# ABSTRACT


Title of Thesis: ANALYSIS AND FORECASTING FOR TRAFFIC FLOW DATA

Yitian WANG Masters of Science 2017

Thesis Directed By: Professor Joseph JaJa
Department of Electrical and Computer Engineering



In this thesis, a number of techniques related to Principal Component Analysis (PCA) are used to derive core traffic patterns from streams of traffic data on a large number of road segments. Using a few number of *k* hidden variables, we show that the traffic information on the road segments can be captured by *k* traffic patterns. The dimensionality of the correlated road segments is successfully reduced from *n* to a much smaller number *k* by applying techniques related to Principal Component Analysis (PCA), where *n* is the number of road segments and *k* is the number of hidden variables. We use the *k* nearest neighbor(KNN) method to predict the values of the hidden variables over small time windows. As a result, we are able to forecast the speeds for *n* road segments very quickly. Our results are aimed at network-level and real-time prediction. In general, the computation of PCA is computationally demanding when n is large. A more efficient online version of PCA, called PASTd algorithm is used to reduce the data dimension. As a result, our forecasting method is efficient, flexible, and robust.



Key words: Pattern discovery, Traffic flow data, Short-term forecasting, Principal Component Analysis(PCA), *k* Nearest Neighbor(KNN)

# ANALYSIS AND FORECASTING

# FOR TRAFFIC FLOW DATA

by

Yitian WANG

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of science
2017

Advisory Committee:
Professor, Joseph JaJa Chair
Professor,Gang Qu
Professor,Tudor Dumitras

# Acknowledgements

I would like to thank all the people who contributed in some way to the work described in this thesis.

First and foremost, I am greatly indebted to my thesis advisor, Professor Joseph JaJa, for giving me an invaluable opportunity to work on this interesting project and for providing me with definite direction, professional guidance and constant encouragement during my research period. It has been a pleasure to work with and learn from such an extraordinary person. I would like to thank the staff members for providing advice and support during my study period. The prompt and quality management of Melanie is high appreciable. Finally, I must express my profound gratitude to my family and friends who supported me during my time here. I would like to thank my parents for their unfailing support and continuous encouragement throughout these years of study. Thanks go to Miao Zhang, Xin Xu, Yu Jin, and Lin Li, who made my life in Maryland a lot more fun.

Thank you all!.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

## 1.1 Overview

The urban transportation system involves the challenging task of transferring people and materials across densely populated areas. It constitutes the pumping heart of a city, and hence its operational efficiency directly affects the entire city. In recent years, along with the social and economic development and continued urbanization, the population and the quantity of vehicles in each city have substantially increased. Although continuous improvements have been made to the infrastructure of urban transportation systems, there is still a long way to go to satisfy the demands of people for transportation resources in urban areas. Owing to this, traffic congestion and pollution have increased, which negatively has impacted major enterprises and urban residents in their daily production and lives, thus resulting in a great wastes of social resources. The TTS (Intelligent Transport System) is a comprehensive transport management system used in the 21st century. Through monitoring, control and route guidance of operation states of vehicles on the roads, this system can improve the distribution of traffic flows on the urban road network, the utilization efficiency of the urban road network, and enhance the overall transportation efficiency of a city. Transport route guidance and control constitute two important parts of TTS. Therefore, in addition to the construction of a good urban road network for a city, traffic information should be broadcast in real time to ease traffic congestion and improve the quality of life for commuters. This makes it necessary to analyze and forecast traffic flows using real-time traffic information.

Time series analyses are used in many fields with many methods developed to analyze univariate time series consisting of temporal sequenced of scalars[1,2]. Some applications require the analysis of multi-dimensional time series, where the state of each time unit is indicated by a multi-dimensional vector. Specifically, in the analysis of urban traffic networks, the traffic state in a region with *n* connected links can be represented by a *n*-dimensional traffic flow vector F(t). Evaluation of the traffic network state and prediction of the traffic flow require analysis of how the time series of F(t) evolves over time. Yet the high-dimensionality of F(t) makes difficult to generate and predict traffic patterns in a timely fashion.

One solution is to regard each dimension as a single time series and apply traditional univariate time series analysis methods. However, this method does not consider the underlying correlations between the different dimensions, which is of great importance in traffic time series analyses because the flow rates on different adjacent links have strong temporal relationships[3]. Thus, methods are needed to depict and predict the multi-dimensional time series taking into consideration the correlations between the dimensions. One type of analysis uses Kohonen's self-organizing maps (SOMs)[4,5] for the multi-dimensional time series.

The SOMs serve as a clustering algorithm as well as a dimensionality-reduction technique when applied to traffic flows which allow the data to be analyzed to make short-term predictions. Compared to other clustering algorithms such as hierachical clustering or *k*-means clustering, the SOM method has an essential advantage of easy visualization, which offers an effective depiction of the traffic status by projecting the traffic flow vectors onto the SOM planes, while allowing topological preservation

ability. SOMs also outperform other linear dimensionality techniques such as PCA, since it can capture the nonlinear characteristics of the traffic data and thus provide better prediction quality. Chen, et. al. [6] used the SOM method to organize the traffic flow vectors into clusters. Several methods were presented to visualize and interpret these clusters, to identify the typical patterns in the traffic flow distribution, and to reveal basic rules for the evolution of the regional traffic status. The *k*-nearest neighbor (KNN) prediction technique is then applied to the clustering results to perform short-term predictions of flow throughout the whole region. Tests with real world traffic data revealed some interesting phenomena and promising predictions. These results show that the SOM method is a useful tool for time series analysis.

Short-term prediction of traffic parameters such as volume, travel speeds and occupancies have been researched intensively because they offer significant macroscopic traffic characteristics of the transportation system. Because accurate prediction of these parameters is crucial for taking effective measures to relieve traffic congestion, a method that can improve the predictions of existing methods is always of interest for transportation management. To pursue greater prediction accuracy, one way is to improve the quality of the traffic data by creating new data collection and processing techniques. Moreover, it is widely recognized that the performance of the predictions depends not only on the quality and accuracy of the data collected by devices but also on the method adopted for prediction. In this sense, the new forecasting methods and techniques that can improve the predictions deserve more attention.

## 1.2 Traffic data streams

In the late 20th century, data streams are widely used in business areas as a new and more realistic data model. The data streams are characterized by an almost unlimited data size, concept drift, rapid change, the need for rapid response, and large cost of random access to the data. In addition, such data contains valuable enterprise information, such as the operation procedures, management requirements, influencing factors, and variation trends, which can reflect the business operation, service contents, and service targets. At the same time, the wide variability of data streams also brought some challenges to computer storage space, computing speed and communication capacity. The data mining technology has achieved substantial results in mining static data sets, but expanded to the dynamic data streams mining, the problem is still a great challenge.

In the dynamic data streams environment, the rapid growth of data and higher dimensionality lead to deciding against the use of current techniques that mainly focus on static and lower dimensionality.

Currently, data streams have gained considerable importance in various communities (theory, database, data mining network, system), and in applications such as network analysis, sensor network detection, target tracking, financial data analysis, data process and scientific research. All of these applied programs have several points in common: (1) large amounts of data received at a very high speed which makes the speed of database systems very slow in general; (2) the need of real-time processing and mining algorithms that can make prediction in real-time( for example, in network intrusion detection).

Traffic data streams offer an important application which can lead to the development and application of traffic management systems and intelligent transportation systems and traveler information systems. The prediction should be based according to traffic parameter such as flow, density, speed, delay, queueing length and occupancy, etc. Traffic flow information is gained mainly through sensors installed on the roads and GPS-enabled devices resulting in a large amount data of dynamic traffic information collected.

Therefore, this makes it crucial to develop algorithms that pay great attention to multi-dimensional, high order, random, time-varying, non-linear characteristics of traffic flow.

## 1.3 Short- term traffic forecasting

Since the early 1980s, short-term traffic forecasting has been an integral part of most Intelligent Transportation Systems (ITS). It concerns predictions made from few seconds to possibly few hours into the future based on current and past traffic information. Most of the interest has focused on developing methodologies that can be used to model traffic characteristics such as volume, density and speed, or travel times, and produce anticipated traffic conditions. The field of short-term traffic forecasting has been studied during the past 40 years[7]: in the first part of its development, most – if not all – of the research employed 'classical' statistical approaches to predicting traffic at a single point. Later, applications of data driven approaches were the focal point in the literature, where a rich variety of algorithmic specifications – often creatively applied – were proposed. The weight placed recently on empirical computational intelligence-based approaches, including neural and

Bayesian networks, fuzzy and evolutionary techniques, can be considered as inevitable, particularly as most classical approaches have been shown to be ‘weak’ or inadequate under unstable traffic conditions, complex road settings, as well as when faced with extensive datasets with both structured and unstructured data. Recent findings support the shift of research interest towards: (i). more responsive forecasting schemes under non-recurrent conditions; (ii). developing prediction systems with increased algorithmic complexity; (iii). attempting to understand data coming from novel technologies and fuse multi-source traffic data to improve predictions; and (iv). the applicability of AI methodologies to the short-term traffic prediction problem. Although much work has been conducted in short-term traffic forecasting, there are still important research directions that will attract the interest of many researchers in the coming years.

The short-term traffic forecasting interest is the direct result of an increasing need for developing user friendly applications which can provide accurate information to drivers in a timely fashion. The ability to provide such information is the result of phenomenal technological and computational advances that have enabled researchers to collect data and subsequently make prediction at very high temporal resolutions. Both the technological aspects of this analysis (ITS Technology) and the analytical (data analysis), have been the focus of countless research papers over the past few years. The combination of unprecedented data availability and the ability to rapidly process these data has brought on immense development and acceptance of ITS technologies. At the same time, a novel research area, based on data driven empirical algorithms, has been systematically growing in parallel leading to the

mathematical models that are based on macroscopic and microscopic theories of traffic flow [8-12]. This significant leap from analytical to data driven modeling has been marked by an overwhelming increase of Computational Intelligence (CI) – Data Mining (DM) approaches to analyzing the data. Researchers have moved from what can be considered as a classical statistical perspective (the ARIMA Family of models), to Neural and evolutionary computational approaches [13]. Short-term traffic forecasting based on data driven methods is one of the most dynamic and developing research arenas with enormous published literatures.

## 1.4 Processing of traffic flow data

Traffic data information plays a very important role in our daily life. As a result, considerable work has been done for generating and analyzing traffic time series [3,4, 5]. In general, we can regard traffic data as streaming data is generated at regular time intervals. At each time step, we receive traffic information about a large number of road segments, which has to be analyzed and disseminated in real time. On the other hand, vehicle operators would like to receive immediate up-to-date traffic summaries and cannot afford any post-processing[14].

An effective way to manage our problem is to reduce the large volumes of traffic data into a small meaningful trends that can be updated and broadcast in real time. For the traffic flow data, the information for most road segments is correlated. Hence one possible research direction is to use clustering algorithms to group together road segments that follow similar traffic patterns. For example, within a group, if the speed of one segment decreases at a certain time step, then the speed of almost all other segments of the group also decrease. Instead of analyzing the traffic of the *n* road

segments, we can analyze the patterns of the *k* groups, where k ≪ n. Applying self-organizing maps (SOM) is a possible approach, as has been done in [15] for the case of post processing analysis. Mixture Models present good possibilities for streaming data analysis [16, 17].

Short-term traffic prediction plays crucial role in intelligent transportation system (ITS). It refers to forecast traffic flow data for the next few minutes up to around 60 minutes depending on the network. With reliable forecasting data, administrators can manage traffic networks effectively and travelers can decide on departure time or travel routes easily [18].

Many statistical models have been proposed for short-term traffic forecasting. For example, time series models [19, 20], Bayesian models [21], Kalman filter models [22], and support vector machine regression models [23] have been widely applied to predict motorways and freeways traffic conditions. Neural network models using artificial intelligence algorithms [24, 25, 26] and unsupervised machine learning algorithms [27] also gain researchers' attention recently.

## 1.5 Study contents of the thesis

### 1.5.1 Pattern discovery for traffic flow data

In this thesis, we pursue a different approach based on finding *k* patterns (or hidden variables) such that the time series of the average speeds of each road segment can be estimated using a linear combination of these patterns. What we need is to find such *k* patterns and the corresponding weights for each pattern. In our case, we are given a large number *n* of such time series. We would like to find a small number *k*

hidden variables that can represent all the n time series as accurately as possible. We would like to find a method which can find patterns in online fashion with linear complexity, and don't need to buffer data.

We describe two techniques to determine *k* critical patterns (or hidden variables) for a large number *n* road segments. These patterns can be used to reconstruct the traffic data for all the segments. One is based on the conventional PCA method and the other is based on the online PASTd algorithm.

We also compared the performance between PCA and PASTd. The comparison shows that PASTd is more suitable for finding patterns in terms of accuracy and time efficiency. The reason is mainly because it demands low computation - $O(kn)$ floating point operations. We don't need to use the expensive SVD decomposition. Its space requirement is also low since it doesn't need to buffer any data except the mean.

We evaluate the quality of the solutions in terms of the reconstruction error of the traffic information from the *k* patterns and in terms of time efficiency as the patterns are updated in real-time. The two methods described provide very good results in terms of reconstruction error and computational efficiency.

### 1.5.2 Short-term forecasting for traffic flow data

Until now, most models focus on motorways and freeways [11, 13]. A network-level method is needed for better prediction. This challenge requires the prediction method to be efficient and scalable. Even though the number of road segments can get very large, the method is still able to make reliable prediction in real time.

We successfully reduce the dimensionality of the correlated road segments from *n* to a much smaller number *k* by applying techniques related to Principal Component

Analysis (PCA), where *k* is the number of hidden variables. We try to predict future values of the hidden variables by using the *k* nearest neighbor(KNN) method. Then we are able to forecast the vehicle speeds for all the *n* road segments. Instead of only freeways, we aim at the network-level and real-time prediction. Principal Component Analysis (PCA) is a possible way to address this challenge.More specifically, PCA is able to reduce the data dimension from a large number *n* to a much smaller number *k*. Instead of dealing with *n*-dimensional traffic data, we can focus on a few *k* hidden variables. However, the computation of PCA is quite high when *n* is large. To solve this problem, we use a more efficient online version of PCA, called the PASTd algorithm [14] to reduce data dimension. The advantage of PASTd algorithm is that it is an online algorithm, in that it can incorporate new data at every time step, making PASTd capable of real time forecasting. Also, the relevant spatio-temporal patterns of the network can be inferred from a long series of historical data used as the sample data for KNN. Thus, our algorithm does not depend on the network structure. As a result, the forecasting method is efficient, flexible, and robust.

## 1.6 Outline of the thesis

The rest of the thesis is organized as follows: Chapter 2 provides an overview of the PCA approach as applied to predicting traffic patterns of a complete transportation network. Chapter 3 introduces our method for short-term forecasting of traffic flows, comparing our approach with the KNN approach. Chapter 4 concludes with possible future research directions.

# Chapter 2 Pattern Discovery for Traffic Flow Data.

## 2.1 Introduction

Traffic data information plays a very important role in our daily life. As a result, considerable work has been done for generating and analyzing traffic time series [7, 28, 29]. In general, we regard traffic data as streaming data that is generated at regular time intervals. At each time step, we receive traffic information about a large number of road segments, which has to be analyzed and disseminated in real time. On the other hand, vehicle operators would like to receive immediate up-to-date traffic summaries and cannot afford any post-processing[30].

An effective way to manage our problem is to reduce the large volumes of traffic data into a small meaningful trends that can be updated and broadcast in real time. For the traffic flow data, the information for most road segments is correlated. Hence one possible research direction is to use clustering algorithms to group together road segments that follow similar traffic patterns. For example, within a group, if the average speed of vehicles on one road segment decreases at a certain time step, then the speed of almost all other segments in the group will also decrease. Instead of analyzing the traffic of the $n$ road segments, we can analyze the patterns of the $k$ groups, where $k$ is much smaller than $n$. Applying self-organizing maps (SOM) is a possible approach, which has been reported in [15] for the case of post processing analysis. Mixture Models present good opportunities for streaming data analysis [16, 17]. Here we pursue a different approach based on finding $k$ patterns (or hidden variables) such that the time series of the average speeds of each road segment can be

generated using a linear combination of these patterns. What we need is to find such *k* patterns and the corresponding weights for each pattern.

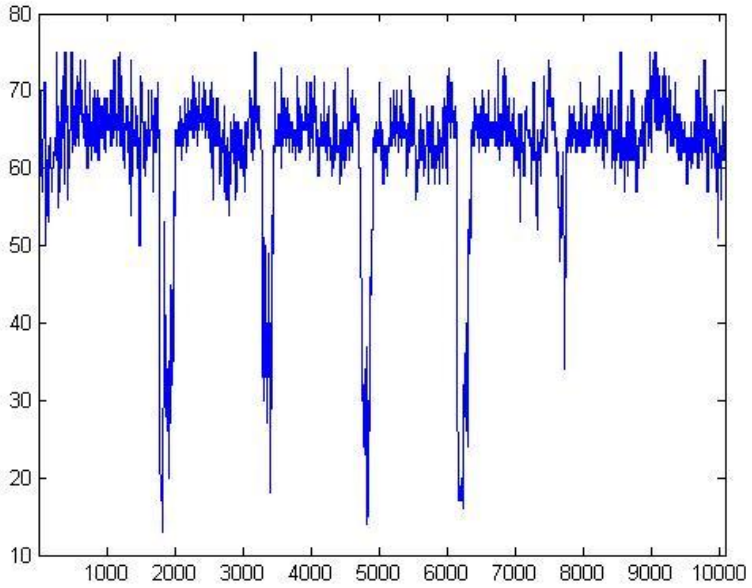


Figure 2.1: time series for one segment

Figure 2.1 is an example of a time series of average speeds for a road segment in Maryland over a week. In our case, we are given a large number *n* of such time series. We would like to find a small number *k* hidden variables that can represent all the *n* time series as accurately as possible. The authors of [31, 32, 33] explicitly focus on discovering hidden variables. In CluStream [31], patterns are found by an offline strategy on stored data. Braid [32] determines lag correlations among multiple streams. StatStream [33] uses DFT to summarize streams within a finite window size. We would like to find a method that can find a relatively few inherent patterns in an online fashion with linear complexity, with no need for data buffering.

