AUTOREGRESSIVE TENSOR DECOMPOSITION FOR NYC TAXI DATA ANALYSIS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Zongwei Li

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2020

Purdue University

Fort Wayne, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Jin Soung Yoo, Chair

Department of Computer Science

Dr. Peter Ng

Department of Computer Science

Dr. Zesheng Chen

Department of Computer Science

Approved by:

Dr. Jin Soung Yoo

Head of the Computer Science Graduate Program

Dedicated to my beloved mother.

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my supervisor Dr. Jin Soung Yoo, who gave me great advice along the way. She guided and encouraged me to overcome all the obstacles during the unprecedented pandemic, otherwise I would never have completed this project. In the meantime, I would like to pay my special regards to Dr. Peter Ng, who helped me adapt to living in a new country both in my academic study and my daily life, which I whole-heartedly appreciate: I would never come this far without your help. I also want to thank all the people whose assistance was a milestone in the completion of my thesis, especially Dr. Zesheng Chen. In his class of software project management, I learned a lot about how to manage a project, which proved to be critical in my thesis.

Last but not least, I am extremely grateful to my parents for their love, caring and sacrifice for giving me a promising future, even when mishaps have been striking them hard. Also I express my thanks to my grandfather, who enlightened and inspired me in many ways. He was the one who opened the door to a whole new world for me. My heartfelt thanks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Li, Zongwei M.S., Purdue University, August 2020. Autoregressive Tensor Decomposition for NYC Taxi Data Analysis. Major Professor: Jin Soung Yoo Associate Professor.

Cities have adopted evolving urban digitization strategies, and most of those increasingly focus on data, especially in the field of public transportation. Transportation data have intuitively spatial and temporal characteristics, for they are often described with when and where the trips occur. Since a trip is often described with many attributes, the transportation data can be presented with a tensor, a container which can house data in N-dimensions. Unlike a traditional data frame, which only has column variables, tensor is intuitively more straightforward to explore spatio-temporal data-sets, which makes those attributes more easily interpreted. However, it requires unique techniques to extract useful and relatively correct information in attributes highly correlated with each other. This work presents a mixed model consisting of tensor decomposition combined with seasonal vector autoregression in time to find latent patterns within historical taxi data classified by types of taxis, pick-up and drop-off times of services in NYC, so that it can help predict the place and time where taxis are demanded. We validated the proposed approach using the experiment evaluation with real NYC tax data. The proposed method shows the best prediction among alternative models without geographical inference, and captures the daily patterns of taxi demands for business and entertainment needs.

1. INTRODUCTION

Tensors are generalizations of matrices, which can be more than two dimensions. Thus, tensors can represent a data set whose responder has more than three predictors. In particular, a responder is a cell or entry of the tensor, which normally represents what we want to predict. Predictors are the "coordinates" of this specific entry, normally representing features, in which case has a number of more than three. It has always been an interesting topic whether methods used to find latent information hidden behind a specific matrix can be applied to a tensor as well, considering that tensors are generalizations of matrices. Among them, how to decompose a tensor so that some hidden patterns or information can be revealed is one of the most explored of all.

Historically, tensor and its decomposition were first introduced in the psychometric community [1] in 1927, but it was not until around the late 20th century that it caught computer scientists' attention [2], and then gained its popularity in machine learning [3]. Many tensor decompositions are inspired from matrix decompositions [1, 4–6]. One of them that is frequently used in machine learning is rank decomposition, which is originated from matrix rank decomposition based on Spearman's Hypothesis [7]. In his hypothesis, Spearman states that if a matrix M that is a data-set can be factorized into the following pattern:

$$M = AB^{\intercal}$$
$$\text{where} \quad M \in \mathbb{R}^{n\times m}, A \in \mathbb{R}^{n\times r}, B \in \mathbb{R}^{m\times r} \tag{1.0.1}$$

then intuitively the data-set M can be explained by r different hidden factors, which are encoded in the matrices A and $B^{\intercal}$. For example, in his original experiment, he tried to find out latent patterns behind test results from multiple students. He chose the number of latent factors r to be two, representing educative and reproductive

capabilities to see whether those results could be explained. This meant the data-set M would be able to be decomposed into two small matrices A and B as shown in Figure 1.0.1. As implied, in practice the latent variable r is normally pre-defined by

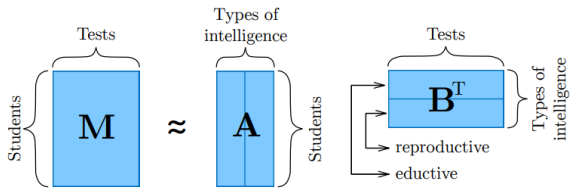

Fig. 1.0.1. Spearman's Hypothesis

user, and M cannot always be factorized into this pattern. This yields to a variation of the method by finding the "closest" matrix that fits the pattern, for error tolerance can be introduced in the real data-set M. Therefore, the mission is to find the M^*:

$$\begin{aligned} M^* &= \min_{\widehat{M}\in\mathbb{R}^{n\times m}} \left\| M - \widehat{M} \right\| \\ \text{subject to } & \widehat{M} = AB^\intercal, A \in \mathbb{R}^{n\times r}, B \in \mathbb{R}^{m\times r} \end{aligned} \tag{1.0.2}$$

However, unfortunately this decomposition of matrix is not unique, since we can always insert a rotation matrix $R \in \mathbb{R}^{r\times r}$ between A and B, and construct new A and B as shown below.

$$\begin{aligned} M &= AB^\intercal = AR^{-1}RB^\intercal = \left(AR^{-1}\right)\left(BR^\intercal\right)^\intercal \\ &= \widetilde{A}\widetilde{B}^\intercal \end{aligned} \tag{1.0.3}$$

Therefore, in order to get a unique decomposition, there are always stringent conditions. Even though, this method can still be applied to tensors with more than two dimensions but not directly since the product of tensors needs to be defined, and what is more interesting is that when it comes to tensor with more than two dimensions, decomposition is largely unique with much milder restrictions [3]. Like

matrix decomposition, which can reveal latent factors through lower rank matrices, tensor decomposition can also uncover underlying low-dimensional structures, which can be applied to tackle typical machine learning problems, such as feature reduction and pattern recognition.

One of common usages of tensors is in the field of spatio-temporal data-sets, which are aimed to manage both space and time information. There have been a lot of applications recently, including spatial statistics [8] and economics [9], remote sensing [10], public health [11] and so on. One of the reasons using tensors is that we can operate decomposition on a spatial and temporal data-set, since it allows multiple dimensions, and after the decomposition, the latent models generated can potentially predict future data or other data from unknown geographical information. [12]

In this paper, we will explore the famous data-set from NYC Taxi and Limousine Commission [13], where contains trip records of two kinds of taxis: yellow and green including when and where a specific taxi picks up and drops off passengers. Our goal is to forecast future demand of taxis in each location based on the history data. Since the data itself is highly correlated internally, it is not wise to analyze each type of taxis separately: we have to analyze it as a whole. Therefore, apart from the temporal and spatial dimensions, we need one additional dimension to describe the type of services: this motivates us to deal with data in a tensor format.

To analyze NYC Taxi Data, first, we will log the time and location of pick-ups and drop-offs of two different taxis in a 30-minute window, and then merge them into a tensor with three dimensions: timestamp, location and types of service. In order to reveal latent factors, we will employ Canonical Polyadic Decomposition on this tensor, which will break down the tensor into three components: one for each dimension. Considering that we are going to forecast future demands, we will try to extend the component of time window by using Vector Autoregression with seasonal variables. Finally, we can recover the original tensor by using n-mode product, which will immediately give us the prediction we need.

We conducted the experimental evaluation of the proposed approach. The experimental result shows that the outcome produced by our model is the best among classic models [14–16] which have already applied on NYC Taxi Data-set except DAR [12], which uses geographical inference. Since the data-set shows periodicity through time, by adding seasonal variables to Vector Autoregression model, our mixed model could capture daily pattern of taxi demands, and successfully inferred business and entertainment needs of taxis.

To illustrate, the paper will be laid out in the following sections: Section II introduces some basic definitions and notations for tensors; Section III further discusses about related works both in the field of tensor and applications on spatio-temporal data; Section IV illustrates our model concerning Canonical Polyadic Decomposition (CPD) and vector autoregression (VAR), which will be applied to NYC Taxi Data Section V is our case study on a NYC taxi data-set to evaluate the approach we propose; Section VI concludes this work and discusses about future work and some related topics explored.

2. INTRODUCTION TO TENSORS

Tensor can be thought of as multi-way collections of numbers, which typically comes from a field (like $\mathbb{R}$). For example, a tensor with three dimensions can be treated as a stack of multiple matrices of the same size. To begin with, we need to clarify some definitions and notations of tensors, since there have been so far multiple different notations in the field of tensor. We will borrow most of our notations from Rabanser et al. [17].

Definition 2.0.1 (Tensor Order) *The order of a tensor is the number of its dimension. A Nth-order tensor is denoted as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where I_n denotes the number of elements in this specific dimension.*

In the simplest high-dimensional case, such a tensor would be a three-dimensional array, which can be thought of as a data cube. We will predominately use third-order tensors in this paper. However, most of notations can be naturally extended into higher dimensional.

Definition 2.0.2 (Fibers and Slices) *Fibers of a tensor are defined by fixing all but one index, while slices are defined by fixing one index.*

For example, in a third-order tensor $\mathcal{X}$, fibers are $x_{:jk}$, $x_{i:k}$, $x_{ij:}$, and slices are $x_{::k}, x_{:j:}, x_{i::}$. Figures 2.0.1 and 2.0.2 show the graphical examples of fibers and slides for a third-order tensor.

We need to define tensor product to generalize the idea of matrix decomposition. There are multiple types of tensor products, such as Kronecker product, Khatri-Rao product [18], Hadamard product, n-mode product, etc. We will only mention Kronecker product and n-mode product in this paper, since this is what we will use later.

