14(4):291–297, 2006.

REFERENCES

- [KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.
- [KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.
- [KF⁺12] Ludmila Kuncheva, William J Faithfull, et al. Pca feature extraction for change detection in multidimensional unlabelled streaming data. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1140–1143. IEEE, 2012.
- [KHA99] Mark G Kelly, David J Hand, and Niall M Adams. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 367–371. ACM, 1999.
- [KLC06] Yufeng Kou, Chang-Tien Lu, and Dechang Chen. Spatial weighted outlier detection. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 614–618. SIAM, 2006.
- [Kli04] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *intelligent data analysis*, 8(3):281–300, 2004.
- [KLLK10] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y Lee, and Tae-Seong Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *Information Technology in Biomedicine, IEEE Transactions on*, 14(5):1166–1172, 2010.
- [KM⁺03] Jeremy Z Kolter, Marcus Maloof, et al. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 123–130. IEEE, 2003.
- [KP03] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, (1857):74–84, 2003.

REFERENCES

- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [KTP18] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):36, 2018.
- [KTS⁺14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
- [KW95] Miroslav Kubat and Gerhard Widmer. Adapting to drift in continuous domains. 1995.
- [Las06] R Lasaponara. On the use of principal component analysis (pca) for evaluating interannual vegetation anomalies from spot/vegetation ndvi temporal series. *Ecological Modelling*, 194(4):429–434, 2006.
- [LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LBE15] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.
- [LBF13] Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on ITS*, 14(2):871–882, 2013.
- [LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.
- [LDK⁺15] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on ITS*, 16(2):865–873, 2015.

REFERENCES

- [LGA17] Tao Lin, Tian Guo, and Karl Aberer. Hybrid neural networks for learning the trend in time series. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2273–2279, 2017.
- [Lin07] Chuan-bi Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.
- [LKC16] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.
- [LL13] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *Communications Surveys & Tutorials, IEEE*, 15(3):1192–1209, 2013.
- [LLG17] Weixin Luo, Wen Liu, and Shenghua Gao. Remembering history with convolutional lstm for anomaly detection. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 439–444. IEEE, 2017.
- [LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [LSJ13] Cewu Lu, Jianping Shi, and Jiaya Jia. Abnormal event detection at 150 fps in matlab. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2720–2727. IEEE, 2013.
- [MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.
- [MCL16] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.
- [MHS⁺13] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

REFERENCES

- [MKB⁺10] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTER-SPEECH*, pages 1045–1048. ISCA, 2010.
- [MP03] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.
- [MS16a] Yasuko Matsubara and Yasushi Sakurai. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1045–1054. ACM, 2016.
- [MS16b] Jefferson Ryan Medel and Andreas Savakis. Anomaly detection in video using predictive convolutional long short-term memory networks. *arXiv preprint arXiv:1612.00390*, 2016.
- [MSF15] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. The web as a jungle: Non-linear dynamical systems for co-evolving online activities. In *Proceedings of the 24th International Conference on World Wide Web*, pages 721–731. International World Wide Web Conferences Steering Committee, 2015.
- [MSJ⁺15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.
- [MTW⁺15] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.
- [NHW⁺11] EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569, 2011.

REFERENCES

- [NLRK⁺10] Masahiro Nakano, Jonathan Le Roux, Hirokazu Kameoka, Yu Kitano, Nobutaka Ono, and Shigeki Sagayama. Nonnegative matrix factorization with markov-chained bases for modeling time-varying patterns in music spectrograms. In *Latent Variable Analysis and Signal Separation*, pages 149–156. Springer, 2010.
- [OF97] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.
- [OGL⁺15] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, pages 2863–2871, 2015.
- [PEK⁺06] Juha Pärkkä, Miikka Ermes, Panu Korpiä, Jani Mäntyjärvi, Johannes Peltola, and Ilkka Korhonen. Activity classification using realistic data from wearable sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):119–128, 2006.
- [QAWZ15] Abdulhakim A Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944. ACM, 2015.
- [RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [RNS⁺17] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. 09 2017.
- [Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
- [RS12] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pages 108–109. IEEE, 2012.