## 2.2 Principal component analysis

The Principal Component Analysis (PCA) method is a popular tool in data analysis which projects the high–dimensional data onto a low-dimensional subspace while preserving most of the variance in the data. The method is simple and non-parametric [30]. In essence, PCA can be applied to reduce the dimension of a complex data set while revealing the hidden, simplified patterns underlying the data. In this section, we will give a brief overview of PCA. For the notation, we use lower-case bold to represent column vectors, upper-case bold for matrices, and plain font for scalars. In the following, $\mathrm{x}_t = [\mathrm{x}_{1,t}\ \mathrm{x}_{2,t} \ldots \mathrm{x}_{n,t}]^{\mathrm{T}} \in \mathbb{R}^n$ is an $n$-dimensional column vector of average speed values of the different road segments at time step $t$. $\mathrm{X}_t = [\mathrm{x}_1 \mathrm{x}_2 \ldots \mathrm{x}_t] \in \mathbb{R}^{n \times t}$ can be viewed as an $n \times t$ matrix, where a new column is added at each time step $t$.

There are several ways to explain the PCA technique. One possibility is to model the vector $\mathrm{x}_t$ as a linear combination of $k$ hidden variables. That is, we express $\mathrm{x}_t = \mathrm{W}\mathrm{z}_t$, where $\mathrm{z}_t$ are $k$ hidden variables whose values depend on the time step $t$, and $k \ll n$. The matrix W is an $n \times k$ orthonormal matrix to be determined. Since W is orthonormal, $\mathrm{W}\mathrm{W}^{\mathrm{T}} = \mathrm{I}_{k \times k}$. Hence we deduce that $\mathrm{z}_t = \mathrm{W}^{\mathrm{T}} \mathrm{x}_t$. Using this model, we can reconstruct each $\mathrm{x}_t$ using $\tilde{x}_t = \mathrm{W}\mathrm{W}^{\mathrm{T}} \mathrm{x}_t$. Assume we want to focus on a time window of size T, and that we would like to reconstruct all the data within this window, say $\mathrm{X}_{\mathrm{T}} = [\mathrm{x}_1 \ldots \mathrm{x}_T]$. Then our optimization (minimizing reconstruction error) can be formulated as follows.

$$\min_{\text{w ortho.}} \sum_{\tau=1}^{T} \left\| x_\tau - \left(WW^T\right) x_\tau \right\|^2 \qquad (1)$$

Using the Singular Value Decomposition (SVD) technique, the solution can be expressed as $W = [w_1 \ldots w_k]$ where each column $w_i$ is the eigenvector corresponding to *i*-th largest eigenvalues of $X_T X_T^T$ . Then the hidden variables are given by $z_t = [z_{1,t} \ldots z_{k,t}]^T = \left[ w_1^T x_t \ldots w_K^T x_t \right]^T$ . Thus, for any given $k < n$, we can find an orthonormal matrix W and the *k* hidden variables $z_t$ to reconstruct the data.

Another interpretation of PCA is that we want to determine an orthonormal basis $w_1, \ldots, w_k$ such that projecting the data onto this subspace will capture the maximum statistical variances. We can start by determining $w_1$ such that that variance of the projections on this axis is as large as possible:

$$\begin{aligned} \text{Variance} &= \frac{1}{n}\sum_{t=1}^{T}\left( w_1^T x_t - w_K^T \bar{x} \right)^2 \\ &= \frac{1}{n} w_1^T \left( \sum_{t=1}^{T}\left( x_t - \bar{x} \right)\left( x_t - \bar{x} \right)^T \right) w \qquad (2) \\ &= w_1^T S w_1 \end{aligned}$$

Where $S = \frac{1}{n}\sum_{t=1}^{T}\left( x_t - \bar{x} \right)\left( x_t - \bar{x} \right)^T$ is the sample convariance matrix.

Then our problem becomes:

$$\begin{aligned} &\max w_1^T S w_1 \\ &\text{s.t. } 1 - w_1^T w_1 = 0 \qquad (3) \end{aligned}$$

Using a Lagrange multiplier, we find that $w_1$ is the eigenvector corresponding to the maximum eigenvalue of S. Similarly, we can find $w_2, \ldots, w_k$, where $w_i$ is the eigenvector corresponding to the *i*-th largest eigenvalue of S. The solution is consistent to the first interpretation. The projection of $x_t$ onto these orthonormal basis is $\tilde{x}_t = \left( w_1^T x_t \right) w_1 + \ldots + \left( w_k^T x_t \right) w_k$ .

If we represent each *n*-dimensional column vector $x_t$ with all the *n* principal components, then the error $\|x_t - \tilde{x}_t\| = 0$. However we can usually use only the first *k* principal components, where $k \ll n$, to reconstruct all the data with a very small error.

## 2.3 Discovering hidden variables

In this section, we show how to use PCA to find the most important patterns underlying our complex traffic data set. We first introduce a conventional PCA method to find the hidden variables so as to reconstruct the data set using these hidden variables as accurately as possible. Then we describe the online method which can update hidden variables at every time step with linear complexity.

**Problem Formulation** Given *n* time series corresponding to average speeds on *n* road segments, updated at each time step *t*, we aim at determining *k* hidden variables $z_t$, where $k \ll n$, such that linear combinations of these *k* hidden variables can be used to reconstruct the data matrix within any time window of size T. Thus, the dimension of the data set is significantly reduced. As a result, we could make more effective, low-cost prediction for speeds in the near future.

### 2.3.1 Offline PCA pattern discovery

As discussed in Section 2.2, we can identify the hidden variables by computing the eigenvectors of the sample covariance matrix of our input data. Then we can use the first *k* eigen-vectors to reconstruct the data matrix. More details are described in the following algorithm that generates the *k* hidden variables corresponding to the traffic data of *n* road segments over a time window of size T.

Algorithm 1 PCA Pattern Discovery

**Given**: window size T, number of hidden variables *k.* After receiving every set of T streaming data vectors, we do:

1. Organize the data into an $n \times t$ matrix, i.e. $X_T \in \mathbb{R}^{n \times k}$.

2. Normalize $X_T$. $x_i = \frac{x_i - \bar{x}_i}{std(x_i)}, i = 1, ..., n$, where $x_i$ is the *i*-th row of $X_T$.

3. Calculate the *k* eigenvectors corresponding to the *k* largests eigenvalues of $X_T$ $X_T^T$ , i.e. $w_1, \ldots, w_k$.

4. Compute the *k* hidden variables. $z_t = [z_{1,t} \ldots, z_{k,t}] = \left[ w_1^T x_t, \ldots, w_k^T x_t \right], t = 1, ..., T,$ where $x_t$ is the *t*-th column of $X_T$. These represent the *k* patterns.

5. The data matrix can be reconstructed as follows:

$\tilde{x}_t = z_{1,t} w_1 + ... + z_{k,t} w_k, t = 1, ..., T.$

The matrix $W = [w_1 \ldots w_k] \in \mathbb{R}^{n \times k}$ is called the weight matrix. For each element, $w_{i,j}$, $1 \le i \le n, 1 \le j \le k$ , the magnitude $|w_{i,j}|$ provides some indication on how much the *i*-th segment is dependent on the *j*-th hidden variable [30].

We will later analyze the performance of this algorithm for different window sizes. As we will see, the reconstruction error is quite small even for just two hidden variables as long as the size of the time window is reasonably small.

### 2.3.2 Online PASTd pattern discovery

The previous PCA algorithm requires the buffering of the data for every time window and requires a significant amount of computation, namely computing the first

few eigen-vectors of the sample covariance matrix, which can be fairly large. We now describe an online method that updates the hidden variables and the weight matrix incrementally in linear time (as a function of the traffic information for all the road segments received at each time step). We use PASTd algorithm, which is based on adaptive filtering techniques and PCA. The PASTd algorithm has been shown to perform very well in a variety of settings and on different applications, such as signal tracking for antenna arrays and image compression [34].

---

Algorithm 2 PASTd Pattern Discovery

---

0. **Initialize**: k orthonormal weight vectors $w_1(0) = [10 \ldots 0]^T$ , $w_2(0) = [010 \ldots 0]^T$ , etc. $d_i(0),\ i = 1, \ldots, k$ to a small positive value. Then:

1. As each point $x_t$ arrives, set $\mathbf{x}_1 = \mathbf{x}_t$ .

2. For $1 \le i \le k$ we perform the following assignments and updates:

$$
\begin{aligned}
z_i(t) &= \mathbf{w}_i^T(t-1)\mathbf{x}_i \\
d_i(t) &= \gamma d_i(t-1) + z_i(t)^2 \\
e_i(t) &= \mathbf{x}_i - z_i(t)\mathbf{w}_i(t-1) \\
w_i(t) &= w_i(t-1) + \frac{1}{d_i(t)} z_i(t) e_i(t) \\
\mathbf{x}_{i+1} &= \mathbf{x}_i - z_i(t)\mathbf{w}_i(t)
\end{aligned}
$$

---

Algorithm 2 enables the explicit computation of eigencomponents [34]. In fact, $w_i(t)$ is an estimate of the *i*-th eigenvector at time step *t*, and $d_i(t)$ is an estimate of the corresponding eigenvalue of the matrix $S_t$, where $\mathbf{S}_t = \gamma \mathbf{S}_{t1} + \mathbf{x}_t \mathbf{x}_t^T$ . These eigenvalues

may be used to estimate the number k of hidden variables if it is not given. The use of forgetting factor $0 < \gamma \leq 1$ is intended to ensure that data matrix $S_t$ is more dependent on the most recent data. Since the traffic data is non-stationary, $\gamma$ can guarantee the tracking ability and will give more precise estimates of the eigen-components. The vector $e_i(t)$ is the error between the true data and the reconstruction, and $\mathbf{e}_i(t) \perp \mathbf{w}_i(t)$. The step for updating the eigenvector $w_i(t)$ can be interpreted as a gradient descent method with a self-tuning step size $\frac{1}{d_i(t)}$.

**Complexity** For each *n*-dimensional data vector, we only need *k* iterations for updating weight vectors $w_i$, $1 \leq i \leq k$. Thus, the total cost is $O(nk)$ both in terms of time and of space. The update cost does not depend on time window size T. This significantly outperforms the conventional PCA, where the computation cost for eigenvalue decomposition alone is $O(n^3)$.

## 2.4 Reconstruction results

In this section, we show the reconstruction results by using both the classical PCA method and the online PASTd method. We then compare these two methods in terms of accuracy as well as time efficiency. In our tests, we chose *n* = 48 road segments over a whole week, which amount to 7∗24 ∗60 = 10080 vectors each of dimension 48. These 48 segments were randomly chosen from all the road segments of the State of Maryland. The order of the days are Sunday, Monday, . . . , to Saturday. If any data is missing, we use the data of the previous time step to fill-in the gap.

### 2.4.1 PCA Performance

In our test, the time window size is selected to be T = 30, meaning that we buffer the data and compute the covariance matrix corresponding to every 30 minutes. We use $k$ = 2 hidden variables to reconstruct the data matrix within each time window.

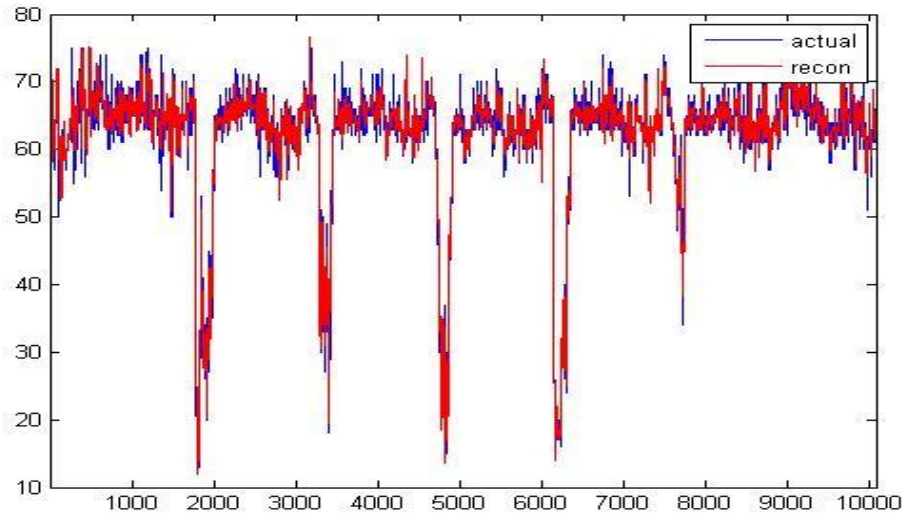


Figure 2. 2: PCA reconstruction using $k$ = 2 patterns for one road segment over a week

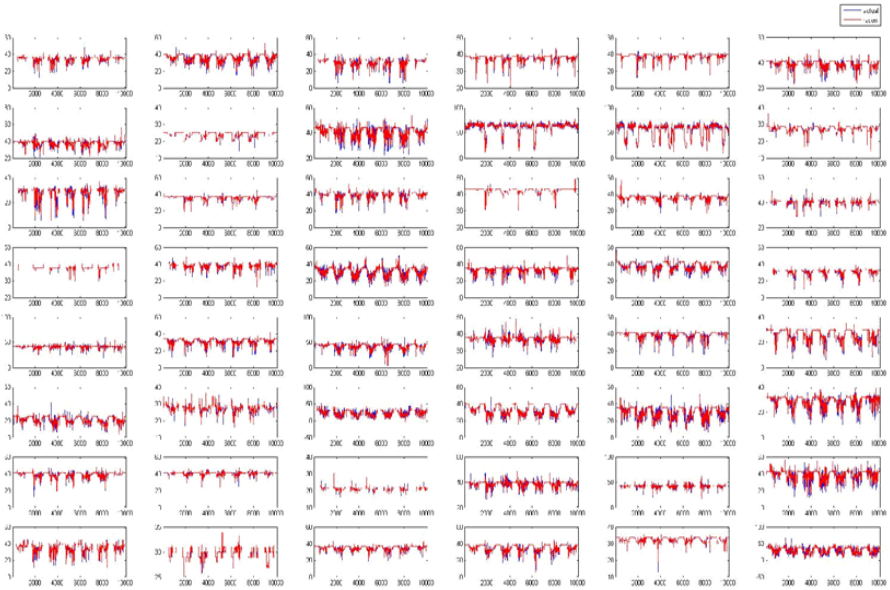

Figure 2.3: PCA reconstruction using $k$ = 2 patterns for all 48 segments over a week

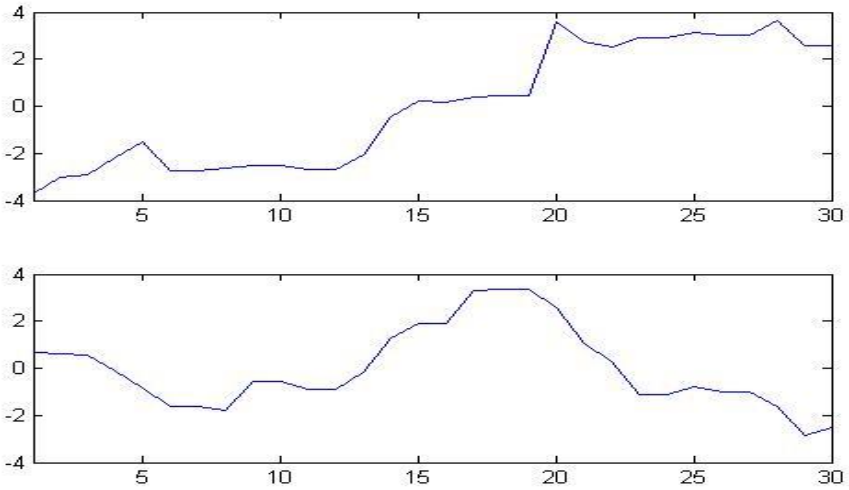


Figure 2. 4: hidden variables for one window

**Reconstruction Results** In both Figures 2.2 and 2.3, the blue line is the original data while the red line corresponds to the reconstructed data. Figure 2.2 shows the result for one road segment while Figure 2.3 shows the results for all the 48 segments. As we can see, our reconstruction captures most statistical variance and has small deviation from the true data.

**Hidden Variables** Figure 2.4 shows the first two hidden variables for one window with size T = 30. The top plot corresponds to the first hidden variable while the bottom corresponds to the second hidden variable. Note that these are the hidden variables for the normalized data. Thus, the value of y-axis illustrates the deviation from the mean. Although this figure represents the patterns for only one time window, we find that the hidden variables for other windows follow similar patterns: the first hidden variable is monotonically increasing or decreasing, and the second is first increasing then decreasing or vice versa. This is to be somewhat expected since the window size is relatively small, the speed for each road segment within one window

does not change a lot, leading the frequency of critical patterns to be low. Due to such characteristics, we are able to make predictions for hidden variables. Then we can save a lot time and energy for prediction since we only need to deal with *k* hidden variables instead of all *n* road segments.

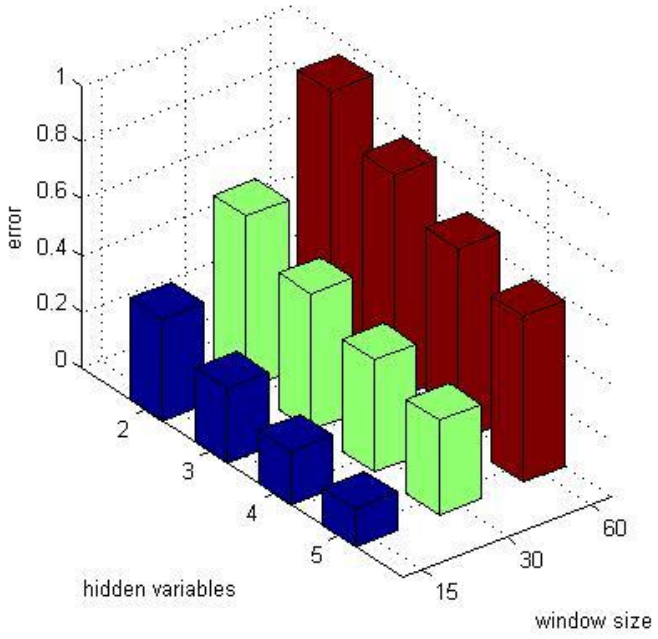


Figure 2.5: Reconstruction error using PCA

**Impact of Parameters** In this subsection, we discuss how changing the values of the parameters affect the performance. In our test, we vary the time window size T from 15 minutes to 60 minutes and vary the number of hidden variables from 2 to 5. We evaluate the two methods in terms of the reconstruction error and computational time. Here, the definition of the reconstruction error is given by:

$$\text{Error} = \frac{1}{n \times T} \sum_{t=1}^{T} \left| x_t - x_t \right| \qquad (4)$$

where $\left| x_t - x_t \right|$ is the $l_1$ norm between the two vectors.