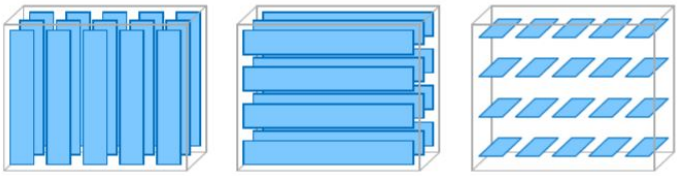

Fig. 2.0.1. Fibers of Third-Order Tensor

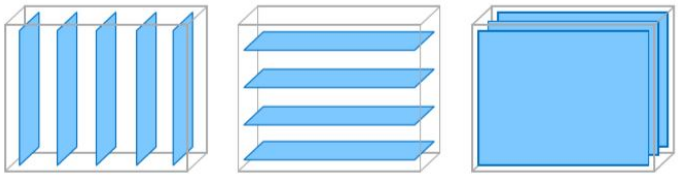

Fig. 2.0.2. Slices of Third-Order Tensor

Definition 2.0.3 (Kronecker Product) *The Kronecker product between two arbitrary matrices $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{K\times L}$ is defined as*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix}_{IK\times JL} \tag{2.0.1}$$

This yields that $A \otimes B \in \mathbb{R}^{IK\times JL}$.

Let us give an example to illustrate the idea. Say $A = \begin{bmatrix} 1 & -1 & 0 \\ -2 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}$ and $B = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \in \mathbb{R}^{2\times 1}$, then according to the definition, we can compute their Kronecker product as follows:

$$A \otimes B = \begin{bmatrix} 1 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} & -1 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} & 0 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\ -2 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} & 1 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} & 1 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 3 & -3 & 0 \\ 1 & -1 & 0 \\ -6 & 3 & 3 \\ -1 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 3} \tag{2.0.2}$$

If we generalize the definition of Kronecker Product on the field $\mathbb{R}$ to an abstract vector space, then we can define tensor product, where is still denoted as $\otimes$.

Definition 2.0.4 (Tensor Product) *Considering two vector spaces $\boldsymbol{V}$ and $\boldsymbol{W}$ on a field $\mathbb{K}$, tensor product is a bi-linear mapping $\boldsymbol{V} \times \boldsymbol{W} \to \boldsymbol{V} \otimes \boldsymbol{W}$, and no other restriction is imposed.*

If each of the two vector spaces is finite and has a basis, say $\{\boldsymbol{e}^i\}_{i=1}^m$ and $\{\boldsymbol{f}^j\}_{j=1}^n$, then $\forall \boldsymbol{v} \in \boldsymbol{V}$ and $\forall \boldsymbol{w} \in \boldsymbol{W}$ can be represented as

$$\boldsymbol{v} = \sum_{i=1}^{m} v_i \boldsymbol{e}^i, \boldsymbol{w} = \sum_{j=1}^{n} w_j \boldsymbol{f}^j \tag{2.0.3}$$

Thus by definition of tensor product, we can explicitly write out the expression of $\boldsymbol{v} \otimes \boldsymbol{w}$.

$$\boldsymbol{v} \otimes \boldsymbol{w} = \sum_{i=1}^{m} \sum_{j=1}^{n} (v_i w_j) \left(\boldsymbol{e}^i \otimes \boldsymbol{f}^j\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} (v_i w_j) \boldsymbol{g}_{ij} \tag{2.0.4}$$

The above is exactly how Kronecker product is computed. This is the exact reason why they are using the same notation $\otimes$. What's more, this also tells us the structure of product space, of which dimension is product of those of the two vector spaces:

$$\dim (\boldsymbol{V} \otimes \boldsymbol{W}) = \dim \boldsymbol{V} \times \dim \boldsymbol{W} \tag{2.0.5}$$

A n-mode product basically is a tensor multiplying with a matrix along with one specific dimension, which is the n-th dimension.

Definition 2.0.5 (n-mode Product) *The n-mode product $\times_n$ of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $M \in \mathbb{R}^{J \times I_n}$ is defined as*

$$(\mathcal{X} \times_n M)_{i_1 i_2 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_n \ldots i_N} m_{j i_n} \tag{2.0.6}$$

This yields that $\mathcal{X} \times_n M \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$.

Based on the definition, we can observe that n-mode product is actually to let M act on each fiber of the n-th dimension by matrix multiplication. Let us see an example. Suppose $\mathcal{X} = \left(\begin{bmatrix} 1 & 0 & -1 \\ 2 & -1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 2 & 1 \\ 3 & 0 & 1 \end{bmatrix} \right) \in \mathbb{R}^{2\times 2\times 3}$ and $M = \begin{bmatrix} 1 & 0 \\ 0 & -2 \\ -1 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 2}$. Then the 2-mode product of $\mathcal{X}$ and M can be computed as follows.

$$\begin{aligned} \mathcal{X} \times_2 M = & \left(\left[M \times \begin{pmatrix} 1 \\ 2 \end{pmatrix} \; M \times \begin{pmatrix} 0 \\ -1 \end{pmatrix} \; M \times \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right], \right. \\ & \left. \left[M \times \begin{pmatrix} -1 \\ 3 \end{pmatrix} \; M \times \begin{pmatrix} 2 \\ 0 \end{pmatrix} \; M \times \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right] \right) \\ = & \left(\begin{bmatrix} 1 & 0 & -1 \\ -4 & 2 & 0 \\ 1 & -1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & 1 \\ -6 & 0 & -2 \\ 4 & -2 & 0 \end{bmatrix} \right) \in \mathbb{R}^{2\times 3\times 3} \end{aligned} \tag{2.0.7}$$

3. RELATED WORK

There is quite a lot of work having been done on tensor decomposition [2, 19–21]. The most commonly used one is Canonical Polyadic Decomposition [1] as the prototype method of decomposition, a generalized version of Singular Value Decomposition (SVD) [22] which is deployed on matrices. To predict, we will also need to use Vector Autoregression model to forecast temporal component.

3.1 Tensor Decomposition: Canonical Polyadic Decomposition (CPD)

In tensor analysis, the most widely used tensor decompositions are Canonical Polyadic Decomposition (CPD) [1] and Turker decomposition [4]. They are both inspired by the idea of matrix decomposition, more specifically Singular Value Decomposition (SVD) [22].

Definition 3.1.1 (Singular Value Decomposition (SVD)) *For a $m \times n$ matrix A in a field $\mathbb{K}$, it can be always decomposed into the following form:*

$$A = U\Sigma V^\intercal \tag{3.1.1}$$

where U is an $m \times m$ unitary matrix ($UU^\intercal = I_m$), V is an $n \times n$ unitary matrix ($VV^\intercal = I_n$) and Σ is an $m \times n$ diagonal matrix, where only some of diagonal entries are not zero and the number of these entries are exactly the rank of A. Also those non-zero values are called the singular values of A, which are given by $A\vec{v} = \sigma\vec{u}$, where $\vec{v} \in \mathbb{K}^n$ and $\vec{u} \in \mathbb{K}^m$.

In tensor version, typical matrix product in (3.1.1) is replaced by n-mode product. It is usually advised to use CPD for latent parameter estimation and Tucker for subspace estimation, compression, and dimension reduction. CPD, short for Canonical Polyadic Decomposition, gives us a tool to "simplify" a complicated tensor, by

revealing hidden information in tensors. Also after decomposing, the tensor will be broken down into multiple matrices or vectors, which is much easier to interpret.

CPD is also called tensor rank decomposition, because it factorizes a tensor by its rank. In order to demonstrate the idea of it below, we need to give the definition of tensor rank. To begin with, we will introduce the idea of simple tensor.

Definition 3.1.2 (Simple Tensor) *For a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\mathcal{X}$ is called simple, if $\mathcal{X} \in \mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \cdots \otimes \mathbb{R}^{I_N}$ where $\otimes$ is denoted tensor product, which means there exist N vectors $\left\{x_k \in \mathbb{R}^{I_k}\right\}_{k=1}^{N}$, such that $\mathcal{X} = x_1 \otimes x_2 \otimes \cdots \otimes x_N$.*

This will allow us to define tensor rank.

Definition 3.1.3 (Tensor Rank) *For a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, if $\mathcal{X} = 0$, then $rank(\mathcal{X}) = 0$; if $\mathcal{X}$ is simple, then $rank(\mathcal{X}) = 1$. Otherwise, $rank(\mathcal{X}) = r$ is the smallest integer such that there exist r simple tensors $\{\mathcal{B}_i\}_{i=1}^{r}$ such that $\mathcal{X} = \sum_{i=1}^{r} \mathcal{B}_i$.*

Now we can talk about Canonical Polyadic Decomposition (CPD). For a nth-order tensor $\mathcal{X}$ with a rank of r, the Canonical Polyadic Decomposition of $\mathcal{X}$ is defined as

$$\mathcal{X} = \sum_{i=1}^{R} \lambda_i \mathcal{B}_i \tag{3.1.2}$$

where $R \geqslant r$ and every $\mathcal{B}_i$ is a simple tensor with rank 1. It can be easily observed that when $R = r$, the form is exactly the same as that which we define tensor rank by, and it is called minimal CPD or tensor rank decomposition. The key concept of rank decomposition is to express a tensor as the sum of a finite number of rank-one tensor. Figure 3.1.1 illustrates the conception of CPD on a third-order tensor $\mathcal{X}$.

As we mentioned previously, matrix decomposition is mostly non-unique. This is the reason why U and V in SVD have to be unitary to guarantee the uniqueness of SVD. CPD, on the other hand, is often unique, under the definition of tensor decomposition uniqueness that in (3.1.2), $\mathcal{B}_i$ is the only possible combination with scaling and permutation allowed [23]. This uniqueness helps greatly to tackle machine learning problems, since it will be much easier to implement.

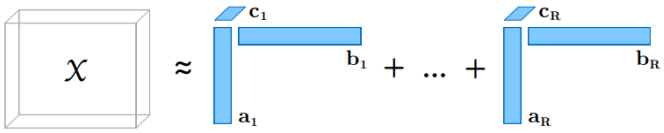

Fig. 3.1.1. Canonical Polyadic Decomposition (CPD)

There is no exact algorithm to determine tensor rank, since it is NP-complete [24]. However, there are several approximation algorithms to compute CP decomposition on a given tensor $\mathcal{X}$, such as Jennrich's Algorithm [25] and Alternative Least Square Algorithm (ALS) [23, 26]. Normally, ALS is more widely used, since Jennrich's Algorithm only uses slices of tensor instead of the full tensor and requires stricter restrictions on factor matrices [27]. Unlike Jennrich's, ALS Algorithm will fix all but one factor matrix and optimize it, and then repeat the procedure till it is convergent. In this paper, we will try to use this algorithm to find hidden information of the original tensor $\mathcal{X}$. One more thing worth noting is that the coefficient λ_i is normally neglected, since it can be merged into $\mathcal{B}_i$ without any side effects.