REFERENCES

- [SAK14] Junming Shao, Zahra Ahmadi, and Stefan Kramer. Prototype-based learning on concept-drifting data streams. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 412–421. ACM, 2014.
- [Sal97] Marcos Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11:133–155, 1997.
- [SH12] Ammar Shaker and Eyke Hüllermeier. Iblstreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4):235–249, 2012.
- [SK01] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM, 2001.
- [SLZ02] Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. Detecting graph-based spatial outliers. *Intelligent Data Analysis*, 6(5):451–468, 2002.
- [SMS15] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.
- [SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [SSW⁺17] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *Lecture Notes in Computer Science Information Processing in Medical Imaging*, page 146–157, 2017.
- [Sut13] Ilya Sutskever. Training recurrent neural networks. *University of Toronto, Toronto, Ont., Canada*, 2013.

REFERENCES

- [SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [SWJR07] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 667–676. ACM, 2007.
- [SZ14a] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.
- [SZ14b] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [Tay03] James W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003.
- [TBF⁺15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4489–4497. IEEE, 2015.
- [TPP00] Ruey S Tsay, Daniel Peña, and Alan E Pankratz. Outliers in multivariate time series. *Biometrika*, 87(4):789–804, 2000.
- [Tsy04] Alexey Tsymbal. The problem of concept drift: definitions and related work. 2004.
- [VB06] Peter Vorburger and Abraham Bernstein. Entropy-based concept shift detection. In *Sixth International Conference on Data Mining (ICDM’06)*, pages 1113–1118, 2006.
- [VdODS13] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

REFERENCES

- [Vir03] Tuomas Virtanen. Sound source separation using sparse coding with temporal continuity objective. In *Proc. ICMC*, volume 3, pages 231–234, 2003.
- [Vir07] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, 2007.
- [VPT16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.
- [VXD⁺14] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.
- [vZl10] Indre vZliobaite. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.
- [WCP13] Weiran Wang and Miguel A Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.
- [Wer90] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [WFYH03] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM, 2003.
- [WH03] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.
- [WHL04] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems*, 5(4):276–281, 2004.

REFERENCES

- [WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.
- [WLC10] Elizabeth Wu, Wei Liu, and Sanjay Chawla. Spatio-temporal outlier detection in precipitation data. In *Knowledge discovery from sensor data*, pages 115–133. Springer, 2010.
- [WSC⁺16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
- [XBK⁺15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.
- [XCW⁺15] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.
- [XHD⁺15] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu Petuum. A new platform for distributed machine learning on big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, volume 15, 2015.
- [XRY⁺15] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.
- [XYRS17] Dan Xu, Yan Yan, Elisa Ricci, and Nicu Sebe. Detecting anomalous events in videos by learning deep representations of appearance and motion. *Computer Vision and Image Understanding*, 156:117–127, 2017.

REFERENCES

- [YGH⁺15] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1351–1361. International World Wide Web Conferences Steering Committee, 2015.
- [YHNV⁺15] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.
- [YKX⁺05] Sun Yuxiang, Xie Kunqing, Ma Xiujun, Jin Xingxing, Pu Wen, and Gao Xiaoping. Detecting spatio-temporal outliers in climate dataset: A method study. In *Geoscience and Remote Sensing Symposium, 2005. IGARSS’05. Proceedings. 2005 IEEE International*, volume 2, pages 4–pp. IEEE, 2005.
- [YYD⁺16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Edouard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.
- [ZCQJ08] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15(2):181–214, 2008.
- [ZCWY14] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM TIST*, 5(3):38, 2014.
- [ZDDG17] Ali Ziat, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari. Spatio-temporal neural networks for space-time series forecasting and relations discovery. In *ICDM*, pages 705–714. IEEE, 2017.
- [ZDS⁺17] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1933–1941. ACM, 2017.

REFERENCES

- [ZFFX11] Bin Zhao, Li Fei-Fei, and Eric P Xing. Online detection of unusual events in videos via dynamic sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3313–3320. IEEE, 2011.
- [ZHKC16] Ali Zonoozi, Qirong Ho, Shonali Krishnaswamy, and Gao Cong. Contrack: A scalable method for tracking multiple concepts in large scale multidimensional data. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 759–768. IEEE, 2016.
- [ZKLC18] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. Periodic-crn: A convolutional recurrent model for crowd density prediction with recurring periodic patterns. In *IJCAI*, pages 3732–3738, 2018.
- [ZMH10] Yang Zhang, Nirvana Meratnia, and Paul Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 12(2):159–170, 2010.
- [ZP17] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM, 2017.
- [ZPH98] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62, 1998.
- [ZSM⁺18] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.
- [ZZQ17] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.
- [ZZS08] Peng Zhang, Xingquan Zhu, and Yong Shi. Categorizing and mining concept drifting data streams. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 812–820. ACM, 2008.