In Figure 2.5 the x-axis represents the window size, ranging from 15 minutes to 60 minutes, the y-axis represents the number of hidden variables, ranging from 2 to 5, and the z-axis represents the corresponding reconstruction error. As we can see, the smaller the window size and the larger the number of hidden variables, the better performance. This is expected since the patterns appearing over a small window size

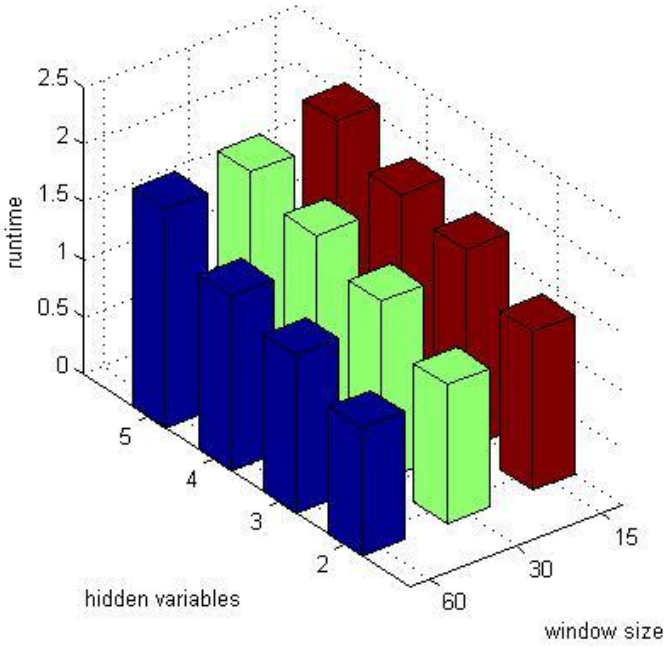


Figure 2.6: Runtime using PCA

are simpler than those over a large window size. As a result, it is easier to reconstruct the data matrix for the smaller window. Also, the more hidden variables we use, the more variance we capture, and hence the smaller reconstruction error.

In Figure 2.6, the x-axis and the y-axis have the same meaning as Figure 2.5, but the z-axis represents now the corresponding runtime in seconds. Note that the orderings along the x-axis and y-axis are different from Figure2.5. As we can see, the larger window size and the smaller number of hidden variables, the faster the

algorithm, which is to be expected. Larger window size means less update, and hence less time. Also, increasing the number of hidden variables consumes more time for any fixed window size. Thus there is a tradeoff between reconstruction error and runtime. However we ran our algorithm on the data for a whole week, and the worst case scenario only took around 2 seconds. Note that we are using a 2.6GHz processor with 32 GB of RAM.

### 2.4.2 PASTd Performance

For PASTd, we use $k = 2$ hidden variables and compute the reconstruction error. We update these two hidden variables and weight matrix at every time step. The reconstruction results are shown in Figure 2.7 and 2.9.

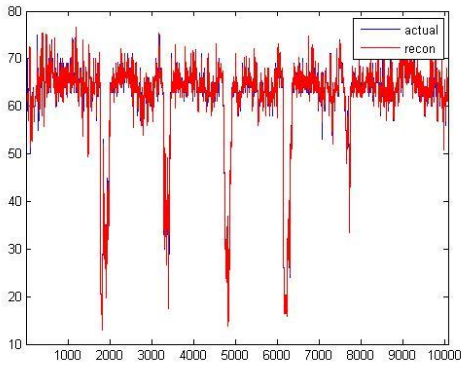


Figure 2.7: PASTd reconstruction using $k = 2$ patterns for one road segment over a week

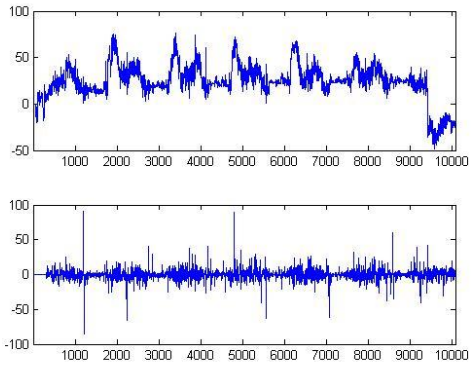


Figure 2.8: hidden variables over a week

**Reconstruction** For the results shown in Figures 2.7 and 2.9, the blue line represents the original data and the red line is the reconstructed data. Figure 2.7 shows the reconstruction result for a single road segment, while Figure 2.9 shows the results for all the 48 segments. We can barely see blue lines in our figures, meaning that our

reconstruction is basically the same as the original data. Hence, we succeed in reducing the data dimension from 48 to 2 by using online PASTd method.

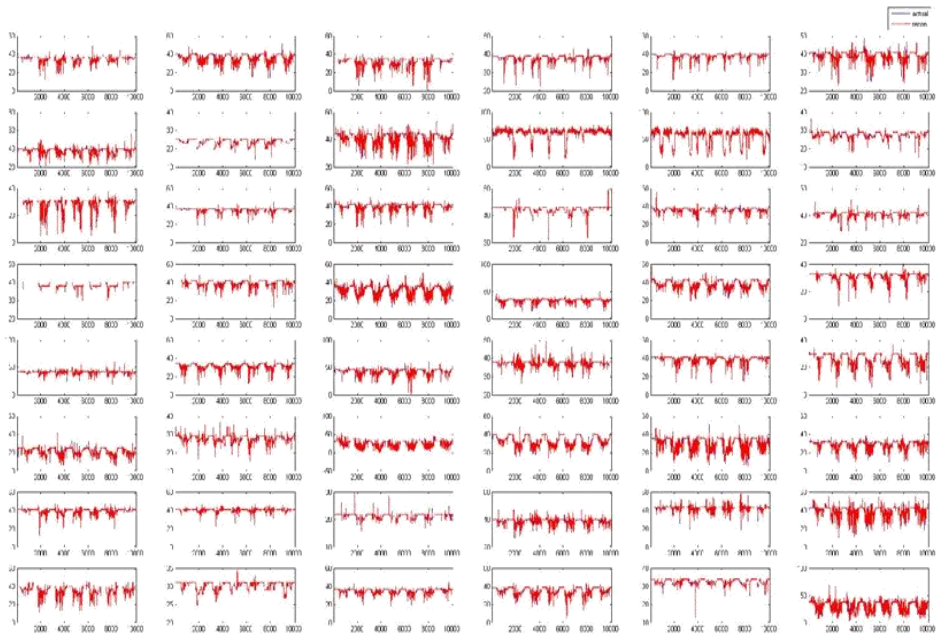

Figure 2.9: PASTd reconstruction using $k = 2$ patterns for all 48 segments over a week

**Hidden Variables** Figure 2.8 illustrates the time series of the two hidden variables determined over a week. The top one corresponds to the first hidden variable and the bottom one corresponds to the second hidden variable. As we can see from the time series of the 1st hidden variable, the five weekdays show similar patterns while weekend days show a different pattern. This can be justified by the fact that weekdays have obvious rush hours while weekends do not necessarily follow that pattern. Also the first hidden variable captures the largest variance and hence there will be significant differences between weekdays and weekends. Note that the second hidden variable, on the other hand, captures the second largest variance. As we can see, it has large variance during daytime and small variance around midnight. This is reasonable

since at midnight, we don't expect much traffic and hence the speed is mainly decided by the speed limit, that is, the speed doesn't change much. During daytime, there is much more traffic and the speed varies significantly, leading to more variance for the second pattern.

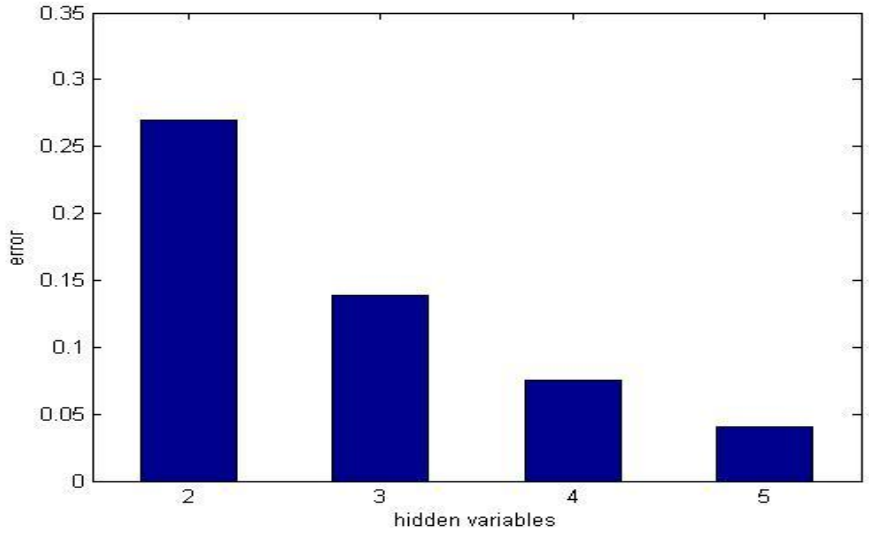


Figure 2.10: Reconstruction error using PASTd

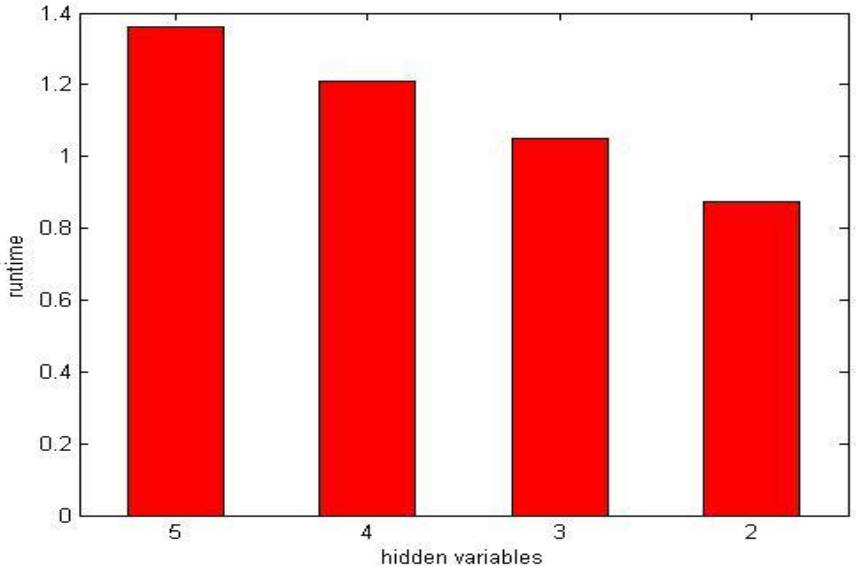


Figure 2.11: Runtime using PASTd

**Impact of Parameters** In PASTd, we update the weight matrix as well as hidden variables incrementally at each time step. The only parameter that matters is the number $k$ of hidden variables. As in the previous subsection, we test the performance of PASTd in terms of reconstruction error and time efficiency as a function of $k$.

Figure 2.10 shows the reconstruction error when using the PASTd method. The definition of the reconstruction error is similar to that used for the PCA. As we can see, by using a larger number of hidden variables, we can get better performance. The reason is that the more hidden variables we find, the more variance of the data we capture, and hence the smaller the reconstruction error. Note that, even for $k = 2$, the error is less than 0.3 mph. This shows that online PASTd method is quite effective in determining the traffic patterns.

Figure 2.11 shows the runtime corresponding to different values of $k$ using PASTd in seconds. As we can see, a smaller number of hidden variables leads to shorter runtime, which should be obvious since the complexity for updating is $O(kn)$. Note that even for $k = 5$, the runtime is less than 1.5 seconds over a week. This means that we are able to update the weight matrix and hidden variables incrementally in a very short time. Thus, PASTd is completely suitable for realtime systems.

**Comparison with PCA** We compare the performance between of PCA and PASTd in terms of accuracy and time efficiency. For accuracy, we choose a window size T = 15 since small window leads to better results. For time efficiency, we pick window size T = 60 since a large window size needs less updating and thus takes less time.

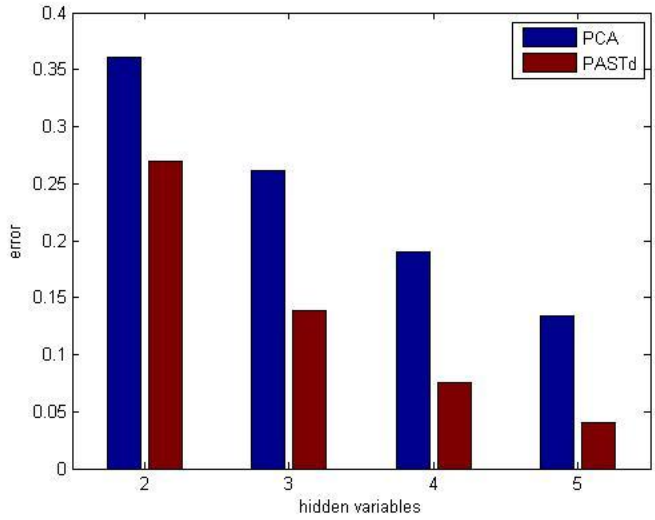


Figure 2.12: Comparison for error with T = 15 in PCA

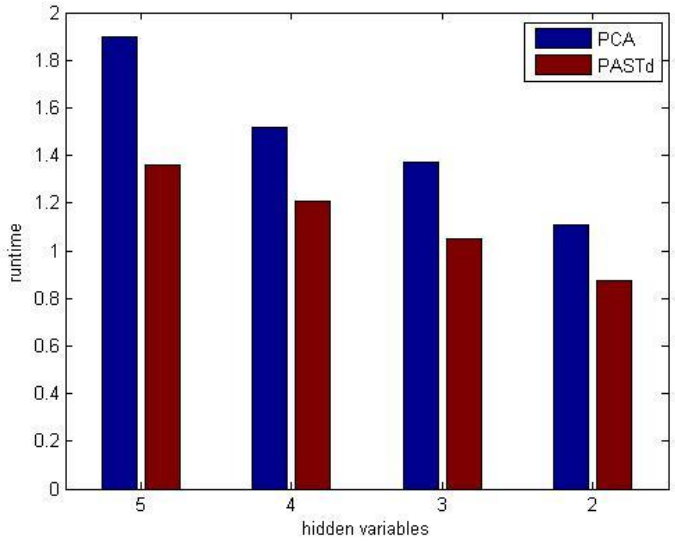


Figure 2.13: Comparison for runtime with T = 60 in PCA

In both Figures 2.12 and 2.13, blue blocks represent PCA while red blocks represent PASTd. Figure 2.12 shows the error comparison between PCA and PASTd as a function of the number of hidden variables. Online PASTd outperforms classical PCA probably because PCA captures only the patterns in a single time window while PASTd incrementally updates the patterns taking into consideration the overall past

with much more weight assigned to the recent past. Another reason might be that PCA is sensitive to window size T. As we shrink to T = 10, PCA will outperform PASTd in that smaller window which has simpler patterns and thus easier to reconstruct.

Figure 2.13 shows the runtime comparison between PCA and PASTd. PASTd beats PCA again because PASTd enjoys a linear complexity $O(n)$ while PCA has at least $O(n^3)$. When $n$ becomes large, the difference will be substantially larger. In conclusion, although online PASTd only gives estimates for eigencomponents, it is able to produce very good results and takes less time compared to PCA. Online PASTd is definitely more suitable for discovering patterns for real-time traffic systems.

## 2.5 Conclusion

In this section, we describe two techniques to determine $k$ critical patterns (or hidden variables) for a large number $n$ road segments. These patterns can be used to reconstruct the traffic data for all the segments. One is based on the conventional PCA method and the other is based on the online PASTd algorithm. The reconstruction results for both methods show that:

- We can reduce the dimensionality of the correlated road segments from $n$ to a much smaller number $k$.
- We can adaptively change values of parameters to improve performance.
- We can update the hidden variables and weight matrix efficiently. For PCA, run-time is around 2 seconds while for PASTd the runtime is around 1 second for a whole week data samples of 48 road segments.

We also compared the performance between PCA and PASTd. The comparison shows that PASTd is more suitable for finding patterns in terms of accuracy and time efficiency. The reason is mainly because it demands low computation - $O(kn)$ floating point operations. We don't need to use the expensive SVD decomposition. Its space demand is also low since it does not need to buffer any data except the mean. We can use PASTd of travel time prediction as follows. We can choose any forecasting model such as AR model to predict hidden variables, and use the previous weight vectors to predict the speed for the next time step [30], i.e.,

$$\hat{\mathbf{z}}(t+1) = f(\mathbf{z}(t))$$

$$\mathbf{x}(t+1) = \hat{z}_1(t+1)\mathbf{w}_1(t) + \ldots + \hat{z}_k(t+1)\mathbf{w}_k(t)$$

By using PASTd, we can save significantly in computational time as well as energy for prediction. When $n$ is large, the advantage of PASTd will be much more significant.

# Chapter 3 Short-term Forecasting for Traffic Flow Data

## 3.1 Introduction

Short-term traffic prediction plays crucial role in intelligent transportation system (ITS). It refers to forecasting traffic flow data for the next few minutes up to around 60 minutes depending on the network. With reliable forecasting data, administrators can manage traffic networks effectively and travelers can decide on departure time or travel routes more easily [18].