3.2 Vector Autoregression (VAR)

Based on the fact that the data-set is temporal, we need to use autoregression which is the time-series version of regression to forecast the future demand of taxis in New York City.

Temporal data has an important feature, that is, there is a high relation between data points, more specifically one data point depends on previous data points in term of time. Classical regression is insufficient to discover interesting trends because of that. Therefore, first of all, let us introduce the idea of ARMA(p, q) model [28], which is explained below:

Definition 3.2.1 (ARMA(p,q) Model) *For a time series $\{x_t\}$, the ARMA(p,q) model is defined as*

$$\begin{aligned} x_t &= \alpha + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q} \\ &= \underbrace{\left(\alpha + \sum_{i=1}^{p} \phi_i x_{t-i}\right)}_{AR\ Model} + \underbrace{\left(w_t + \sum_{j=1}^{q} \theta_j w_{t-j}\right)}_{MA\ Model} \end{aligned} \tag{3.2.1}$$

where p and q are the autogressive and the moving average orders, respectively. $\{w_t\}$ is a Gaussian white noise series with mean zero and variance σ_w^2. If $\{x_t\}$ has a nonzero mean μ, then $\alpha = \mu\left(1 - \sum_{i=1}^{p} \phi_i\right) \neq 0$.

To make the model valid, you have to check if the time series itself is causal and invertible, which means the model itself cannot predict based on the future data points and also unique. To address it, we have define to two polynomials:

Definition 3.2.2 (AR and MA Polynomials) *The AR polynomial $\phi(z)$ and MA polynomial $\theta(z)$ are defined as follows:*

$$\phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p,\ \phi_p \neq 0 \tag{3.2.2}$$

$$\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_q z^q,\ \theta_q \neq 0 \tag{3.2.3}$$

The causality and invertibility of ARMA(p,q) Model are based on the roots of both AR polynomial $\phi(z)$ and MA polynomial $\theta(z)$: the zeros of both polynomials have to lie outside the unit circle $|z| > 1$ on the complex plane, when $\phi(z)$ and $\theta(z)$ are treated as complex polynomials.

The above is the case in one dimension. However, as for in our case, each x_t has become a vector $\boldsymbol{x}_t$. For simplicity, we are only going to use the vector version of AR Model in the following sections as VAR(p) Model [29].

Definition 3.2.3 (VAR(p) Model) *For a multivariate time series $\{\boldsymbol{x}_t\}$ in d dimensions, the VAR(p) model is defined as*

$$\boldsymbol{x}_t = \boldsymbol{\phi}_0 + \sum_{i=1}^{p} \boldsymbol{\phi}_i \boldsymbol{x}_{t-i} + \boldsymbol{a}_t \tag{3.2.4}$$

where $\boldsymbol{\phi}_0$ *is a d-dimension constant vector, whereas* $\{\boldsymbol{\phi}_i\}_{i=1}^p$ *are all* $d \times d$ *matrices, and* $\boldsymbol{a}_t$ *is a white noise at time t with mean zero and covariance matrix* Σ_a*.*

The model is similar to the one-dimensional case. However, in terms of causality, it becomes a little bit different. Indeed, if we treat every $\boldsymbol{x}_t$ as d independent components x_{it}, then we can, of course, define causality on each x_{it}. But if these d series have influence on each other, it is hard to define a universal causality. Therefore, we introduce Granger's Causality [30]. In this framework, we focus on the relations between components within $\{\boldsymbol{x}_t\}$, where $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \cdots, x_{dt})$. We say that x_{it} causes x_{jt} if and only if the prediction or forecast of x_{jt} within x_{it} is more accurate than without, which means that x_{it} has "influence" on x_{jt}, which lead to at least one $\phi_{k,ji} \neq 0$ in $\boldsymbol{\phi}_k$ for $k = 0, 1, 2, \cdots, p$. Therefore, we can define more generally.

Definition 3.2.4 (Granger's Causality) *For a VAR(p) model* $\{\boldsymbol{x}_t\}$*, where* $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \cdots, x_{dt})$ *and* $\boldsymbol{x}_t = \boldsymbol{\phi}_0 + \sum_{i=1}^p \boldsymbol{\phi}_i \boldsymbol{x}_{t-i} + \boldsymbol{a}_t$*, we say* x_{it} *causes* x_{jt}*, if there exists at least one* $\phi_{k,ji} \neq 0$ *in* $\boldsymbol{\phi}_k$ *for* $k = 0, 1, 2, \cdots, p$*.*

3.3 Tensor Applications in Machine Learning

There have been a lot of works using tensor techniques to tackle machine learning problems, especially in data with temporal component and multi-relational data. Whenever the data-set can be represented into matrix, tensors can always provide a intuitive and straightforward way to model temporal component by spreading it as additional axis, making the original data-set a stack of facet data. This will allow the model to capture latent patterns in temporal manner, which is extremely helpful in recommendation systems [31–33], spotting anomalies [34] and etc. Also, tensors can be naturally applied to a streaming data, without dealing with infinite time [35].

Since tensors can be of any dimensions, it can also naturally be applied to multi-relational data. This enables us to find inter-dependencies simultaneously. For instance, it is very useful when it comes to statistical relational learning [36–38]. One prominent application in multi-relational data is knowledge graph network, which

stores information about relationships in the real world. Detecting hidden relations is the main issue, which tensor decomposition has shown state-of-the-art performance in both quality and efficiency [39, 40].

3.4 Previous Work on Spatio-temporal Data

Spatio-temporal data is a data-set with spatial and temporal components which have spatial attributes such as latitude and longitude and a temporal attribute. It becomes increasingly important these days when it comes to forecast future trends, especially in urban data, such as New York City taxi data [13]. Therefore, a lot of research works have been done on this kind of data. There are two main ways to train the data: one is using classical regression method, and the other is using neural network, mainly Recurrent Neural Network (RNN) [41, 42], since data contain temporal component. However, in this paper, we will focus on the former one, based on the reason that RNN is notoriously hard to train [43], which requires more data than classical methods, and also the outcome is not easy to interpret, since neural network itself is a black box. In terms of classical methods, typically Gaussian Process (GP) [14] and vector autoregression (VAR) are commonly used [44]. In terms of GP, it always assumes the output abides by Gaussian Process, where the key problem tackles a covariance matrix, which however demands a very expensive time cost. More recently, Bahadori et al. [15] have proposed a fast multivariate spatio-temporal analysis method (FMST), and Yu et al. [16] have developed a temporal regularized matrix factorization (TRMF) to figure out latent information behind the data-set as matrix format, but both of methods include only temporal component without considering spatial problems, since they both only use matrix decomposition in their methods. On the other hand, Qian et al [45] focused on spatial variation but not the forecasting. Takeuchi et al. [12] used tensor decomposition and DAR model, but they mainly focused on DAR model, which is directed autoregression by splitting the

2π angle at each location into multiple discrete directions, and trying to find out geographical dependence between adjacent locations.

4. METHODOLOGY

In this section, we will discuss about how to decompose a tensor which has temporal and spatial dimensions and predict data within the tensor along the temporal dimension. The whole idea is mostly inspired by the idea from Takeuchi et al. [12] who also use tensor decomposition and regression to find patterns in New York City Taxi Data but unfortunately has ignored the seasonality of the data-set.

We formulate our problem in the purpose of forecasting future needs of taxis in each designated location as follows. Given a spatio-temporal data-set of NYC taxi data [13] which consists of every taxi trip log including pick-up time, drop-off time, pick-up location, drop-off location and some parameters concerning tensor decomposition, our goal is to learn periodic patterns of taxi needs in each designated location by building a tensor model and then use this learned model to predict future trends in each taxi zone.

For a typical spatio-temporal data-set, for example, geographical and dynamic data with locations and timestamps, it is very common to be represented by multiple dimensions: time, space and other features. This is also true for the NYC taxi data-set, which indicates it is a third-order tensor data-set. For generalization, we define the data-set to be explored as $\mathcal{X} \in \mathbb{R}^{F \times T \times L}$, where F represents features, T represents timestamps, L represents locations, and also $x_{ijk} \in \mathcal{X}$ data points. The purpose of tensor decomposition is based on the idea of Spearmann's Hypothesis [7] in the condition of matrices.

Therefore, $\mathcal{X}$ should be decomposed into K latent components for each dimension, which means that n-th dimension is a matrix having order of $I_n \times K$. For instance,

a data-set $\mathcal{X} \in \mathbb{R}^{F\times T\times L}$ can be interpreted as three matrices $M^{(1)} \in \mathbb{R}^{F\times K}, M^{(2)} \in \mathbb{R}^{T\times K}, M^{(3)} \in \mathbb{R}^{L\times K}$ with a unit diagonal tensor $M^{(0)} \in \mathbb{R}^{K\times K\times K}$, meaning

$$\left(m^{(0)}\right)_{k_1,k_2,k_3} = \begin{cases} 1 & \text{if } k_1 = k_2 = k_3 \\ 0 & \text{otherwise} \end{cases} \tag{4.0.1}$$

where $m^{(0)}_{k_1k_2k_3}$ is the element of $M^{(0)}$ at the entry k_1, k_2, k_3. For clarification, in the following demonstration, $m^{(i)}_{k_1k_2k_3}$ always means the element of $M^{(i)}$ at the entry k_1, k_2, k_3.

By using the previously defined n-mode product, we can design the following loss function for tensor decomposition:

$$loss = \left\|\mathcal{X} - M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)}\right\| + \sum_{i=1}^{3} \lambda_i g_i(M^{(i)}) \tag{4.0.2}$$

which we need to minimize, and the last sigma is the regularization to restrict the choosing of matrices. In order to get familiar with n-mode product, we will prove the following result.