Many statistical models have been proposed for short-term traffic forecasting. For example, time series models [19, 20], Bayesian models [21], Kalman filter models [22], and support vector machine regression models [23] have been widely applied to predict motorways and freeways traffic conditions. Neural network models using artificial intelligence algorithms [24, 25, 26] and unsupervised machine learning algorithms [27] also have gained researchers' attention recently.

Although artificial intelligence algorithms can overcome several problems and provide black box solutions, the implementation can be complex and hard to interpret, and these models are difficult to extend from one application to another [35]. This thesis uses the *k* nearest neighbor model (KNN) as the forecasting method. KNN is a typical example of nonparametric regression method. Compared to the parametric models, some of which are simple to implement but do not achieves accurate prediction [36], the nonparametric regression method is more portable, achieves higher accuracy, and has a simpler structure.

Until now, most models focus on motor-ways and freeways [35, 37]. A network-level method is needed for better prediction. We require that the prediction method is

efficient and scalable. Even though the number of road segments can get very large, the method should be able to make reliable prediction in real time.

The remainder of this chapter is organized as follows: Section 3.2 gives a brief description of KNN and how we apply it to our model. Section 3.3 explains the specific steps of our algorithm, from the dimensionality reduction step to the prediction of the hidden variables. Forecasting results with horizons varying from 1 to 60 minutes are shown in Section 3.4. The impact of parameters and the computational complexity of the algorithm are discussed in Section 3.5. The conclusion is addressed in Section 3.6.

## 3.2 KNN method

In this section, we start by giving a brief introduction about the *k* nearest neighbor (KNN) method. Then we show how to apply KNN to forecast the hidden variables over the next brief time horizon. To make our description clearer, we follow the notations described in Table 3.1.

| Symbol | Description |
|---|---|
| $x, \ldots$ | Column vectors (lowercase boldface) |
| A, . . . | Matrices (uppercase boldface) |
| $x_t$ | The *n* speed values at time *t* |
| *n* | Number of road segments |
| $z_t$ | The *k* hidden variables at time *t* |
| *k* | Number of hidden variables |
| *knn* | Number of nearest neighbors in KNN |

| | |
|---|---|
| $h_f$ | Forecasting horizon |
| $h_p$ | Past horizon |
| $l$ | Number of historical data |

Table 3.1: Description of notation

The KNN method collects historical data as the sample database. In our case, a *k*-dimensional vector $z_t = [z_{1,t} \ldots z_{k,t}]$ is stored, where $z_{i,t}$ is the *i*-th hidden variable for time step *t*. Then, the Euclidean distances between all sample points and current data are calculated to generate the *knn*-sample data, that is, the *k* nearest neighbors to the current data. Finally, future hidden variables are forecasted by using a weighted average of the KNNs.

In our work, we found that for each day of a week, the hidden variables follow similar patterns. For example, in Figure 3.1 and 3.2, the first hidden variable for five consecutive Wednesdays look very similar to each other. This is reasonable since hidden variables intuitively reflect the essential patterns of vehicle speeds. As a result, we can forecast the hidden variables for the $(l + 1)$-th Wednesday by using the hidden variables in the past l Wednesdays.

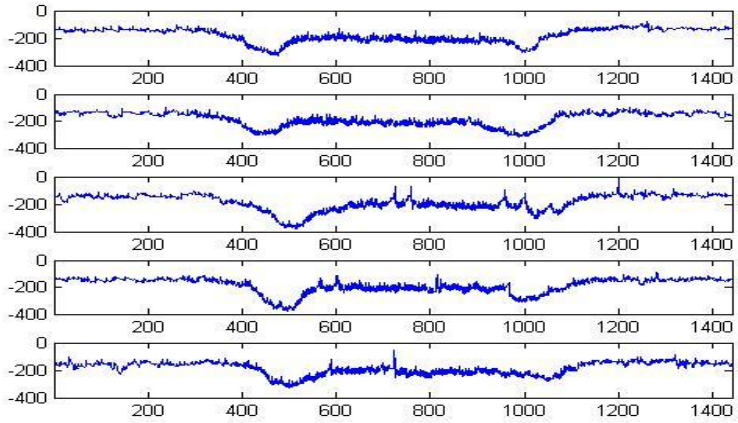

Figure 3.1: first hidden variable for Wednesdays for one county

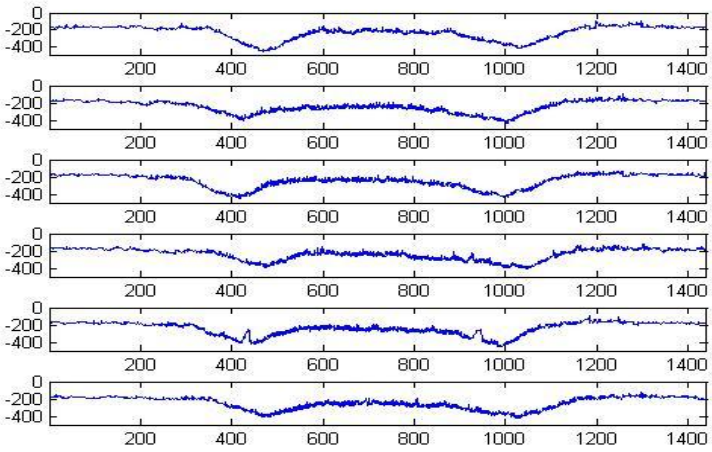


Figure 3.2: first hidden variable for Wednesdays for multiple counties

In our case, we receive $n$ speed values every minute, i.e., $x_t = [x_{1,t}, \ldots x_{n,t}]^T$. Then we compute the corresponding $k$ hidden variables $z_t = [z_{1,t}, \ldots z_{k,t}]^T$. Since there are 1440 minutes in a day, we get a $k \times 1440$ matrix for each day. The matrix is stored as historical data for KNN. After collecting historical data for l weeks, we get 1 such matrices for each day of the week. To forecast $Z_{t+h_f}$ for the $(l+1)$-th Wednesday, where $h_f$ is the forecasting horizon, we first retrieve the past $h_f$ hidden variables $P_{t,hp} = \left[ z_{t-h_p}, \ldots z_{t-1} \right] \in \mathbb{R}^{k \times h_p}$. The $i$-th row of $P_{t,hp}$ means the past $h_p$ values for the $i$-th hidden variable at time step $t$. We apply the KNN method to each hidden variable respectively. Assume that the distances between current data to the nearest neighbors are $d_1, \ldots, d_{knn}$, and the corresponding hidden variables are $z_{i,t+h_f,l_1}, \ldots z_{i,t+h_f,l_{knn}}$. Then $z_{i,t+h_f}$ for the $(l+1)$-th week can be forecasted as:

$$z_{i,t+h_f} = \frac{\sum_{j=1}^{knn} \frac{1}{d_j} z_{i,t+h_f,l_1}}{\sum_{j=1}^{knn} \frac{1}{d_j}} \qquad (1)$$

## 3.3 Forecasting algorithm

In this section, we show the specific steps needed to apply the KNN method, which are presented in Algorithm 1. To deal with missing data of vehicle speeds, we use the values at the previous time step. Thus, we have 1440 time steps for each day.

Algorithm 1. KNN Method

0. **Initialize**: w, d equal to the values from the last sample data.

1. At time step *t* during ($l$ + 1)-th week, get

the l historical data $z_{t-h_f}(j),...,z_{t-1}(j),1 \le j \le l$. Then for $1 \le i \le k$ :

2. Compute the Euclidean distances between

$z_{i,t-hp}$ (l + 1), . . . , $z_{i,t-1}$(l + 1) and the l historical data.

3. Find the *knn* nearest neighbors with the *i* first *knn*

shortest distances $d_1, \ldots, d_{knn}$, the corresponding weeks are $l_1, \ldots, l_{knn}$.

4. Forecast the hidden variables with the horizon $h_f$ using $z_{i,t+h_f} = \frac{\sum_{j=1}^{knn} \frac{1}{d_j} z_{i,t+h_f,lj}}{\sum_{j=1}^{knn} \frac{1}{d_j}}$

5. Forecast the vehicle speeds using $\tilde{x}_{t+h_f} = w^T z_{t+h_f}$

6. Update w, d using the PASTd method and go to the next time step.

Combining the KNN method with PASTd algorithm, our complete forecasting algorithm is shown in Algorithm 2.

---

Algorithm 2. Forcasting Algorithm

---

0. **Initialize**: $k$, $knn$, $h_p$, $h_f$, $l$

1. For each time $t$, receive speed values $x_t \in \mathbb{R}^n$.

2. Compute the corresponding hidden variables $z_t \in \mathbb{R}^k$ by PASTd algorithm.

3. Collect the hidden variables into a matrix for each day for consecutive l weeks as historical data.

4. Forecast the hidden variable for the ($l$ + 1)-th week using KNN method.

5. Forecast the speed values through the hidden variables generated at the last step.

6. Compute the error between forecasting and actual speeds.

---

The description of $k$, $knn$, $h_p$, $h_f$ , $l$ can be found in Table 1. The specific steps of PASTd are explained in the previous chapter , which is an online algorithm whose cost is $O(nk)$ both in terms of time and of space. We have shown that a few hidden variables are able to capture the most important patterns of traffic flow data. For each day, a $k \times 1440$ matrix is stored. Thus, for $l$ consecutive weeks, the cost of storage is $O(k\,l)$, which is much smaller compared to the original $O(n\,l)$ since $k \ll n$.

## 3.4 Forecasting results

In this section, we show the forecasting results by using the proposed forecasting algorithm. We choose both MSE(Mean Square Error) and MAPE(Mean Absolute Proportion Error) as the performance measurements. The definitions of MSE and MAPE are:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - x_i\right)^2$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - x_i}{x_i}\right| \times 100 \qquad (2)$$

The data used to evaluate the performance of our algorithm are the vehicle speeds collected in Baltimore County, Maryland, which contain 1751 road segments. We set the speeds during the first 40 weeks in 2014 as the sample data and try to forecast the speeds for the 41st week. To get the best time and space efficiency, we choose $k$ = 1, $knn$ = 1, $h_p$ = 1, and test the performance by varying the forecasting horizon $h_f$.

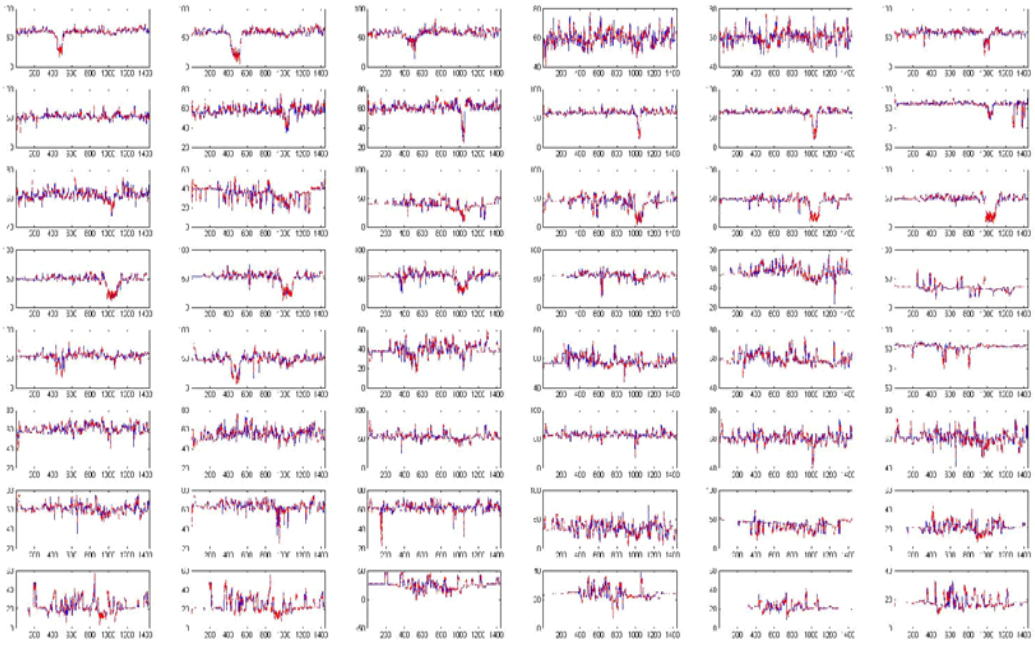

Figure 3.3: Forecasting results with $h_f = 60$ for the first 48 road segments

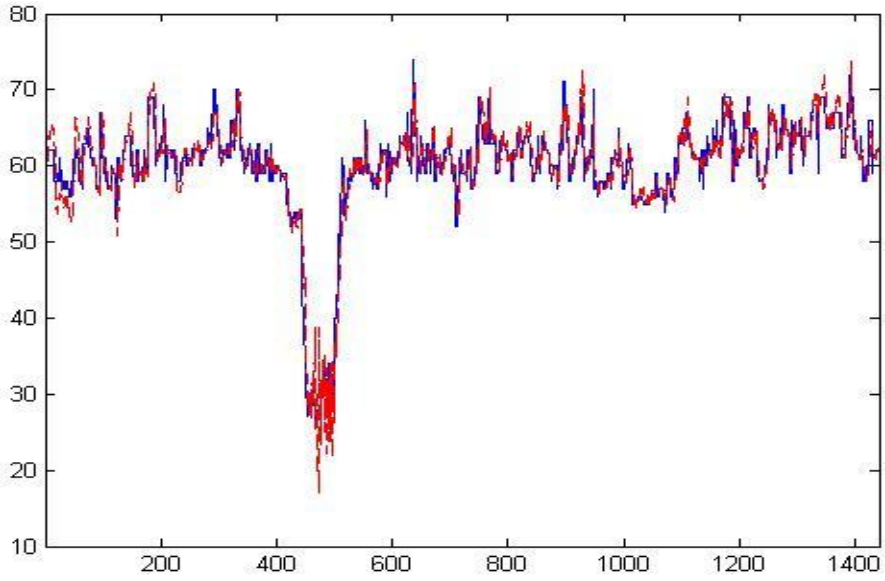


Figure 3.4: Forecasting results with $h_f = 60$ for the first road segments

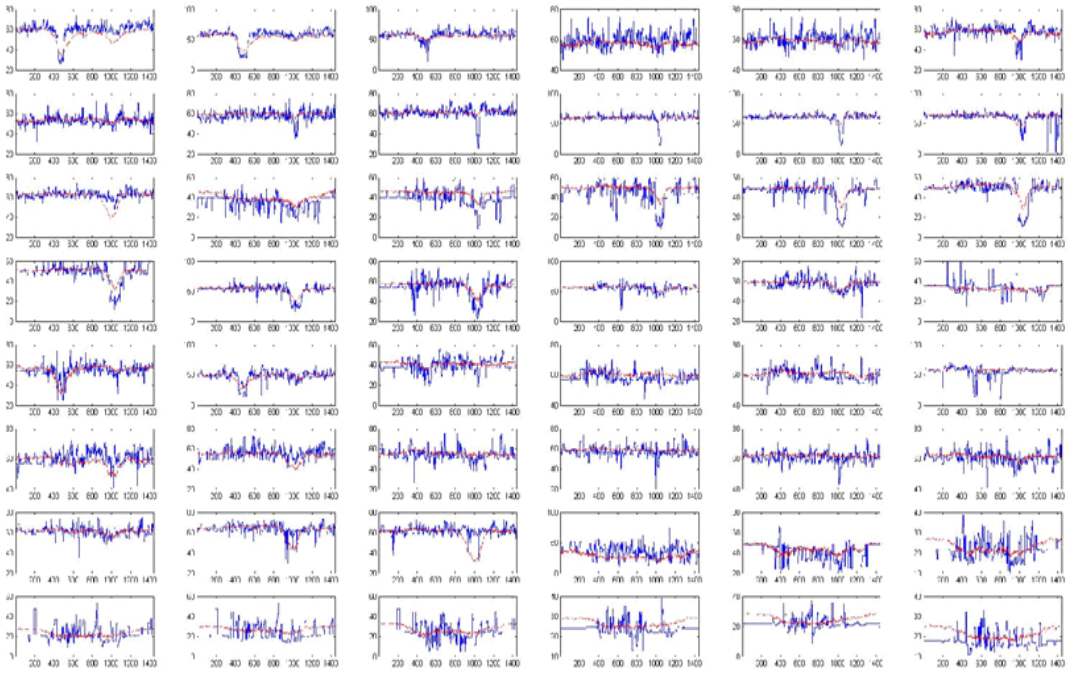

Figure 3.5: Forecasting results for the first 48 road segments using historical mean

**Forecasting Results** Figure 3.3 shows the forecasting results with $h_f = 60$ for the first 48 road segments, where the blue line represent the actual speeds while the red line corresponds to the forecasting speeds. Figure 3.4 shows the results for the first road segment more clearly. As we can see, our forecasting algorithm captures most statistical variance and has small deviation from the true data.

As a result, we reduce the data dimension from 1751 to 1. Instead of taking care of all the road segments, we are able to focus on just one hidden variable and forecast the speeds with tolerable error. Table 3.2 shows the MSE with different forecasting horizons. As we can see, when $h_f$ increases, MSE does not increase fast. Thus, our method is useful for making short-term prediction.

| $h_f$ | 1 | 5 | 10 | 30 | 60 |
|---|---|---|---|---|---|
| MSE | 5.06 | 6.01 | 6.30 | 7.26 | 8.74 |
| MAPE(%) | 3.75 | 4.41 | 4.54 | 4.83 | 5.20 |

Table 3.2: MSE and MAPE with $k = 1$, $knn = 1$, $h_p = 1$

To further show the advantage of our forecasting algorithm, we compare our method to the historical mean method, which also reduces the data dimension to 1. Figure 3.5 shows the forecasting results for historical mean method. As we can see, the historical mean method can barely capture the statistical variance and its MSE=45.60, which is much larger than ours.

Figure 3.6 shows the MSE during Wednesday. It can be seen that the MSE is larger during peak hours than off-peak hours. This is reasonable since during peak hours, speeds vary more significantly, making it more difficult to forecasting. On the other hand, Fig 3.7 shows the MSE over a whole week. Note that our week starts with Sunday. It can be seen that MSE is relatively small for weekends since there is not an obvious peak hour so the predictions are smoother than weekdays.