Proposition 4.0.1 *$M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)}$ is a 3rd-order tensor in $\mathbb{R}^{F\times T\times K}$, and we have*

$$\left(M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)}\right)_{f,t,l} = \sum_{k_1,k_2,k_3=1}^{K} m^{(0)}_{k_1k_2k_3} m^{(1)}_{fk_1} m^{(2)}_{tk_2} m^{(3)}_{lk_3} \tag{4.0.3}$$

$$= \sum_{k=1}^{K} m^{(1)}_{fk} m^{(2)}_{tk} m^{(3)}_{lk} \tag{4.0.4}$$

where $k_1, k_2, k_3 = 1, 2, 3, \cdots, K$.

Proof Since $M^{(0)} \in \mathbb{R}^{K^3}, M^{(1)} \in \mathbb{R}^{F\times K}$, then by definition of 1-mode product, $M^{(0)} \times_1 M^{(1)} \in \mathbb{R}^{F\times K\times K}$. For the same reason, $M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \in \mathbb{R}^{F\times T\times K}$.

Therefore, $M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)} \in \mathbb{R}^{F \times T \times L}$. Now we will see how to compute the product.

$$\begin{aligned} L.S. &= \sum_{k_3=1}^{K} \left(M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)}\right)_{ptk_3} m_{lk_3}^{(3)} \\ &= \sum_{k_3=1}^{K} \left(\sum_{k_2=1}^{K} \left(M^{(0)} \times_1 M^{(1)}\right)_{pk_2k_3} m_{tk_2}^{(2)} \right) m_{lk_3}^{(3)} \\ &= \sum_{k_3=1}^{K} \left(\sum_{k_2=1}^{K} \left(\sum_{k_1=1}^{K} m_{k_1k_2k_3}^{(0)} m_{fk_1}^{(1)} \right) m_{tk_2}^{(2)} \right) m_{lk_3}^{(3)} \\ &= \sum_{k_1,k_2,k_3=1}^{K} m_{k_1k_2k_3}^{(0)} m_{fk_1}^{(1)} m_{tk_2}^{(2)} m_{lk_3}^{(3)} = (4.0.3) \end{aligned} \tag{4.0.5}$$

Based on the definition of $M^{(0)}$ in (4.0.1), we can further simply the expression (4.0.5), since the terms in summation are all zeros except when $k_1 = k_2 = k_3$

$$\begin{aligned} (4.0.5) &= \sum_{k_1,k_2,k_3=1}^{K} m_{k_1k_2k_3}^{(0)} m_{fk_1}^{(1)} m_{tk_2}^{(2)} m_{lk_3}^{(3)} \\ &= \sum_{k=1}^{K} m_{fk}^{(1)} m_{tk}^{(2)} m_{lk}^{(3)} = (4.0.4) \end{aligned} \tag{4.0.6}$$

■

The proposition (4.0.1) gives us an insight about how CPD works. The rank K, which is a latent factor, becomes the column number of matrix in each axis, and the matrix in each axis corresponds to information concerning that specific dimension.

Since right now, the data-set has been decomposed or interpreted into three matrices as $M^{(1)} \in \mathbb{R}^{F \times K}, M^{(2)} \in \mathbb{R}^{T \times K}, M^{(3)} \in \mathbb{R}^{L \times K}$, we can build our prediction based on this tensor decomposition to forecast future demands of taxis in NYC.

Considering our objective in this paper is to forecast future demands of taxis in New York City, therefore the rest of axes: locations and features should remain the same, since in a short time, features and locations will not change too much. This indicates that we only need to predict the axis of time $M^{(2)} \in \mathbb{R}^{T \times K}$. More precisely, we will be needed to count every taxi service in every location in New York City,

where will be later the numbers in entries of tensor. Since the count is non-negative by all means, thus it is wise to assume those factor matrices we will be getting are non-negative in all entries. This yields that we can define our penalty functions as indicator functions $\mathbf{1}_{\mathbb{R}_-}(\cdot)$ [46].

$$g_i(M^{(i)}) = \sum_{j,k} \mathbf{1}_{\mathbb{R}_-}(m^{(i)}_{j,k}) \tag{4.0.7}$$

where

$$\mathbf{1}_{\mathbb{R}_-}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{R}_- \\ 0 & \text{if } x \notin \mathbb{R}_- \end{cases} \tag{4.0.8}$$

Moreover, we want to make sure it is impossible to take negative entries in factor matrices, which leads to define all the coefficients $\lambda_i = +\infty$, which indicates that the whole regularization is actually a characteristic function $\chi_{\mathbb{R}_-}(\cdot)$ [47]:

$$\chi_{\mathbb{R}_-}(x) = \begin{cases} +\infty & \text{if } x \in \mathbb{R}_- \\ 0 & \text{if } x \notin \mathbb{R}_- \end{cases} \tag{4.0.9}$$

Thus, the loss function can be rewritten as

$$loss = \left\| \mathcal{X} - M^{(0)} \times_1 M^{(1)} \times_2 M^{(2)} \times_3 M^{(3)} \right\| + \sum_{i=1}^{3} \chi_{\mathbb{R}_-}(M^{(i)}) \tag{4.0.10}$$

Based on the discussion in previous sections, we will use ALS algorithm to find these factor matrices. Algorithm (1) shows how it works.

After factorization, we can operate on prediction. Since $M^{(2)}$ represents temporal factors, we need to "extend" our $M^{(2)}$ by having more timestamps. It can be assumed that the near future should be developed by data from the past. This brings vector autoregression model (VAR) into place. In our case, the time series $\{\boldsymbol{x}_t\}$ is defined as

$$\boldsymbol{x}_t = \begin{bmatrix} m^{(2)}_{1t} \\ m^{(2)}_{2t} \\ \vdots \\ m^{(2)}_{kt} \end{bmatrix} \tag{4.0.11}$$

Algorithm 1 ALS Algorithm

procedure CPD-ALS($\mathcal{X}, K$)
    Initialize $M^{(1)} \in \mathbb{R}_+^{F\times K}, M^{(2)} \in \mathbb{R}_+^{T\times K}, M^{(3)} \in \mathbb{R}_+^{L\times K}$ // Non-negative factor matrices
    repeat
        for $i = 1, 2, 3$ **do**
            Optimize non-negative $M^{(i)}$ while the rest fixed // By using method in [48]
        end for
    until loss function 4.0.10 stops to improve
    return $M^{(1)}, M^{(2)}, M^{(3)}$
end procedure

Thus, $M^{(2)}$ can be broken down as a k-dimensional time series; note that in order to explain better, $M^{(2)}$ has been transposed, and $M^{(2)}_{pred}$ is the prediction based on VAR model.

$$\left[M^{(2)} \mid M^{(2)}_{pred}\right]^{\intercal} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2} \cdots) \tag{4.0.12}$$

However, since the daily taxi data is largely following a pattern every day, which means there should be a frequency of 24 hours within the factor matrix. By adding seasonal adjustment [49], we can make the VAR model more reasonable. Given that thought and the seasonal period p, then we can add a seasonal dummy variable $\boldsymbol{sd}$ into the model we previously present, where

$$\boldsymbol{sd}_t = \boldsymbol{c}_s, \text{where } s = t \bmod p \tag{4.0.13}$$

In this expression, $\boldsymbol{c}$ will be a $K \times 1$ constant vector, no matter how t changes. Therefore, the model can be inferred as below:

$$\boldsymbol{x}_t = \boldsymbol{\phi}_0 + \sum_{i=1}^{p} \boldsymbol{\phi}_i \boldsymbol{x}_{t-i} + \boldsymbol{sd}_t + \boldsymbol{a}_t \tag{4.0.14}$$

This implies that $\boldsymbol{x}_p$ and $\boldsymbol{x}_{t-p}$ should have the same seasonal dummy variable: this is exactly what we need.

After the prediction, we can recover the cube of three dimensions by using the same factor matrices $M^{(1)}$ and $M^{(3)}$ with the new prediction of $M^{(2)}_{pred}$. Figure 4.0.1 illustrates how the model pipeline is built, in which F represents feature component, L locations, T time and K represents latent factors.

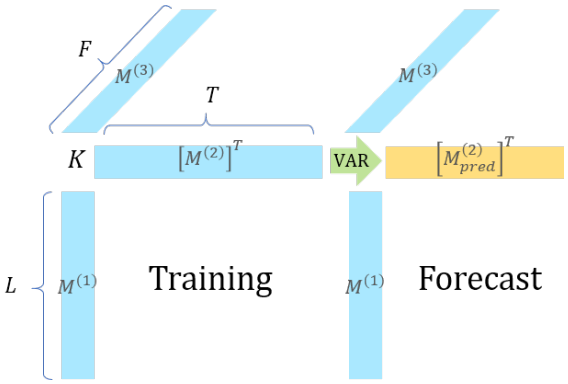

Fig. 4.0.1. Model Pipeline

5. CASE STUDY

We conducted a case study with NYC Taxi and Commission data [13] to predict the number of pickups and drop-offs in the designated zones.

5.1 Data Description

There are two types of taxis in New York City: yellow and green. Yellow taxis are those taxis which are traditionally hailed in every part of New York City, while green taxis are only available in all areas except Mid-town, Manhattan and airports. To train the model, we selected trips of yellow taxis and green taxis during the second seven days of November 2019 as training set and forecast the following two days to evaluate the model we built. The number of training records is 3,534,248 and the number of test records is 1,096,253. The attributes of trip records include pick-up and drop-off times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

5.2 Data Exploration

To get more familiar with the experiment data, we first explored the data-set manually using data visualization. The area of New York City have been divided into 265 zones, and pick-up and drop-off locations are designated by those indices. Also, the geographical information of each zone is stored in a ShapeFile directory. By loading these data, we had the maps of zones and boroughs of New York City as shown in Figure 5.2.1.

Furthermore, we loaded the data of taxis from November 2019 to December 2019 to see which zones were the most popular by grouping all the zones.

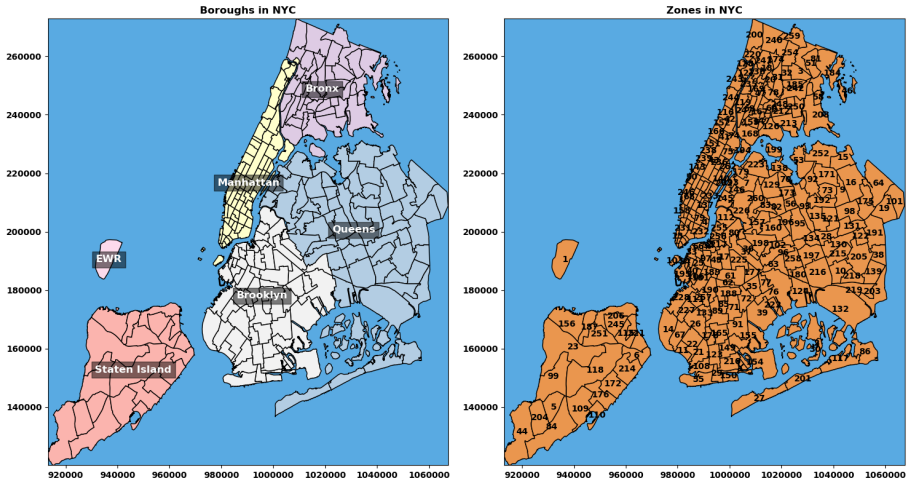

Fig. 5.2.1. Boroughs and zones of NYC through shapefile

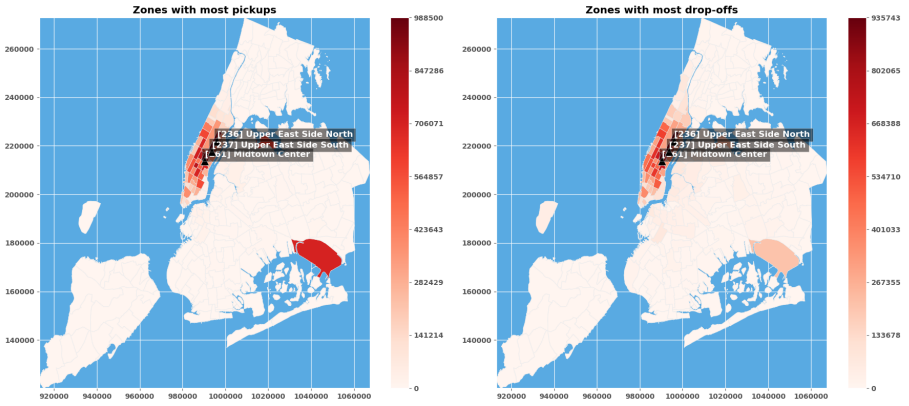

Fig. 5.2.2. Popular zones in NYC

As shown in Figure 5.2.2, the most popular areas are mostly in downtown and Manhattan. There are two exceptions, which are in the right bottom and the right top of the map. These two zones represent JFK International Airport and LaGuardia International Airport, which makes perfect sense. The rest of areas are mainly white gray, meaning there are much fewer street-hails for taxis, compared with those above.

To forecast future daily demand of taxis, it is wise to take a look at the daily pattern of taxis. We divided taxi trips into two groups based on their distance (threshold we chose here is 30 miles), and counted the trips in a hourly basis. Figure 5.2.3 shows

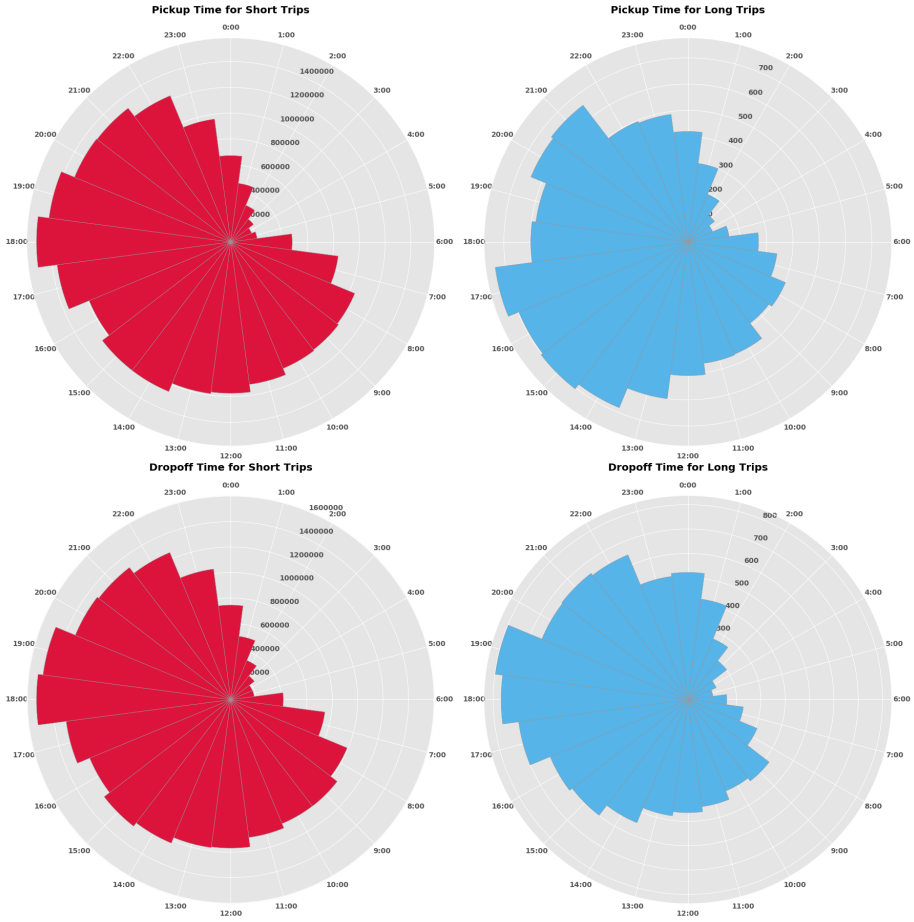

Fig. 5.2.3. Daily Patterns of Pick-up and Drop-off time of Taxi Trips

that long trips and short trips are largely following the same pattern. The peak hours are both around 17:00 to 21:00, when people go back to home or maybe hang out with friends.

5.3 Data Pre-processing

Based on what we had learned in data exploration, we used zone ID's as spatial dimension, timestamps as temporal dimension and other features as the third dimension. These constructed a third-order tensor. Each entry of tensor indicated the corresponding counts given certain time, location and features. For discretion in time, we counted taxi trips on a half-hour basis, which is not too small or too big.

Since taxi trips were counted without considering more details about trips themselves, there were only a few columns within the data-set that were used: pickup time, drop-off time and location ID's for each. These four features later formed four columns in the third dimension of tensor. After selecting these columns, the data-set looked like Table 5.3.1. All the original data-sets was first transformed into this.

Table 5.3.1.
An example of processed after selecting columns

_index	pickup_datetime	dropoff_datetime	PULocationID	DOLocationID
0	2019-11-01 0:25:27	2019-11-01 0:33:52	74	263
1	2019-11-01 0:39:13	2019-11-01 0:46:38	75	74
2	2019-11-01 0:55:35	2019-11-01 1:00:29	75	74
3	...	...	...	...

After selecting the features, we pre-processed our data-set from there to build our tensor. Figure 5.3.1 shows how we converted the pre-processed table into a tensor that we needed to further explore for CPD. It is worthwhile to note that in the procedure of "Turn into spreadsheet", we converted location ID into new variables for every feature (pick-up yellow, drop-off yellow, pick-up green, drop-off green), which means there were 265 columns responsible for each taxi zone: this is how we created the spatial dimension as an example in Table 5.3.2. Therefore, we had four spreadsheets concerning each feature, which would later be stacked together into a tensor.

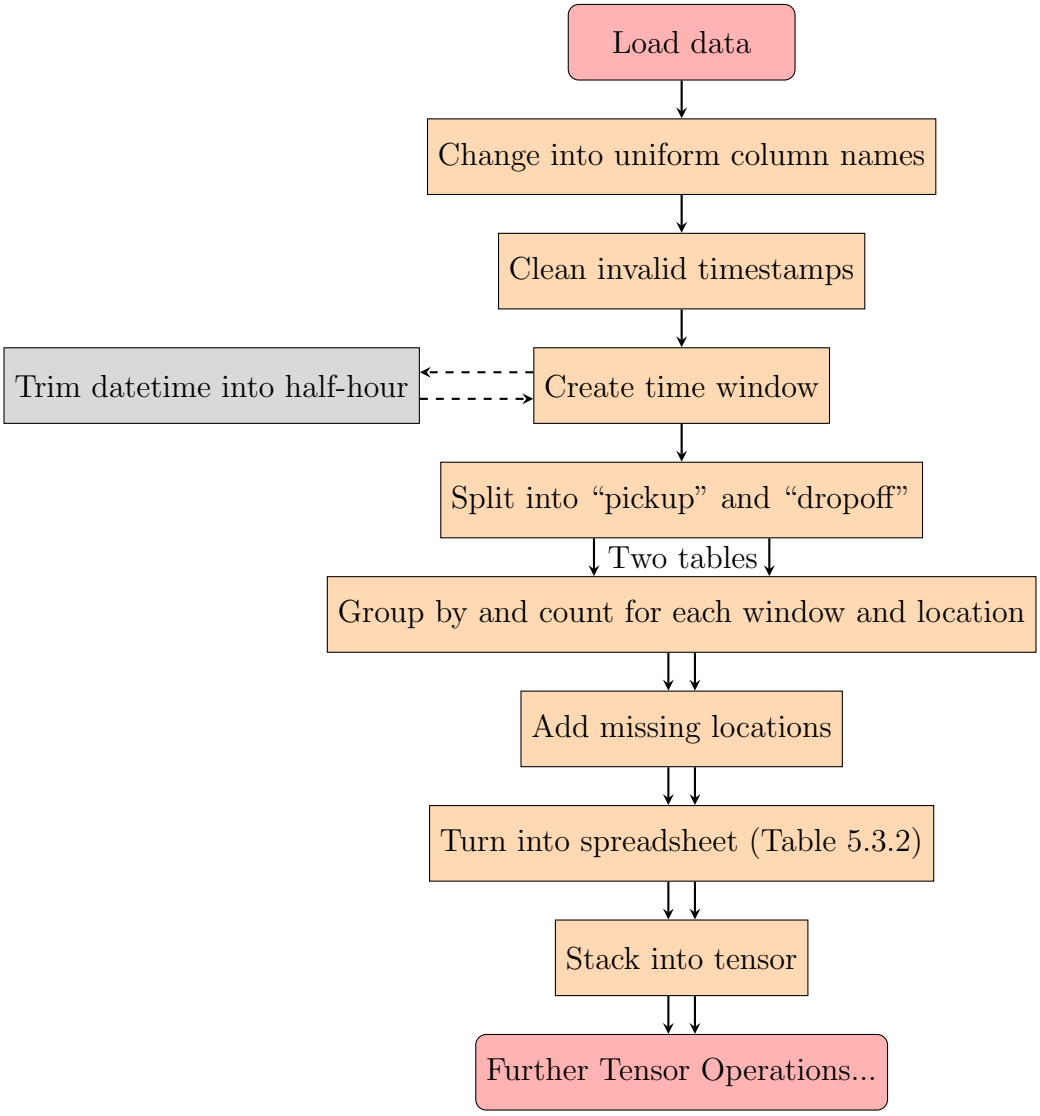

Fig. 5.3.1. Pre-processing Pipeline

After pre-processing the data-set, we had a tensor containing all the trip counts concerning each time window, each location and each feature. Since we used 30 minutes as time window, therefore the tensor $\mathcal{X}$ was a third-order tensor with $4 \times 336 \times 265$ entries.