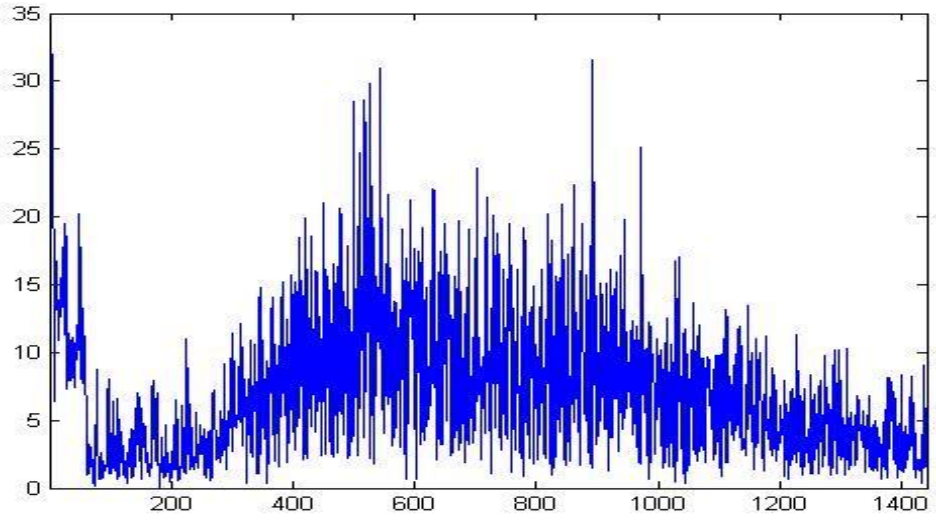


Figure 3.6: MSE with $k$ = 1, $knn$ = 1, $h_p$ = 1, $h_f$ = 60 during Wednesday

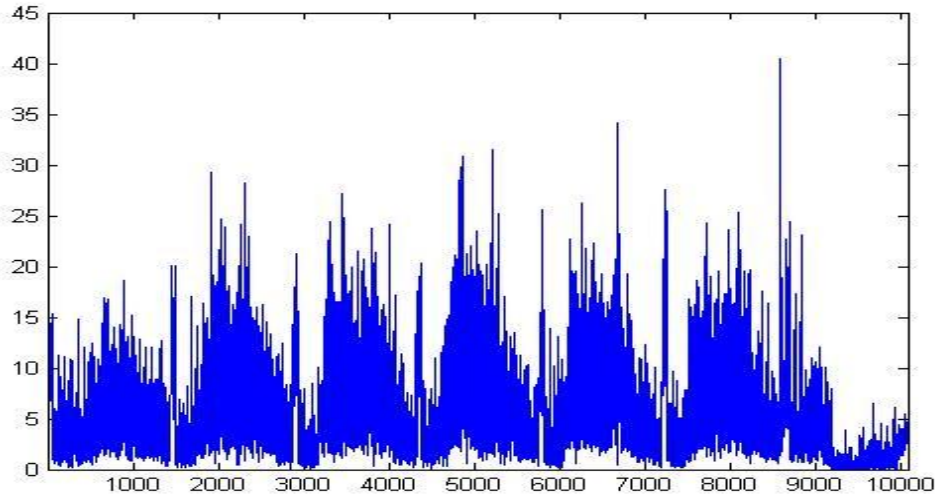


Figure 3.7: MSE with $k$ = 1, $knn$ = 1, $h_p$ = 1, $h_f$ = 60 over a week

**Update Sample data** After the $(l+1)$-th week, sample data need to be updated to forecast the speeds during the $(l+2)$-th week. In our algorithm, we would abandon the sample data at first week and add the data corresponding to the $(l+1)$-th week. Thus, the space to store historical data is fixed. Figure 3.8 shows the forecasting results for one road segment while Figure 3.9 shows the MSE for the whole week. The results are similar to the previous ones, which provide another validation of our method.

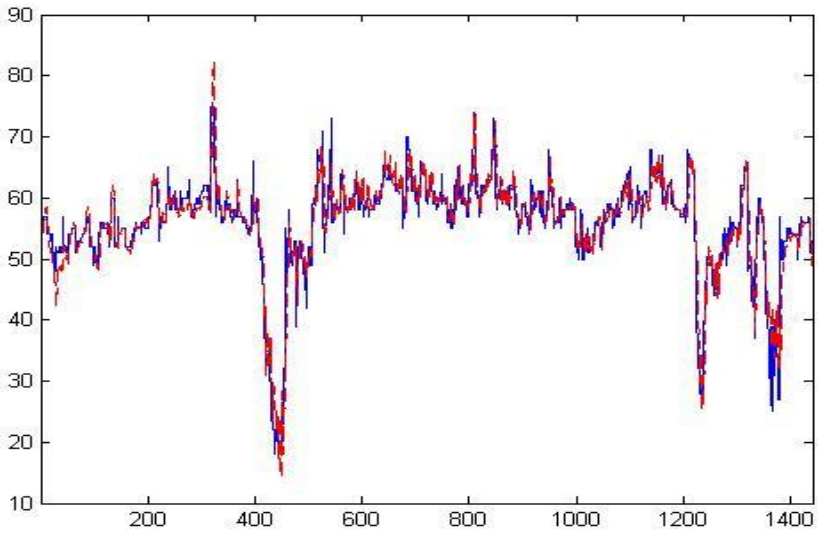

Figure 3.8: Forecasting results with $h_f = 60$ for one segment

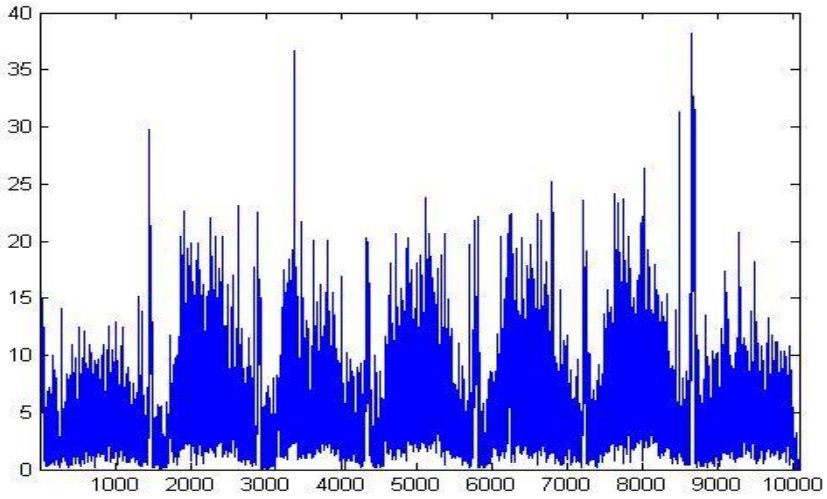

Figure 3.9: MSE with $k$ = 1, $knn$ = 1, $h_p$ = 1, $h_f$ = 60 over the next week

## 3.5 Impact of parameters

In this section, we try to find the best parameters ($k$, $knn$, $h_p$) for our algorithm. We choose both MSE and MAPE as the performance metrics.

### 3.5.1. Dense Road Segments

The data used to evaluate the performance of our algorithm are the vehicle speeds collected in Baltimore County, Maryland, which contain 1751 road segments. We set the speeds during the first 40 weeks in 2014 as the sample data and try to forecast the speeds at the 41st week. We applied $k$ nearest neighbor to forecast the speed.

#### 3.5.1.1 MSE - **horizon=1**

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.80 | 5.75 | 5.75 | 5.78 | 5.82 |
| 2 | 5.75 | 5.72 | 5.73 | 5.76 | 5.80 |
| 3 | 5.75 | 5.73 | 5.75 | 5.78 | 5.82 |
| 4 | 5.76 | 5.75 | 5.77 | 5.81 | 5.85 |
| 5 | 5.77 | 5.77 | 5.80 | 5.84 | 5.88 |

Table 3.3: MSE with $k$ = 1, $\boldsymbol{h_f}$ = **1**

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.81 | 5.73 | 5.74 | 5.82 | 8.49 |
| 2 | 5.61 | 5.53 | 5.65 | 6.15 | 6.82 |
| 3 | 5.58 | 5.52 | 5.56 | 5.96 | 6.34 |
| 4 | 5.59 | 5.50 | 5.66 | 5.84 | 6.10 |
| 5 | 5.57 | 5.50 | 5.63 | 5.78 | 5.99 |

Table 3.4: MSE with $k = 2$, $h_f = 1$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 6.51 | 6.80 | 6.89 | 7.05 | 9.47 |
| 2 | 5.99 | 6.08 | 6.19 | 6.69 | 7.26 |
| 3 | 5.87 | 5.96 | 5.99 | 6.26 | 6.62 |
| 4 | 5.82 | 5.79 | 5.94 | 6.11 | 6.33 |
| 5 | 5.76 | 5.72 | 5.82 | 5.96 | 6.18 |

Table 3.5: MSE with $k = 3$, $h_f = 1$

**Comment** As we can see, the best performance is $k = 2$, which is better than $k = 1$. For $k = 3$, we may need more parameters ($k$ nearest neighbors) to reach the best performance. The following Figures(3.10-3.12) are the forecasting results for two road segments with the best values for $h_p$ and $knn$. The top figure corresponds to the

road segment with the largest MSE while the bottom figure corresponds to the road segment with the smallest MSE.

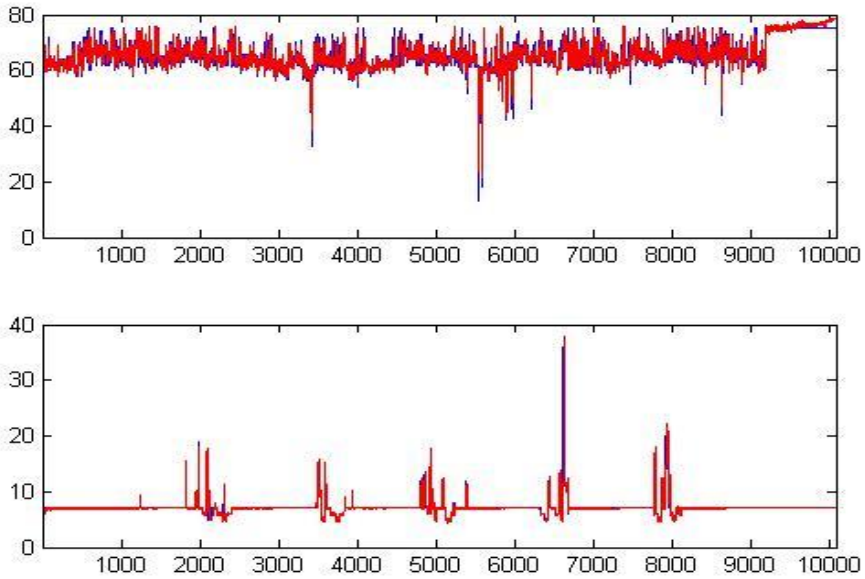


Figure 3.10: Forecasting results with $k = 1$, $h_f = 1$

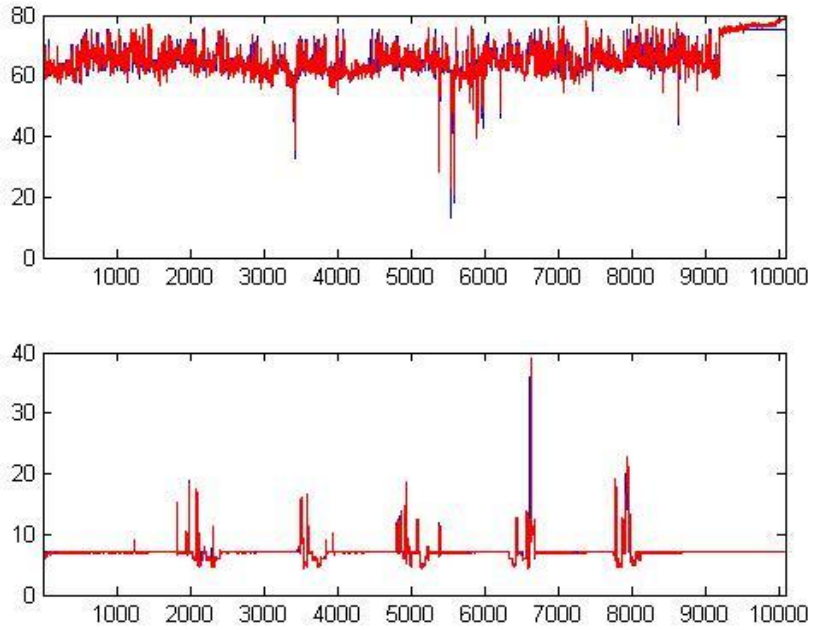


Figure 3.11: Forecasting results with $k = 2$, $h_f = 1$

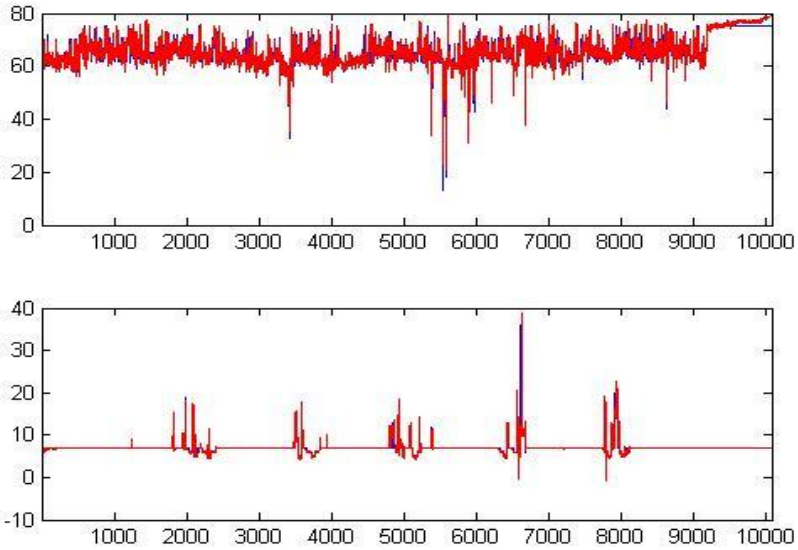


Figure 3.12: Forecasting results with $k = 3$, $h_f = 1$

**Comment** The road segments whose speeds change frequently may have larger MSE. However, for $k = 1$, even in the worst case, our method can still capture the speed changes and make accurate prediction very quickly.

**horizon**=60

To see the limit of our method, this time we try to predict the speeds for the next hour.

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 8.74 | 8.73 | 8.75 | 8.80 | 8.83 |
| 2 | 8.64 | 8.63 | 8.65 | 8.68 | 8.72 |
| 3 | 8.60 | 8.59 | 8.62 | 8.66 | 8.70 |
| 4 | 8.58 | 8.58 | 8.61 | 8.65 | 8.69 |
| 5 | 8.58 | 8.58 | 8.61 | 8.64 | 8.69 |

Table 3.6: MSE with $k = 1$, $h_f = 60$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 9.30 | 9.24 | 125.04 | 125.11 | 125.22 |
| 2 | 13.73 | 26.24 | 55.88 | 52.17 | 41.80 |
| 3 | 12.58 | 20.45 | 34.57 | 31.30 | 24.21 |
| 4 | 12.11 | 17.24 | 26.72 | 23.89 | 17.76 |
| 5 | 11.79 | 15.28 | 22.70 | 20.39 | 15.07 |

Table 3.7: MSE with $k = 2$, $h_f = 60$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 9.94 | 9.87 | 125.65 | 125.65 | 125.82 |
| 2 | 14.12 | 26.59 | 56.21 | 52.48 | 42.12 |
| 3 | 12.90 | 20.70 | 34.81 | 31.52 | |
| 4 | 12.39 | 17.45 | 26.91 | 24.07 | 17.97 |
| 5 | 12.05 | 15.45 | 22.86 | 20.54 | 15.25 |

Table 3.8: MSE with $k$ =3, $h_f = 60$

**Comment** From the tables above(Table 3.6-3.8), we can see that in this case, $k = 1$ gives the best and most stable performance. The reason may be that our equation is $\mathbf{x}(t+h_f) = \hat{z}_{1?}(t+h_f)\mathbf{w}_1(t) + ... + \hat{z}_k(t+h_f)\mathbf{w}_k(t).w_i(t)$ might be quite different from $w_i(t+h_f)$ for larger $i$. As a result, a larger $k$ might lead to larger error. Following are the forecasting results with $h_f = 60$.