Table 5.3.2.
An example of spreadsheet for pick-up data from yellow taxi

Time \ **Location**	**0**	**1**	**2**	**3**	$\cdots$	**264**
2019-11-01 0:00	0	0	0	35	$\cdots$	1
2019-11-01 0:30	0	0	0	29	$\cdots$	2
2019-11-01 1:00	0	0	0	20	$\cdots$	5
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

5.4 Experiment Setting

The latent factor K we chose in the model for CPD was 2, since in theory, this factor should represent the rank of tensor $\mathcal{X}$, which should be no greater than the number in each dimension. This means $K \leqslant 4$, but for CPD, a much less K should be chosen. That is why we chose $K = 2$.

In our implementation, we used Google Colaboratory platform [50] for tensor analysis and all other operations including vector autoregression. It provides free GPU resources to enhance performance throughout the analysis. For programming language, we used Python3 for all operations concerning tensors and pre-processing, and R to deploy Vector Autoregression (VAR), since time series is much better handled in R.

5.5 Evaluation Metrics

When building a VAR model, the lag parameter p needs to be determined. Here we compared the four estimators of Akiake Information Criterion (AIC) [51,52], Hannan–Quinn Information Criterion (HQ) [53], Schwarz Information Criterion (SIC) [54] and Final Prediction Error (FPE) [55, 56] to determine p. The less all the four estimators are, the better the model fits.

After building the model, we evaluated it. Suppose the predicted tensor is $\widehat{\mathcal{X}} \in \mathbb{R}^{F\times\tilde{T}\times L}$ and the ground truth tensor is $\mathcal{X} \in \mathbb{R}^{F\times\tilde{T}\times L}$, which shows real counts of taxi trips in the following two days. Then we used relative error (RE) [57] to determine whether the model was good or not.

$$RE = \frac{\sum_{f,\hat{t},l} |\hat{x}_{f,\hat{t},l} - x_{f,\hat{t},l}|}{\sum_{f,\hat{t},l} x_{f,\hat{t},l}} \tag{5.5.1}$$

5.6 Building Model

After implementing CPD with $K = 2$, we had three matrices $M^{(1)}, M^{(2)}, M^{(3)}$ representing factors of features, time and locations, respectively. Later on, we dealt with temporal factor matrix $M^{(2)} \in \mathbb{R}^{336\times 2}$. The graph below shows two temporal factors as time series format. From Figure 5.6.1, we could find an apparent daily pattern within each factor. Note that the beginning day of the training period was November 8th, 2019, which was Friday. Hence interestingly, f_1, which is the first temporal factor, explained some hidden information concerning business work, since there were peaks repeating on daytime every weekday, and also lower peaks on weekends, whereas f_2 somehow explained demand for entertainment and nightlife of people, since the highest peaks were on Friday night and Saturday night and much lower but still significant spikes on every evening.

To predict future demands, we forecast the temporal matrix. Since the repeating pattern was daily based, thus the length of seasonal dummy variables we chose was

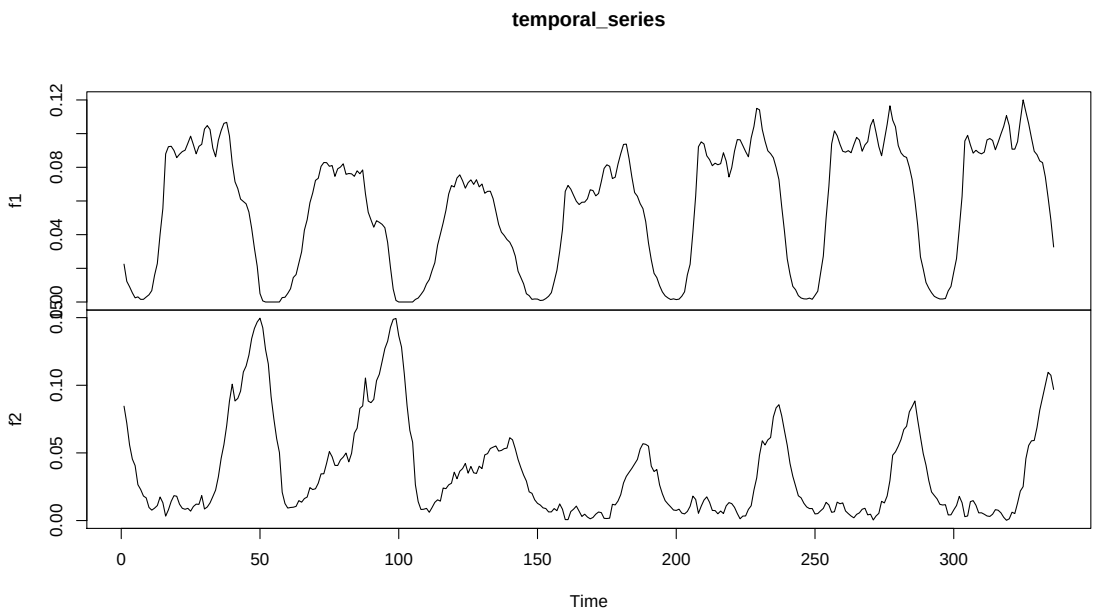

Fig. 5.6.1. Temporal Factor Matrix in Time Series Format

48. (Time window is half an hour.) We got the following result in Table 5.6.1: Since all the four estimators are smaller the better, we used $p = 7$ for our lag parameter. Thus the VAR model we used was VAR(7) with seasonal dummy variable 48. Figure 5.6.2 shows the forecasting result, and seasonal coefficients are shown in Figure 5.6.3. From seasonal coefficients shown, we observed that those of both temporal factors had a spike (only concerned absolute values) around 8:00 - 9:00 in the morning, also f_1 had a mild peak in early evenings while f_2 had the same but at late nights, which accorded with our previous observation that f_1 explained business information while f_2 explained entertainment information. From the predictions of temporal factors, we could clearly see that by using VAR model with seasonal variables, it could learn the daily pattern. Finally we transformed all factor matrices into a predicted tensor by using the same n-mode product we had used before to employ CPD.

Table 5.6.1.
VAR Model Selection

p	1	2	3	4
AIC(p)	$-2.106932\text{e}+01$	$-2.163369\text{e}+01$	$-2.169418\text{e}+01$	$-2.168991\text{e}+01$
HQ(p)	$-2.060576\text{e}+01$	$-2.115159\text{e}+01$	$-2.119354\text{e}+01$	$-2.117073\text{e}+01$
SIC(p)	$-1.990769\text{e}+01$	$-2.042560\text{e}+01$	$-2.043963\text{e}+01$	$-2.038889\text{e}+01$
FPE(p)	$7.109373\text{e}-10$	$4.045714\text{e}-10$	$3.810760\text{e}-10$	$3.829823\text{e}-10$
p	5	6	7	8
AIC(p)	$-2.171945\text{e}+01$	$-2.177926\text{e}+01$	$\mathbf{-2.188674\text{e}+01}$	$-2.187454\text{e}+01$
HQ(p)	$-2.118172\text{e}+01$	$-2.122299\text{e}+01$	$\mathbf{-2.131193\text{e}+01}$	$-2.128119\text{e}+01$
SIC(p)	$-2.037196\text{e}+01$	$-2.038531\text{e}+01$	$\mathbf{-2.044632\text{e}+01}$	$-2.038766\text{e}+01$
FPE(p)	$3.721245\text{e}-10$	$3.508108\text{e}-10$	$\mathbf{3.153430\text{e}-10}$	$3.195172\text{e}-10$
p	9	10		
AIC(p)	$-2.186388\text{e}+01$	$-2.188162\text{e}+01$		
HQ(p)	$-2.125198\text{e}+01$	$-2.125118\text{e}+01$		
SIC(p)	$-2.033053\text{e}+01$	$-2.030181\text{e}+01$		
FPE(p)	$3.232707\text{e}-10$	$3.179289\text{e}-10$		

5.7 Experiment Results

Results in Key Zones In order to evaluation our model, we first chose some key zones in New York City and computed their relative errors. The list of key Zones is shown as Table 5.7.1. After recovering the tensor, we could finally evaluate our model as shown in Figure 5.7.1. From the figure, we can observe that for those key zones, the relative errors are generally small. However, there are some outliers on the top part of the graph: all are bigger than 1. Since they were computed as relative errors, this implied that the scopes of error in these zones were even bigger than those of original data. This seemed unacceptable, but further investigation told us that those zones all had extreme low demands of taxis: the suburb area of NYC. It