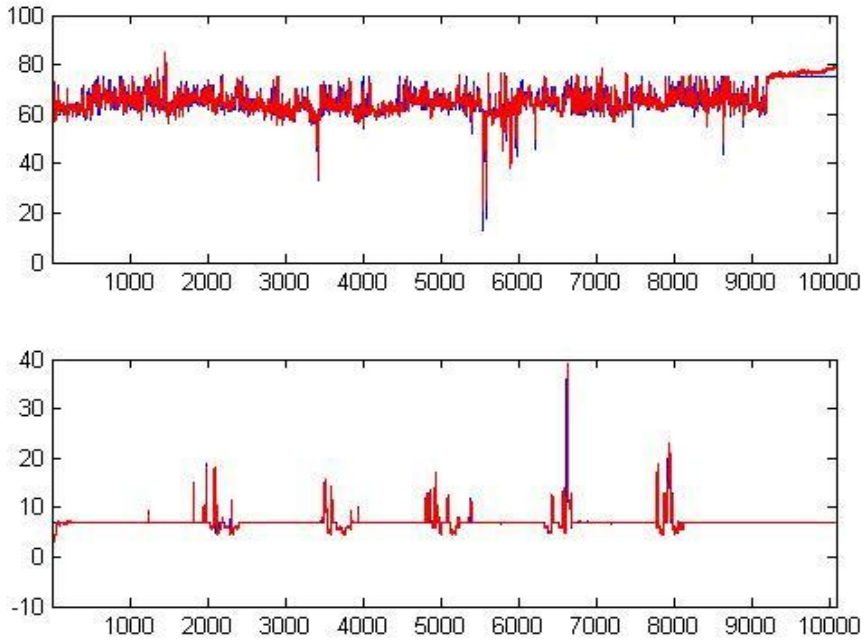


Figure 3.13: Forecasting results with $k = 1$, $h_f = 60$

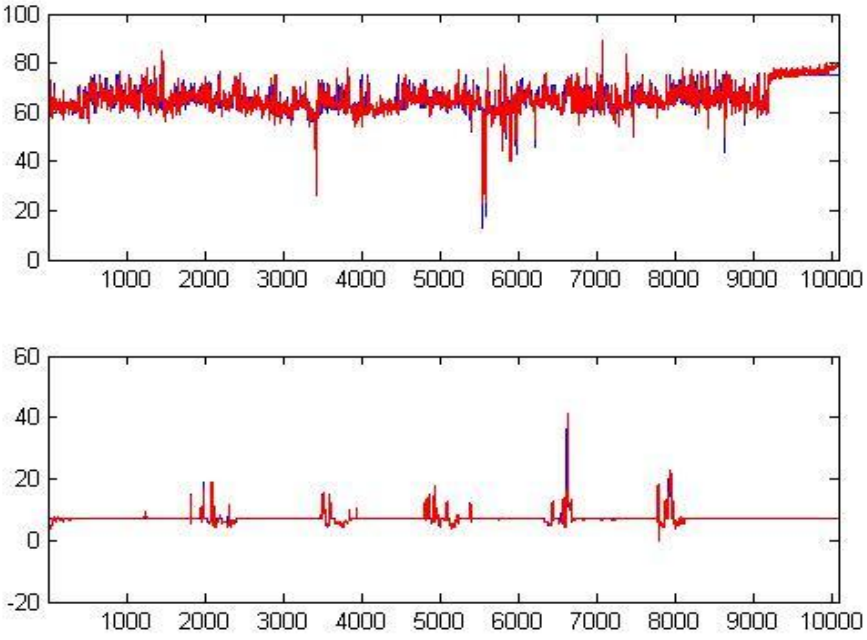


Figure 3.14: Forecasting results $k = 2$, $h_f = 60$

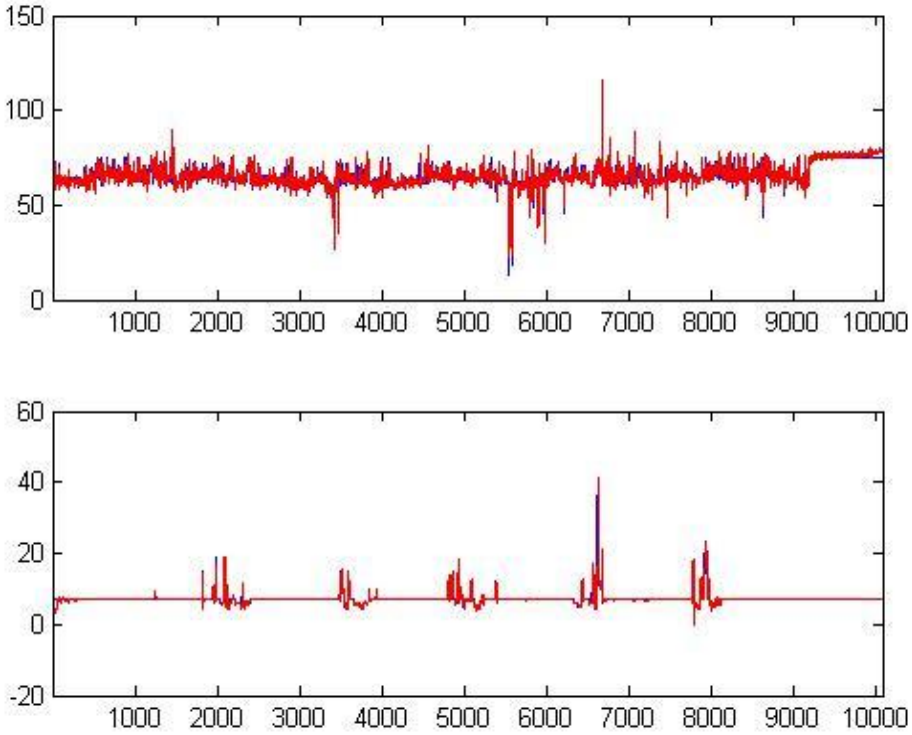


Figure 3.15: Forecasting results with $k = 3$, $h_f = 60$

**Comment** The forecasting results show that our method can predict the speeds with high accuracy (Figure 3.13-3.15).

3.5.1.2 MAPE

Instead of the absolute error, we also would like to see the relative error. For example, how far our predicted speed is from the actual speed. Thus, we choose MAPE to capture the performance.

**horizon**=1

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 4.30 | 4.22 | 4.19 | 4.19 | 4.20 |
| 2 | 4.20 | 4.13 | 4.11 | 4.11 | 4.12 |
| 3 | 4.17 | 4.10 | 4.10 | 4.10 | 4.11 |
| 4 | 4.16 | 4.11 | 4.10 | 4.11 | 4.12 |
| 5 | 4.16 | 4.12 | 4.12 | 4.13 | 4.15 |

Table 3.9: MAPE with $k = 1$, $h_f = 1$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 4.14 | 4.05 | 4.04 | 4.05 | 4.09 |
| 2 | 3.98 | 3.91 | 3.90 | 3.92 | 3.93 |
| 3 | 3.94 | 3.86 | 3.87 | 3.88 | 3.90 |
| 4 | 3.92 | 3.85 | 3.86 | 3.88 | 3.90 |
| 5 | 3.91 | 3.86 | 3.87 | 3.89 | 3.91 |

Table 3.10: MAPE with $k = 2$, $h_f = 1$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 4.31 | 4.27 | 4.27 | 4.29 | 4.33 |
| 2 | 4.10 | 4.05 | 4.05 | 4.07 | 4.09 |
| 3 | 4.03 | 3.97 | 3.98 | 3.99 | 4.01 |
| 4 | 3.99 | 3.93 | 3.94 | 3.96 | 3.99 |
| 5 | 3.98 | 3.92 | 3.93 | 3.95 | 3.99 |

Table 3.11: MAPE with $k = 3$, $h_f = 1$

**Comment** From the tables above (Table 3.9-3.11) we can see that $k = 2$ gives the best performance. For $k = 3$, we may need more parameters to get the best performance. Overall, MAPE is around 4%, meaning that the results are generally accurate.

The following Figures(3.16-3.18) are the forecasting results for two road segments. The top one is the road segment with largest MAPE while the bottom one corresponds to the road segment with the smallest MAPE value.

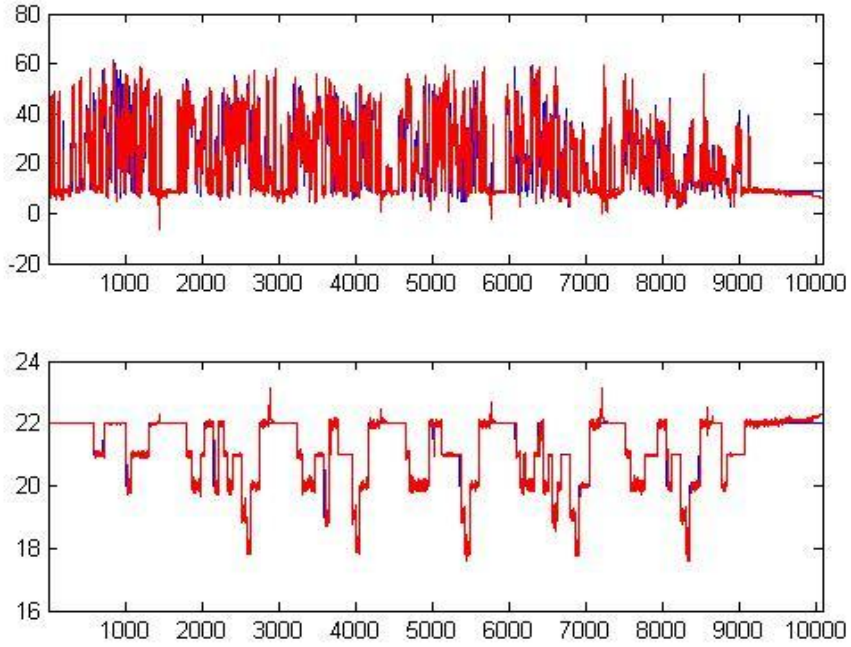


Figure 3.16: Forecasting results with $k = 1$, $h_f = 1$

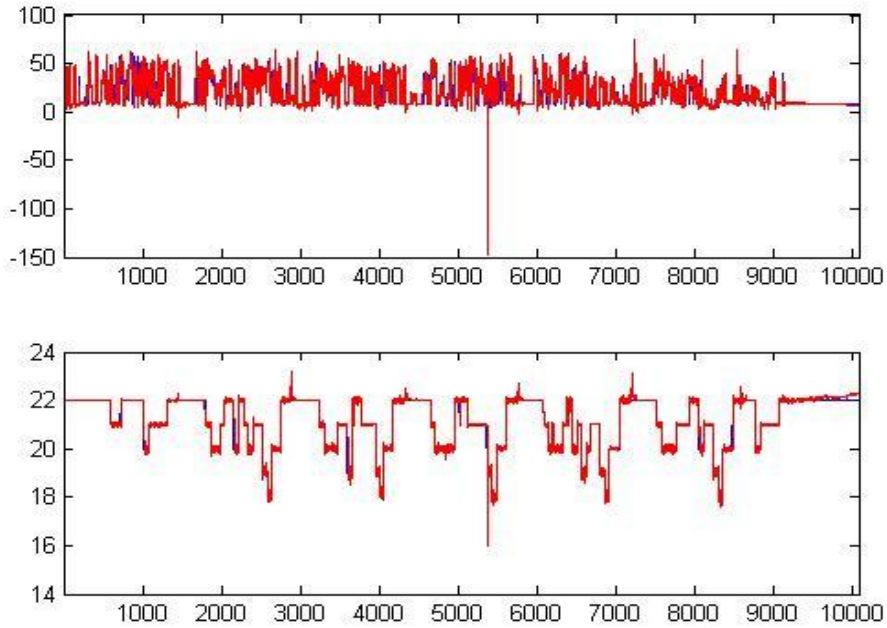


Figure 3.17: Forecasting results with $k = 2$, $h_f = 1$

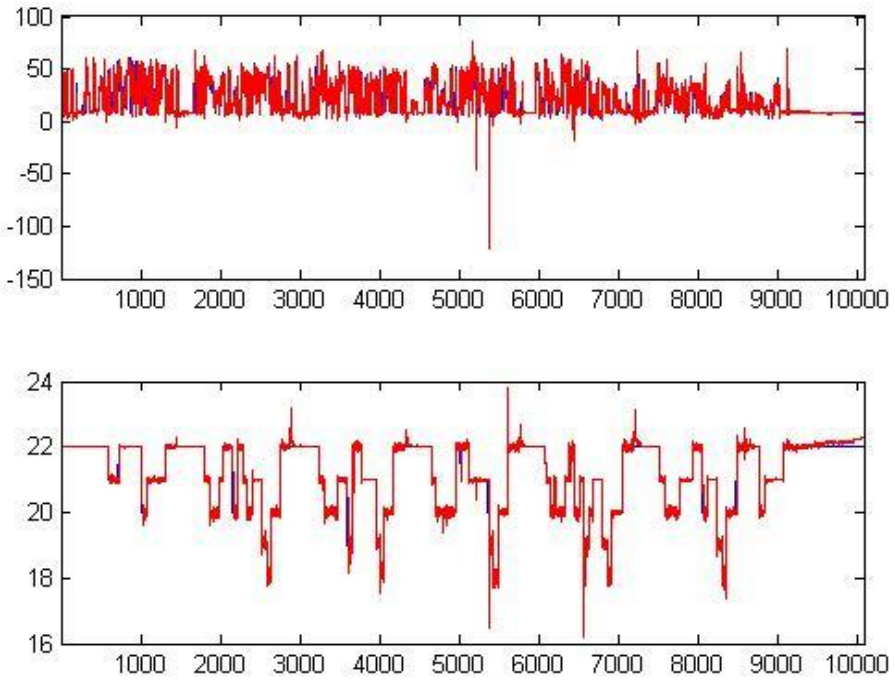


Figure 3.18: Forecasting results with $k = 3$, $h_f = 1$

**Comment** For $k = 2$ and $k = 3$, there are some unusual errors. For instance, speeds are negative in some time steps. The reason may be that higher order hidden variables change more frequently and irregularly so they are more difficult to predict. On the other hand, for $k = 1$, we achieve the most accurate prediction.

**horizon**=60

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.20 | 5.16 | 5.15 | 5.16 | 5.17 |
| 2 | 5.07 | 5.03 | 5.01 | 5.01 | 5.02 |
| 3 | 5.02 | 4.97 | 4.96 | 4.96 | 4.97 |
| 4 | 4.99 | 4.94 | 4.93 | 4.94 | 4.94 |
| 5 | 4.98 | 4.93 | 4.91 | 4.92 | 4.93 |

Table 3.12: MAPE with $k = 1$, $h_f = 60$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.40 | 5.34 | 5.57 | 5.59 | 5.59 |
| 2 | 5.24 | 5.22 | 5.26 | 5.26 | 5.26 |
| 3 | 5.16 | 5.13 | 5.14 | 5.14 | 5.13 |
| 4 | 5.11 | 5.07 | 5.09 | 5.08 | 5.07 |
| 5 | 5.09 | 5.04 | 5.05 | 5.05 | 5.03 |

Table 3.13: MAPE with $k = 2$, $h_f = 60$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.55 | 5.49 | 5.71 | 5.74 | 5.75 |
| 2 | 5.35 | 5.32 | 5.36 | 5.35 | 5.35 |
| 3 | 5.25 | 5.21 | 5.21 | 5.21 | 5.21 |
| 4 | 5.20 | 5.14 | 5.15 | 5.13 | 5.13 |
| 5 | 5.17 | 5.09 | 5.10 | 5.10 | 5.09 |

Table 3.14: MAPE with $k = 3$, $h_f = 60$

**Comment** From the tables above (Table 3.12-3.14) we can see that for larger horizons, it might be better choose $k = 1$, which is consistent with MSE. Even for $h_f = 60$, MAPE is around 5%, showing that our algorithm is able to make accurate short-term predictions (Figure 3.19-3.21).

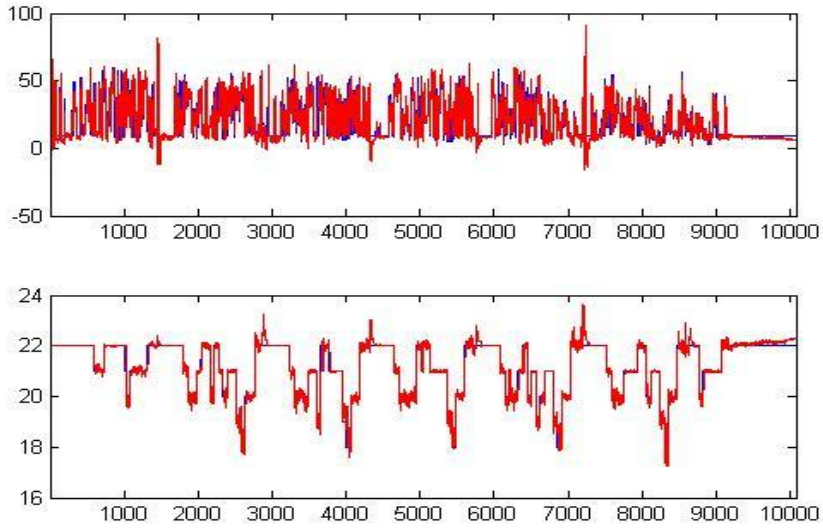


Figure 3.19: Forecasting results with $k = 1$, $h_f = 60$

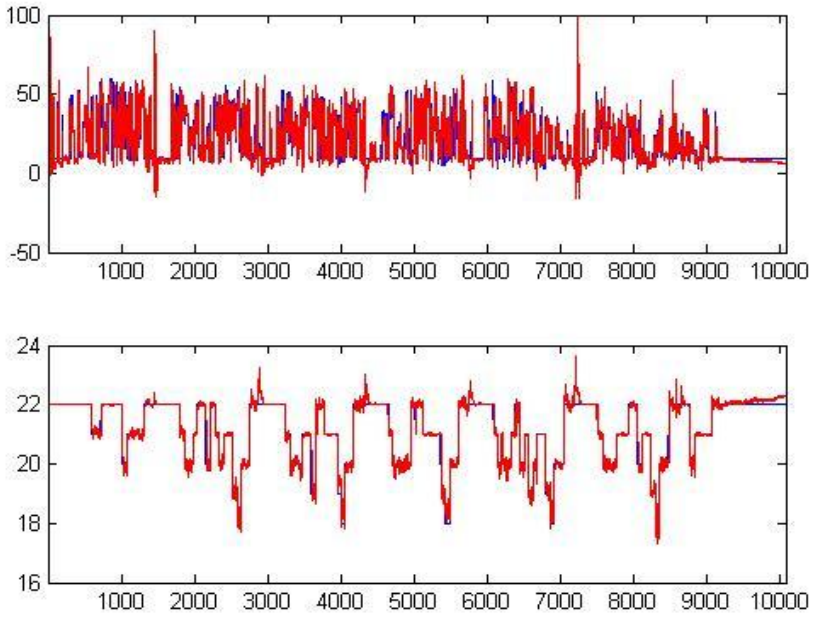


Figure 3.20: Forecasting results with $k = 2$, $h_f = 60$

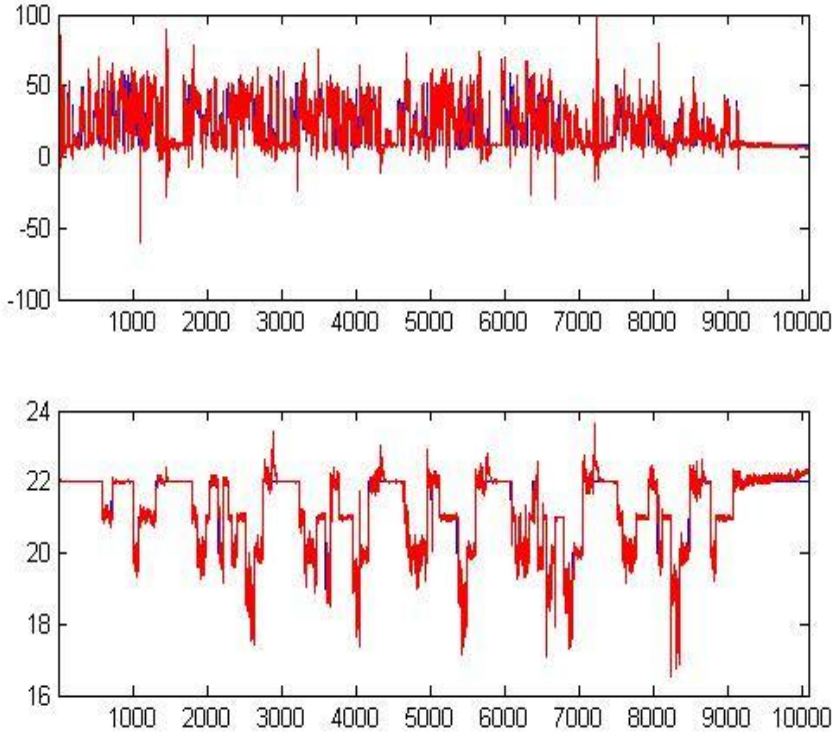


Figure 3.21: Forecasting results with $k = 3$, $h_f = 60$

3.5.1.3 Conclusion

In this section, we predict the speeds of all road segments in Baltimore county over a week using the *k* nearest neighbor method. We explore the impact of the values of the parameters by using the performance metrics defined by MSE and MAPE. For small horizon ($h_f = 1$), $k = 2$ leads to the best results while for large horizon ($h_f = 60$) we better choose $k = 1$. As for $h_p$ and *knn*, the tables show that $h_p = 2$ or 3 and *knn* = 4 or 5 can offer stable results. To wrap up, the best parameters in this case are $k = 1$, $h_p = 2$, *knn* = 4 for forecasting horizons from 1 to 60.