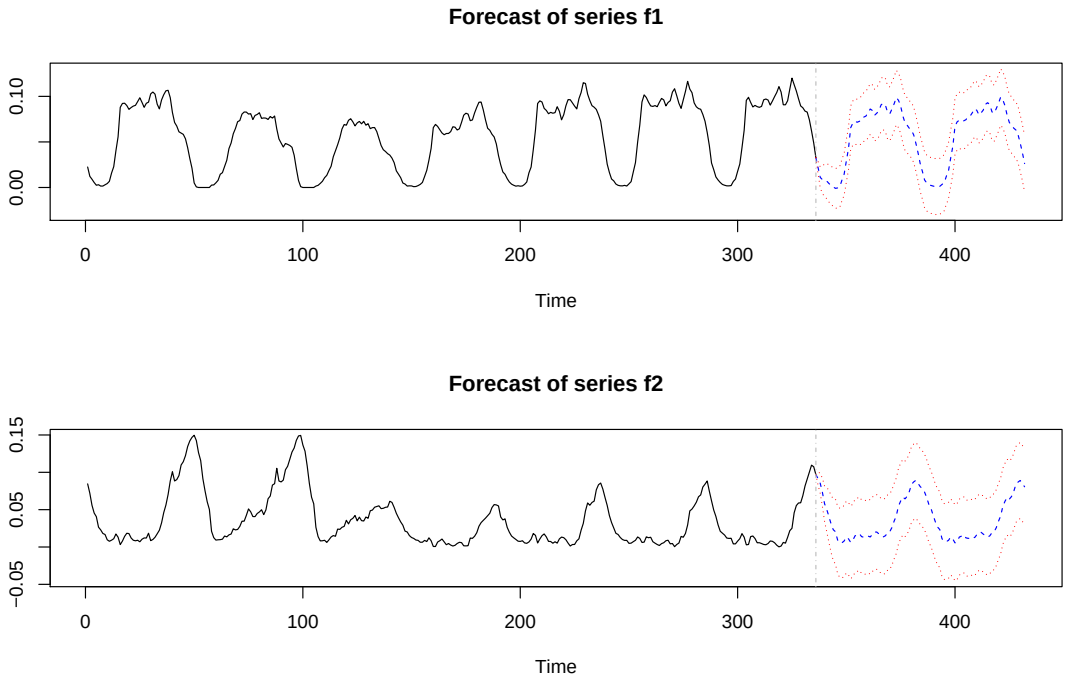

Fig. 5.6.2. Forecast of Temporal Factors

was because the scopes of data were too small that made relative errors meaningless and also the predictions. Figure 5.7.2 shows how one of those zones having the most demands of taxis is predicted. Note that Upper Eastside N. is in Manhattan, where a lot of business occurs. From the graph, we find out that green taxis are much worse predicted than the yellow ones, which is understandable, since green taxis are not allowed in this area, yielding that the scope of data is too small, which leads to the meaninglessness of prediction. On the other hand, both pick-up and drop-off predictions of yellow taxis capture the daily pattern and are well fit.

It seems that the model works well in the central business district, but the prediction of the airport is slightly worse as shown in Figure 5.7.3. The main reason is because the daily pattern at the airport is very different from other places: there are no two factors concerning business and entertainment as mentioned early, and there

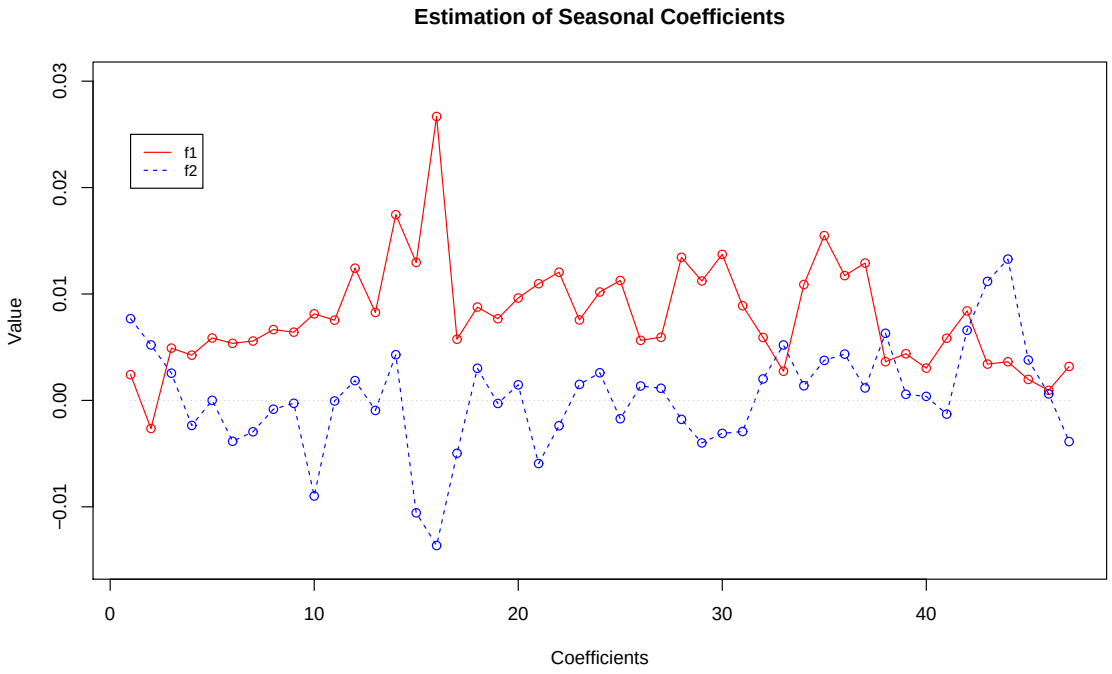

Fig. 5.6.3. Seasonal Coefficients

are literally no street-hails at the airport, which is very different from how things work in Manhattan.

Comparison with other models We compared our proposed model with other models that were used in New York Taxi Data, such Gaussian Process [14], FMST [15], TRMF [16] and DAR [12] by relative errors. The previous three models have been tested systematically and reviewed by Takeuchi et al. Only Takeuchi et al. use tensor methods, while the rest focus on matrix methods. Table 5.7.2 shows the testing results concerning RE. In the table, it is obvious to see that except DAR model, our proposed model records the best behavior concerning relative error. This gives us some insights that tensor methods, especially tensor decomposition, work much better than other methods, when it comes to spatio-temporal data, since the data naturally are better displayed in tensor. Therefore, it is reasonable that tensor methods are more capable

Table 5.7.1.

List of Key Zones

ID	Zone Name
47	Clinton East
99	Garment District
131	JFK International Airport
137	LaGuardia International Airport
185	Penn Station
235	Upper Eastside North
236	Upper Eastside South
260	Midtown Center

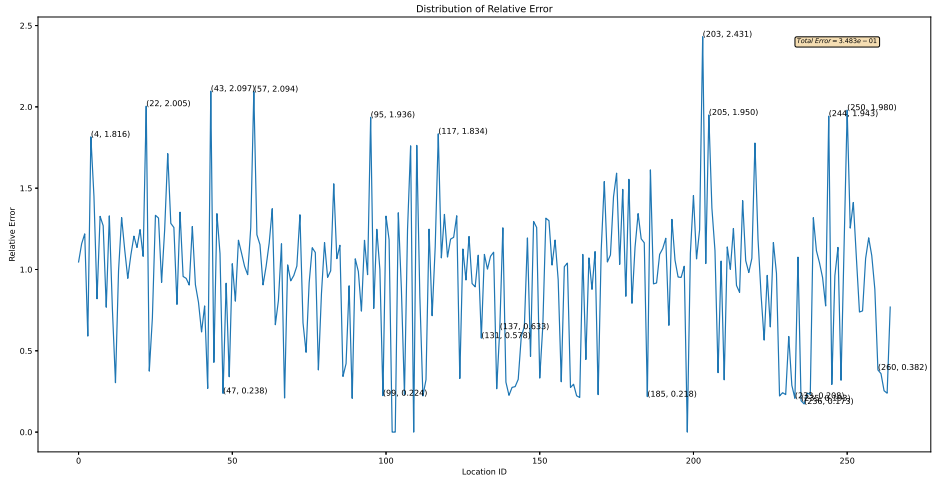

Fig. 5.7.1. Relative Error for Each Zone

of capturing latent inter-dependencies. However, our model performances not as well as DAR model. The main reason behind it is perhaps in our model did not apply

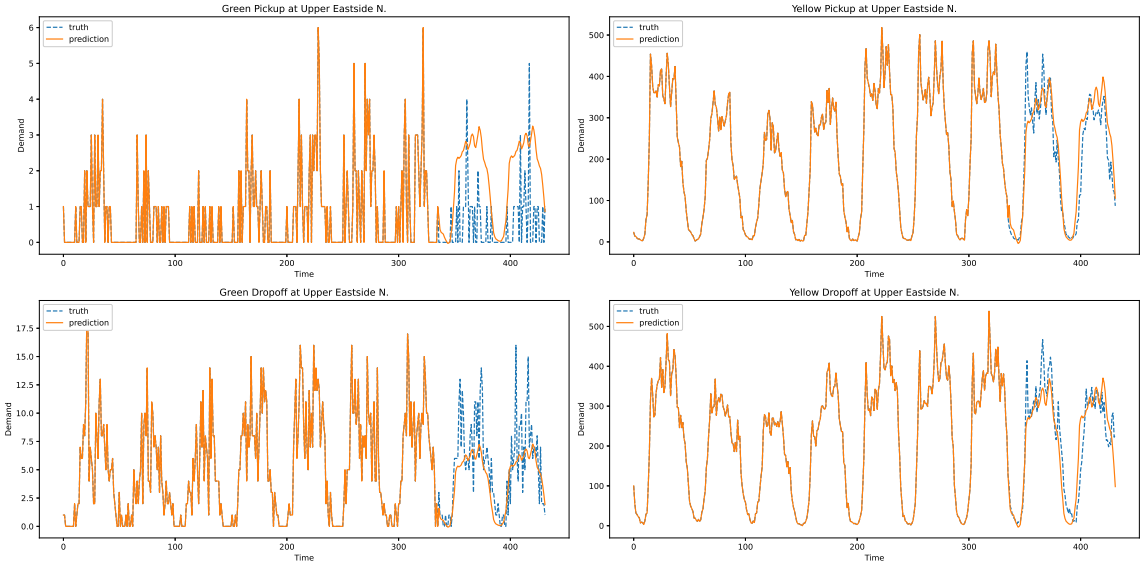

Fig. 5.7.2. Taxi Prediction at Upper Eastside N.