### 3.5.2.Scattered Road Segments

In the last section, we tried to predict average speeds on the roads of only one county, where road segments are close to each other and have high spatial correlation. In this section, we used the data of three counties: Frederick, Prince Georges', and Wicomico in Maryland. Again, we test our algorithm regarding both MSE and MAPE.

#### 3.5.2.1 MSE - **horizon**=1

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.77 | 5.92 | 5.86 | 5.89 | 5.90 |
| 2 | 5.80 | 5.71 | 5.69 | 5.70 | 5.71 |
| 3 | 5.79 | 5.71 | 5.69 | 5.71 | 5.72 |
| 4 | 5.80 | 5.73 | 5.72 | 5.74 | 5.75 |
| 5 | 5.83 | 5.77 | 5.76 | 5.78 | 5.79 |

Table 3.15: MSE with $k = 1$, $h_f = 1$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5.85 | 5.99 | 37.44 | 37.48 | 6.05 |
| 2 | 5.78 | 10.78 | 14.74 | 14.07 | 11.01 |
| 3 | 5.70 | 8.49 | 10.13 | 9.54 | 8.40 |
| 4 | 40.45 | 7.46 | 8.45 | 7.92 | 7.33 |
| 5 | 30.09 | 6.94 | 7.56 | 7.33 | 6.82 |

Table 3.16: MSE with $k = 2$, $h_f = 1$

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 6.58 | 7.22 | 38.74 | 38.47 | 7.24 |
| 2 | 6.22 | 11.43 | 15.42 | 14.56 | 11.60 |
| 3 | 6.04 | 8.89 | 10.56 | 9.88 | 8.81 |
| 4 | 40.72 | 7.75 | 8.73 | 8.16 | 7.61 |
| 5 | 30.31 | 7.18 | 7.95 | 7.51 | 7.02 |

Table 3.17: MSE with $k = 3$, $h_f = 1$

**Comment** From the Table(3.15-3.17), $k = 1$ gives the best and most stable performance. The reason may be that different counties result in more frequently changing hidden variables. This can also be seen in the following Figure(3.22-3.24).

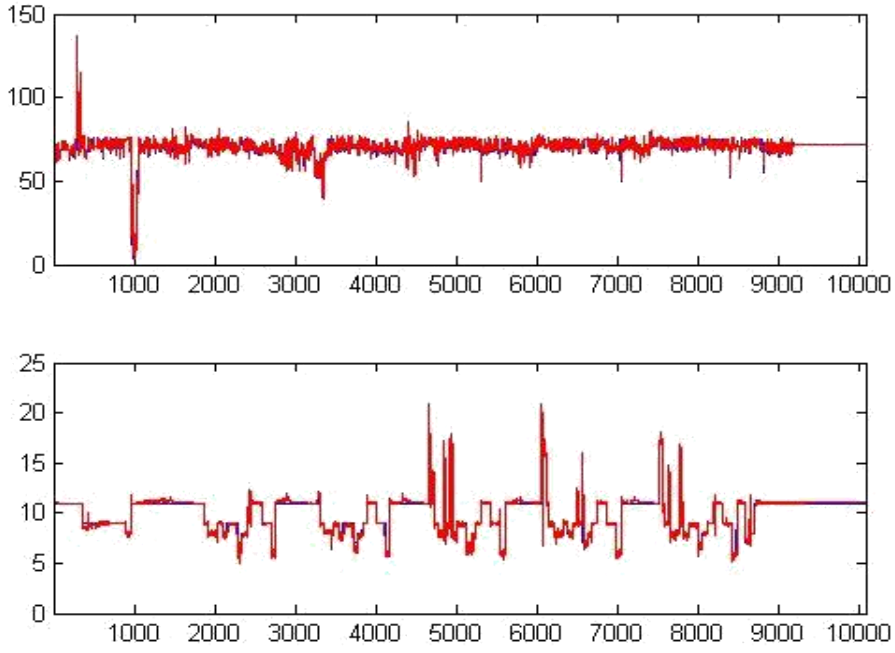


Figure 3.22: Forecasting results with $k = 1$, $h_f = 1$

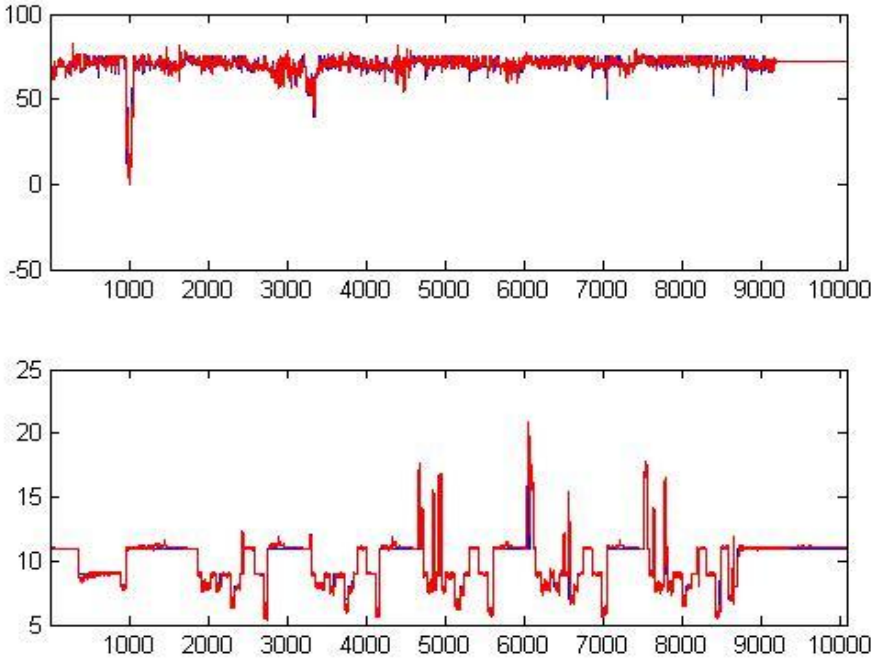


Figure 3.23: Forecasting results with $k = 2$, $h_f = 1$

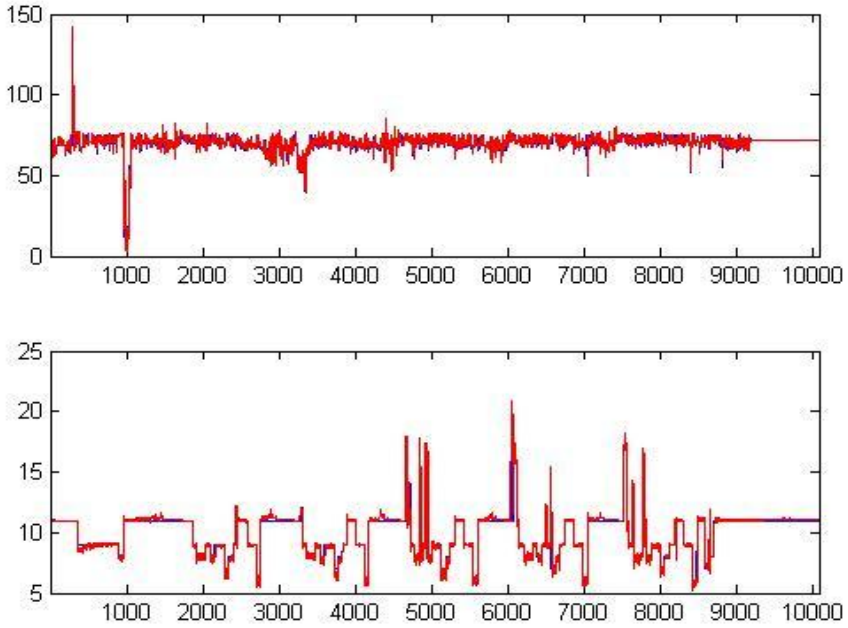


Figure 3.24: Forecasting results with $k = 3$, $h_f = 1$

**Comment** For $k$ = 2 and $k$ = 3, there are some unusual errors. For example, the speeds are over 100 mph. The reason may also be the number of nearest neighbors and past values are too small. Only one past value is not enough. $k$ = 1 gives the best prediction.

**horizon**=60

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 7.38 | 7.33 | 7.32 | 7.33 | 7.32 |
| 2 | 7.24 | 7.18 | 7.17 | 7.17 | 7.18 |
| 3 | 7.21 | 7.16 | 7.15 | 7.15 | 7.16 |
| 4 | 7.21 | 7.16 | 7.15 | 7.16 | 7.17 |
| 5 | 7.22 | 7.17 | 7.16 | 7.17 | 7.18 |

Table 3.18: MSE with $k$ = 1, $h_f$ = 60

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 7.95 | 7.92 | 7.88 | 7.92 | 7.90 |
| 2 | 7.62 | 7.51 | 7.47 | 7.49 | 7.49 |
| 3 | 7.52 | 7.39 | 7.36 | 7.37 | 7.36 |
| 4 | 7.48 | 7.35 | 7.32 | 7.32 | 7.33 |
| 5 | 7.46 | 7.33 | 7.30 | 7.31 | 7.31 |

Table 3.19: MSE with $k$ = 2, $h_f$ = 60

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 8.51 | 8.49 | 8.71 | 8.69 | 8.50 |
| 2 | 8.04 | 7.87 | 7.89 | 7.88 | 7.82 |
| 3 | 7.85 | 7.65 | 7.66 | 7.64 | 7.60 |
| 4 | 7.77 | 7.56 | 7.56 | 7.53 | 7.52 |
| 5 | 7.72 | 7.51 | 7.49 | 7.47 | 7.46 |

Table 3.20: MSE with $k = 3$, $h_f = 60$

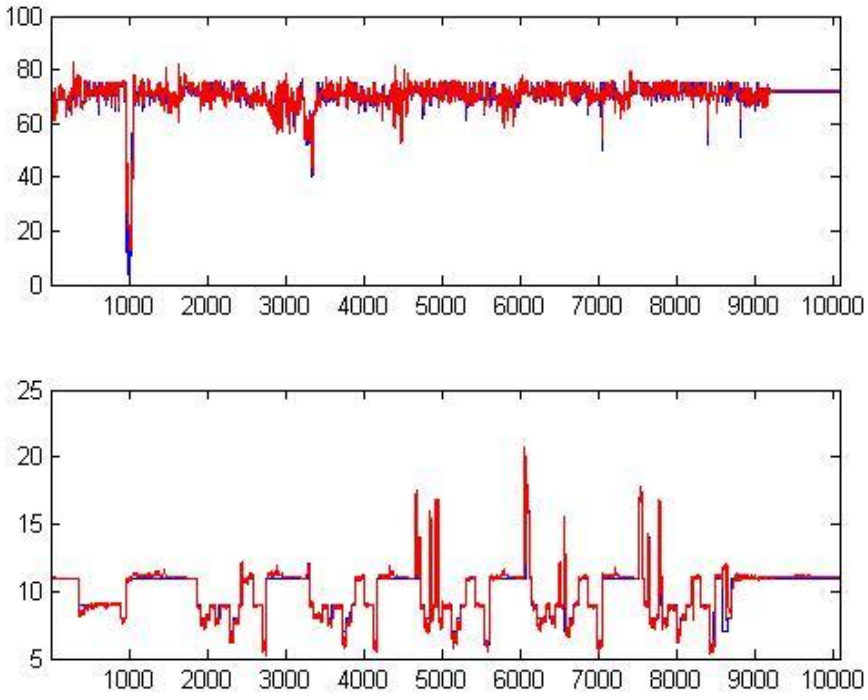


Figure 3.25: Forecasting results with $k = 1$, $h_f = 60$

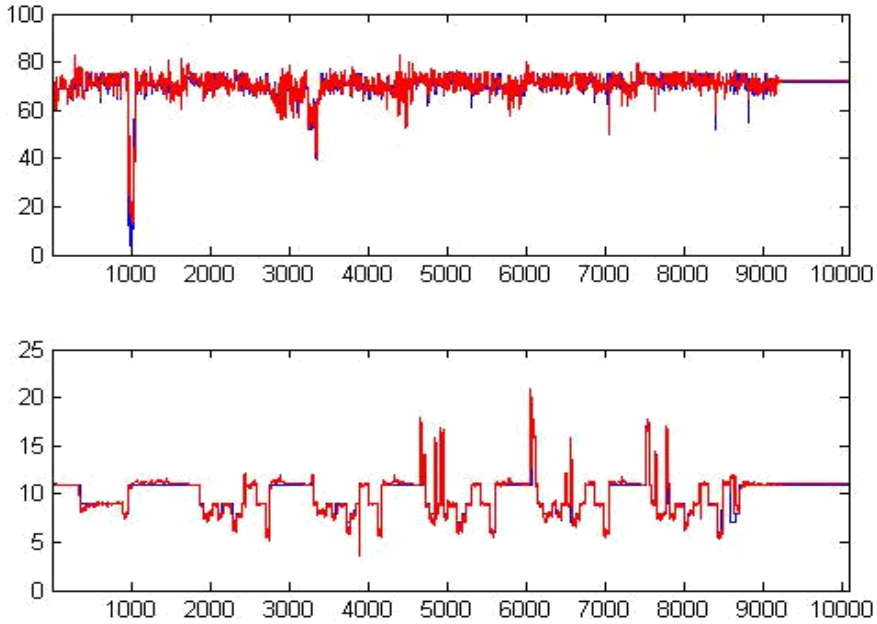


Figure 3.26: Forecasting results with $k = 2$, $h_f = 60$

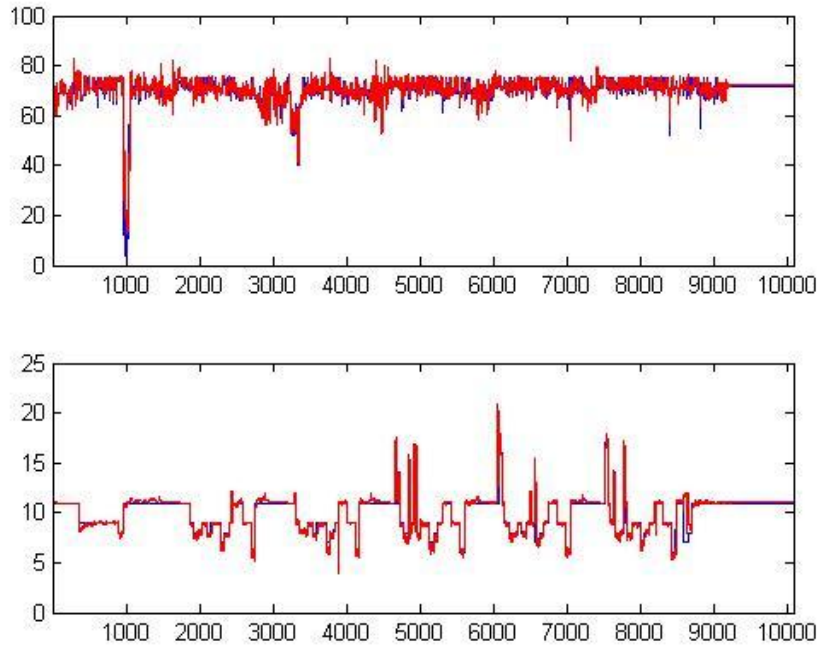


Figure 3.27: Forecasting results with $k = 3$, $h_f = 60$

**Comment** This time(Table 3.18-3.20) $k$ = 1, 2, 3 all give good prediction while $k$ = 1 is the best. As for $h_p$ and *knn*, it seems that $h_p$ = 3 and *knn* = 5 can give stable results (Figure 3.25-3.27).