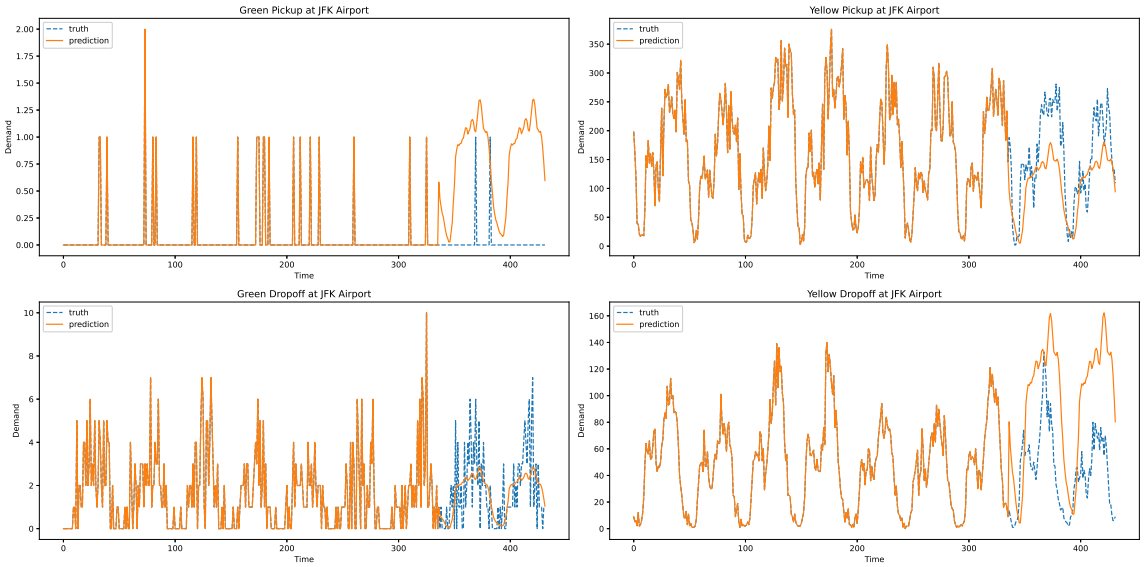

Fig. 5.7.3. Taxi Prediction at JFK Airport

geographical correlation explicitly as DAR model does: we only implicitly learned the spatial dependency during tensor decomposition. When we imposed CPD on the

Table 5.7.2.
Comparison of Different Models

Model	GP	FMST	TRMF	DAR	Our Model
Relative Error	1.683	0.653	0.350	0.312	0.348

tensor, overlooking this correlation might have some impact on the model, which led to worse performance.

Causality As we mentioned before, the two temporal factors represent two different kinds of information: business and entertainment. It seems that they are not correlated with each other. To validate that, we employed F-type Granger-causality [30] test and Wald-type test [58] to see if p-value is significant. The result is shown in Table 5.7.3. It is clearly to see all the p-values are not significant, which means that

Table 5.7.3.
Causality Test

H0: There is no Granger-cause.		
p-value	f_1 versus f_2	f_2 versus f_1
Granger Test	$2.354e-04$	$4.770e-07$
Wald Test	$1.623e-05$	$1.623e-05$

the hypothesis $H0$ is refused, meaning there is no Granger-cause between these two temporal factors. This implies that they are not correlated and it is possible to predict them separately as uni-variate time series.

6. CONCLUSION

In this paper, we proposed a combined model consisting of CPD and VAR Model with seasonal variables to find latent daily patterns of taxi demands in each taxi zone within NYC Taxi Data, and therefore used that information to forecast future demands of taxis. The result produced by the proposed model is the best among previous models except DAR, which uses geographical inference. Our model compared with other models is more straightforward. We believe that by introducing seasonal variables can help improve results, especially in temporally periodic data-set of large scale like NYC Taxi Data-set.

For our model, there is still a lot of work that is worth exploring in the future. First of all, we have not considered geographical inference in our model, which is a potential way to improve our model, given the smaller relative error in DAR model. Secondly, because taxis are mainly served in the areas having dense population, it is potentially not wise to include those suburban zones where taxis are rare. Perhaps it is better to exclude those zones, and also predictions in those areas should not be trusted due to limited scopes of data. Furthermore, VAR model is a simplified version of a more complicated model VARMA model, which introduces moving average of white noises. This will be interesting to try out. Last but not least, in our model, there are typically two models: CPD and VAR. To achieve the pipeline, we estimated the two models separately, that is, we chose the best estimation of the first, and then based on this estimation, we optimized the next. This would potentially lead to the problem of local optimum. It may be better to estimate two models together, and then forecast, which potentially needs global optima of all parameters. This could be a very interesting problem to dig into: how to combine two models together to produce a globally optimal outcome. It will definitely need more theoretical deductions, which still remains open.

REFERENCES

REFERENCES

[1] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[5] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in Measuring Change*, vol. 15, pp. 122–137, 1963.

[6] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, "Matrix product state representations," *arXiv preprint quant-ph/0608197*, 2006.

[7] A. R. Jensen, "The nature of the black–white difference on various psychometric tests: Spearman's hypothesis," *Behavioral and Brain Sciences*, vol. 8, no. 2, pp. 193–219, 1985.

[8] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[9] L. Anselin, "Thirty years of spatial econometrics," *Papers in Regional Science*, vol. 89, no. 1, pp. 3–25, 2010.

[10] P. J. Curran and P. M. Atkinson, "Geostatistics and remote sensing," *Progress in Physical Geography*, vol. 22, no. 1, pp. 61–78, 1998.

[11] L. A. Waller and C. A. Gotway, *Applied spatial statistics for public health data*. John Wiley & Sons, 2004, vol. 368.

[12] K. Takeuchi, H. Kashima, and N. Ueda, "Autoregressive tensor factorization for spatio-temporal predictions," in *Proceedings of 2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 1105–1110.

[13] "New york city taxi and limousine commission dataset," https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, accessed: 2020-04-20.

[14] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[15] M. T. Bahadori, Q. R. Yu, and Y. Liu, "Fast multivariate spatio-temporal analysis via low rank tensor learning," in *Advances in Neural Information Processing Systems*, 2014, pp. 3491–3499.

[16] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in Neural Information Processing Systems*, 2016, pp. 847–855.

[17] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to tensor decompositions and their applications in machine learning," *arXiv preprint arXiv:1711.10781*, 2017.

[18] C. Khatri and C. R. Rao, "Solutions to some functional equations and their applications to characterization of probability distributions," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 167–180, 1968.

[19] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—online stochastic gradient for tensor decomposition," in *Conference on Learning Theory*, 2015, pp. 797–842.

[20] G. W. McNeice and A. G. Jones, "Multisite, multifrequency tensor decomposition of magnetotelluric data," *Geophysics*, vol. 66, no. 1, pp. 158–173, 2001.

[21] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of eeg signals: a brief review," *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015.

[22] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[23] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[24] C. J. Hillar and L.-H. Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–39, 2013.

[25] R. I. Jennrich, "Orthogonal rotation algorithms," *Psychometrika*, vol. 35, no. 2, pp. 229–235, 1970.

[26] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[27] D. Hsu, S. M. Kakade, and T. Zhang, "A spectral algorithm for learning hidden markov models," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1460–1480, 2012.

[28] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2017.

[29] R. S. Tsay, *Multivariate time series analysis: with R and financial applications*. John Wiley & Sons, 2013.

[30] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.

[31] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010, pp. 81–90.

[32] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, 2010, pp. 79–86.

[33] E. Frolov and I. Oseledets, "Tensor methods and recommender systems," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 3, p. e1201, 2017.

[34] E. Papalexakis, K. Pelechrinis, and C. Faloutsos, "Spotting misbehaviors in location-based social networks using tensors," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 551–552.

[35] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 374–383.

[36] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in *Proceedings of the 28th International Conference on Machine Learning*, vol. 11, 2011, pp. 809–816.

[37] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski, "A latent factor model for highly multi-relational data," in *Advances in Neural Information Processing Systems*, 2012, pp. 3167–3175.

[38] E. E. Papalexakis, L. Akoglu, and D. Ience, "Do more views of a graph help? community detection and clustering in multi-graphs," in *Proceedings of the 16th International Conference on Information Fusion*. IEEE, 2013, pp. 899–905.

[39] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing yago: scalable machine learning for linked data," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 271–280.

[40] A. Padia, K. Kalpakis, and T. Finin, "Inferring relations in knowledge graphs with tensor decompositions," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 4020–4022.

[41] F. Rodrigues, I. Markou, and F. C. Pereira, "Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach," *Information Fusion*, vol. 49, pp. 120–129, 2019.

[42] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5668–5675.

[43] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1310–1318.

[44] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *Advances in Neural Information Processing Systems*, 2008, pp. 153–160.

[45] X. Qian and S. V. Ukkusuri, "Spatial variation of the urban taxi ridership using gps data," *Applied Geography*, vol. 59, pp. 31–42, 2015.

[46] B. Folland, "Modern techniques and their applications," *Real Analysis (Pure and Applied Mathematics)*, 1999.

[47] R. T. Rockafellar, *Convex analysis*. Princeton university press, 1970, no. 28.

[48] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd International Conference on Machine learning*, 2005, pp. 792–799.

[49] E. Ghysels and D. R. Osborn, *The econometric analysis of seasonal time series*. Cambridge University Press, 2001.

[50] E. Bisong, "Google colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 59–64.

[51] H. Akaike, "Statistical predictor identification," *Annals of the Institute of Statistical Mathematics*, vol. 22, no. 1, pp. 203–217, 1970.

[52] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[53] E. J. Hannan and B. G. Quinn, "The determination of the order of an autoregression," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 2, pp. 190–195, 1979.

[54] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[55] H. Akaike, "Fitting autoregressive models for prediction," *Annals of the Institute of Statistical Mathematics*, vol. 21, no. 1, pp. 243–247, 1969.

[56] H. Akaike, "Autoregressive model fitting for control," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 153–170.

[57] A. D. Helfrick and W. D. Cooper, *Modern electronic instrumentation and measurement techniques*. Prentice Hall Englewood Cliffs, NJ, 1990.

[58] V. Martin, S. Hurn, and D. Harris, *Econometric modelling with time series: specification, estimation and testing*. Cambridge University Press, 2013.