3.5.2.2 MAPE

**horizon**=1

| *knn*/$h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3.04 | 2.95 | 2.92 | 2.91 | 2.91 |
| 2 | 2.98 | 2.89 | 2.87 | 2.86 | 2.86 |
| 3 | 2.97 | 2.89 | 2.87 | 2.87 | 2.87 |
| 4 | 2.98 | 2.90 | 2.89 | 2.89 | 2.89 |
| 5 | 2.99 | 2.92 | 2.91 | 2.90 | 2.91 |

Table 3.21: MAPE with $k$ = 1, $h_f$ = 1

| *knn*/$h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 2.97 | 2.87 | 2.95 | 2.95 | 2.86 |
| 2 | 2.86 | 2.81 | 2.81 | 2.81 | 2.80 |
| 3 | 2.84 | 2.79 | 2.78 | 2.78 | 2.77 |
| 4 | 2.92 | 2.78 | 2.77 | 2.78 | 2.78 |
| 5 | 2.92 | 2.79 | 2.78 | 2.78 | 2.79 |

Table 3.22: MAPE with $k$ = 2, $h_f$ = 1

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3.08 | 3.02 | 3.10 | 3.10 | 3.02 |
| 2 | 2.94 | 2.92 | 2.91 | 2.91 | 2.91 |
| 3 | 2.90 | 2.86 | 2.85 | 2.85 | 2.87 |
| 4 | 2.97 | 2.84 | 2.83 | 2.83 | 2.84 |
| 5 | 2.95 | 2.83 | 2.82 | 2.83 | 2.84 |

Table 3.23: MAPE with $k = 3$, $h_f = 1$

**Comment** As we can see (Table 3.21-3.23), MAPE values are less than 3%, meaning that overall our prediction is quite accurate. Although MSE is large in some cases, MAPE is generally small. If we can use some technique to avoid those unusual errors, the performance can be further improved. The following Figures (Figure3.28 through 3.30) are the corresponding forecasting results.

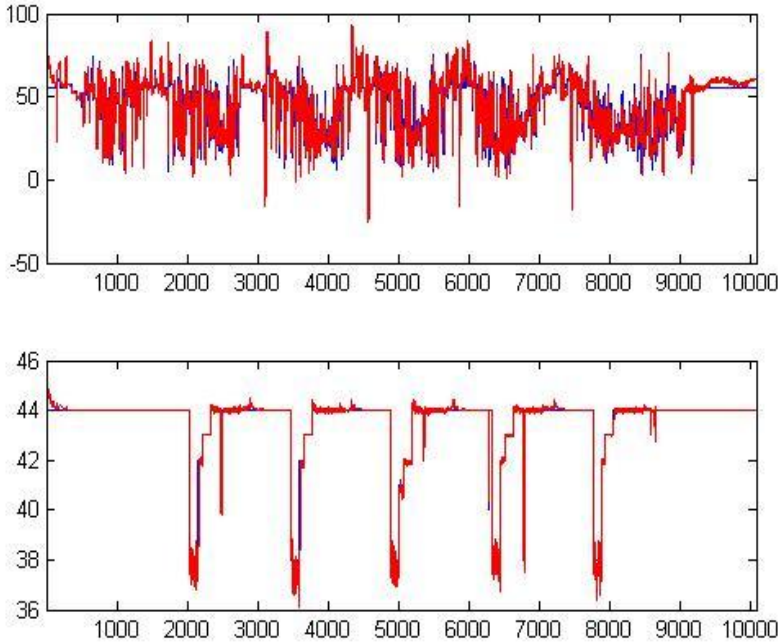


Figure 3.28: Forecasting results with $k = 1$, $h_f = 1$

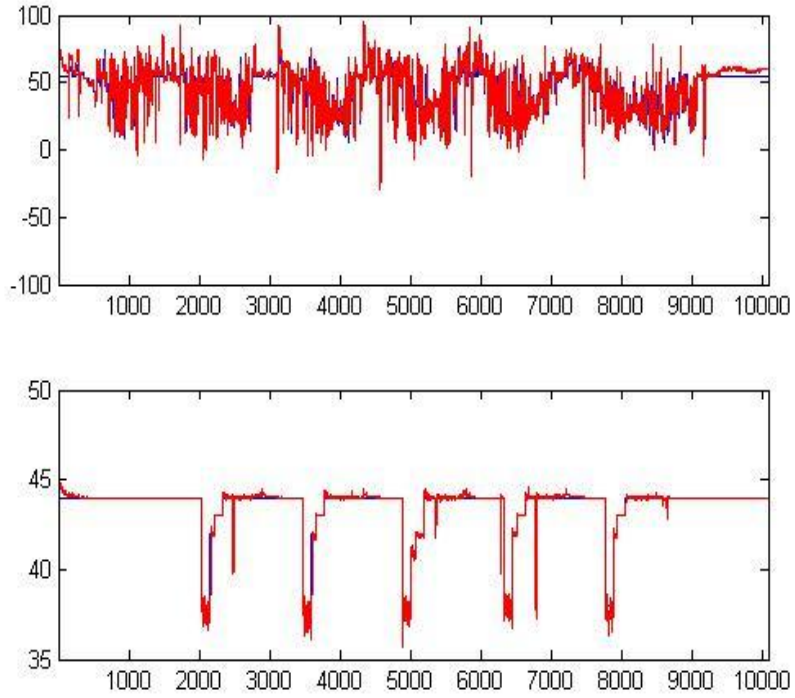


Figure 29: Forecasting results with $k = 2$, $h_f = 1$

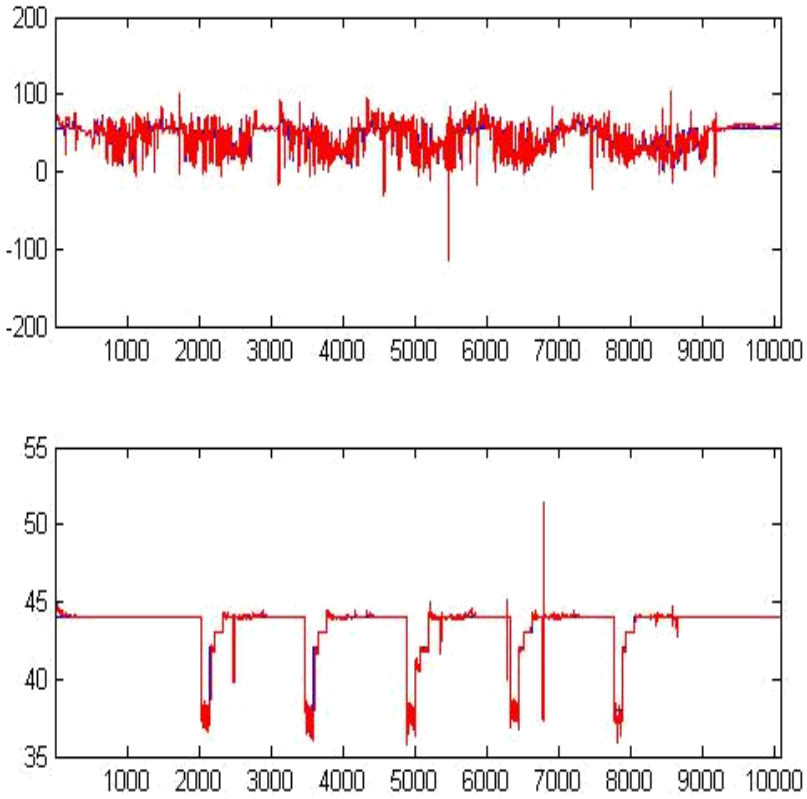


Figure 3.30: Forecasting results with $k = 3$, $h_f = 1$

**horizon**=60

| $knn/h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3.65 | 3.60 | 3.59 | 3.60 | 3.60 |
| 2 | 3.54 | 3.50 | 3.49 | 3.48 | 3.48 |
| 3 | 3.51 | 3.46 | 3.46 | 3.45 | 3.45 |
| 4 | 3.50 | 3.45 | 3.45 | 3.44 | 3.45 |
| 5 | 3.50 | 3.45 | 3.44 | 3.44 | 3.44 |

Table 3.24: MAPE with $k = 1$, $h_f = 60$

| knn/ $h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3.79 | 3.76 | 3.74 | 3.74 | 3.74 |
| 2 | 3.64 | 3.58 | 3.57 | 3.56 | 3.57 |
| 3 | 3.59 | 3.53 | 3.51 | 3.51 | 3.51 |
| 4 | 3.57 | 3.50 | 3.49 | 3.48 | 3.49 |
| 5 | 3.56 | 3.49 | 3.47 | 3.47 | 3.47 |

Table 3.25: MAPE with $k = 2$, $h_f = 60$

| knn/ $h_p$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3.89 | 3.86 | 3.85 | 3.84 | 3.84 |
| 2 | 3.72 | 3.65 | 3.64 | 3.63 | 3.63 |
| 3 | 3.66 | 3.58 | 3.57 | 3.56 | 3.56 |
| 4 | 3.63 | 3.55 | 3.54 | 3.53 | 3.53 |
| 5 | 3.61 | 3.53 | 3.52 | 3.51 | 3.51 |

Table 3.26: MAPE with $k = 3$, $h_f = 60$

**Comment** MAPE is around 3.5% for $h_f$ = 60(Table 3.24-3.26), showing our algorithm is able to predict the speeds for the next hour. As we can see, for larger horizons, we need to have more nearest neighbors (Figure 3.31-3.33).

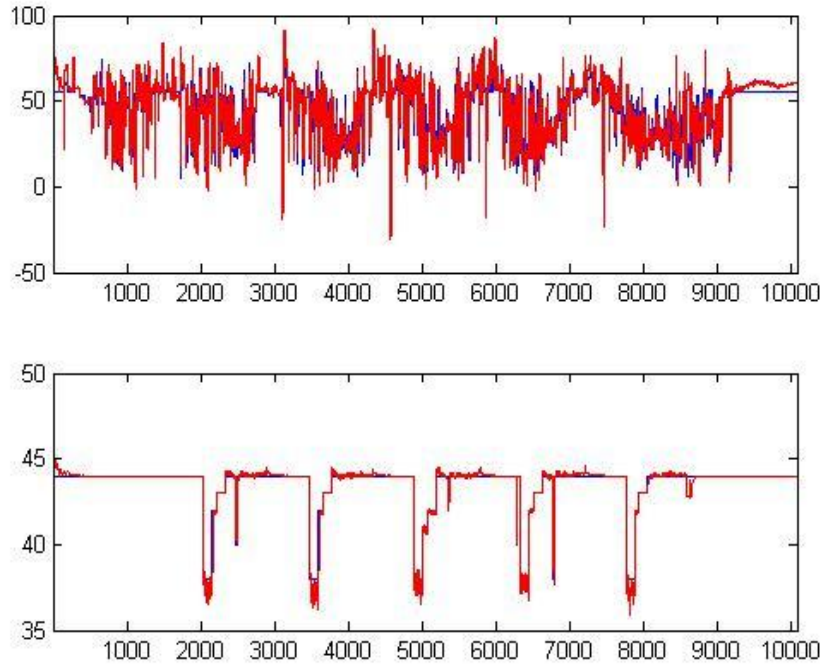


Figure 3.31: Forecasting results with $k = 1$, $h_f = 60$

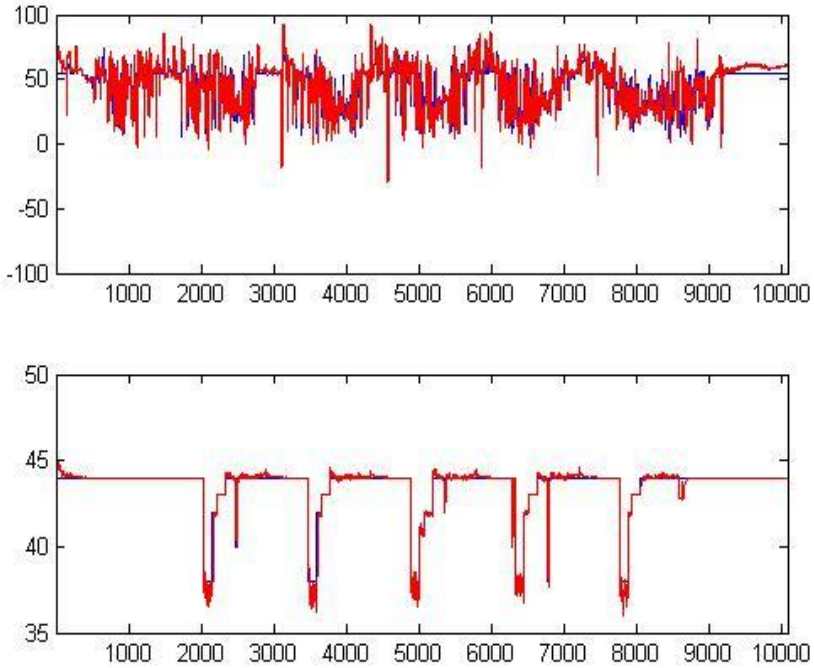


Figure 3.32: Forecasting results with $k = 2$, $h_f = 60$

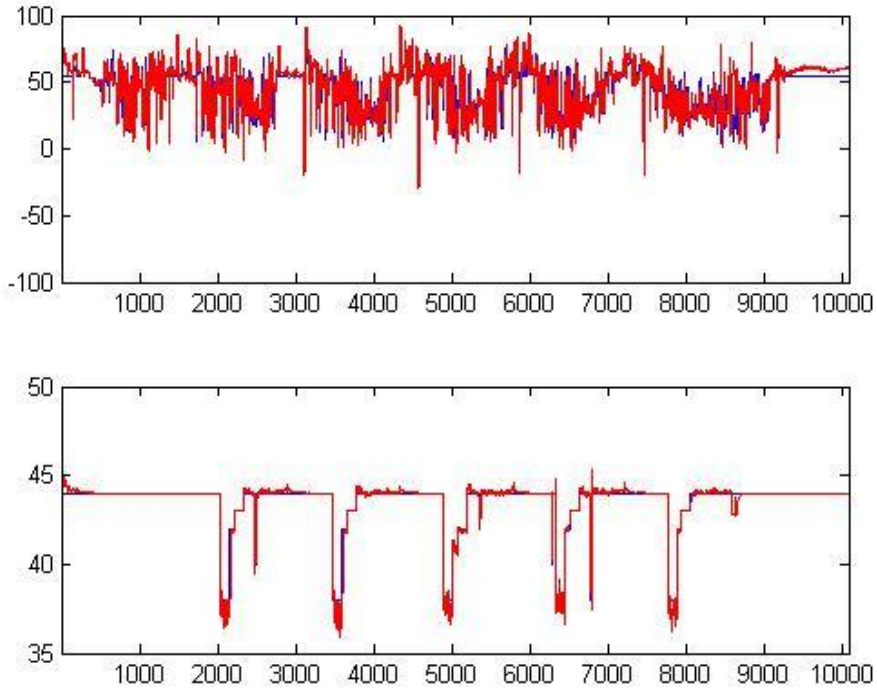


Figure 3.33: Forecasting results with $k = 3$, $h_f = 60$

#### 3.5.2.3 Conclusion

For scattered road segments, our algorithm is also able to predict the speeds for time horizons from 1 to 60. This time, $k = 1$ seems to be the best choice. The reason may be that for scattered road segments the high order hidden variables change too fast. As a result, the KNN method is unable to predict those hidden variables accurately, leading to larger errors. However, $k = 1$ gives quite satisfactory results. $h_p = 3$, $knn = 5$ are good choices for these parameters.

## 3.6 Conclusion

In this chapter, we forecast the vehicle speeds using a combination of the PASTd algorithm and the KNN method. Firstly, PASTd method reduces the data dimension

from a large number $n$ to a much smaller $k$ (in our test, $n$ = 1751, $k$ = 1). As has been discussed in this chapter, we can adaptively change values of parameters in PASTd algorithm to improve performance. Hidden variables and weight matrices are updated with linear complexity, which makes our algorithm time efficient. Then, we apply KNN method to forecast the hidden variables, and further forecast the speeds by multiplying weight matrices with hidden variables, i.e.,

$$\hat{z}\left(t+h_f\right)=f\left(z\left(t\right)\right)$$

$$x\left(t+h_f\right)=\hat{z}_{1?}\left(t+h_f\right)\mathbf{w}_1\left(t\right)+...+\hat{z}_k\left(t+h_f\right)\mathbf{w}_k\left(t\right)$$

where f ($t$) is the forecasting method for hidden variables z(t), and $x\left(t+h_f\right)$ is the forecasting speed at time $t$. We update the weight matrix W for each time step so that the current state can be kept current, contributing to more accurate forecasting .

We also try to find the best parameters for our forecasting algorithm. As discussed in the last section, $k$ = 1 is enough to give satisfactory results. As for the rest of the parameters, $h_p$ = 3, $knn$ = 5 can provide robust results.

# Chapter 4 Conclusion and Future Work

## 4.1 Conclusion

### 4.1.1 Pattern discovery for traffic flow Data

Two techniques are used to determine *k* critical patterns (or hidden variables) for a large number *n* road segments. These patterns can be used to reconstruct the traffic data for all the segments. One is based on the conventional PCA method and the other is based on the online PASTd algorithm. The reconstruction results for both methods show that:

• Reduce the dimensionality of the correlated road segments from *n* to a much smaller number *k*.

• Adaptively change values of parameters to improve performance.

• Update the hidden variables and weight matrix efficiently. For PCA, runtime is around 2 seconds while for PASTd runtime is around 1 second for a whole week data samples of 48 road segments.

We also compared the performance between PCA and PASTd. The comparison shows that PASTd is more suitable for finding patterns in terms of accuracy and time efficiency. The reason is mainly because it requires low computation - $O(kn)$ floating point operations. We do not need to use the expensive SVD decomposition. Its space requrement is also low since it does not need to buffer any data except the mean. We can use PASTd for travel time prediction as follows. We can choose any forecasting model such as AR model to predict hidden variables, and use the previous weight vectors to predict the speed for the next time step, i.e.,

$$\hat{\mathbf{z}}(t+1) = f\left(\mathbf{z}(t)\right)$$

$$\mathbf{x}(t+1) = \hat{z}_1(t+1)\mathbf{w}_1(t) + \ldots + \hat{z}_k(t+1)\mathbf{w}_k(t)$$

By using PASTd, we can save significantly in computational time as well as energy for prediction. When n is large, the advantage of PASTd will be more significant.

### 4.1.2 Short-term forecasting for traffic flow data

The vehicle speeds have been forecasted using a combination of the PASTd algorithm and the KNN method. First, the PASTd method reduces the data dimension from a large number *n* to a much smaller *k* (in our test, *n* = 1751, *k* = 1). We can adaptively change values of parameters in PASTd algorithm to improve performance. Hidden variables and weight matrices are updated in linear complexity, which makes our algorithm time efficient. Then, we apply KNN method to forecast the hidden variables, and further forecast the speeds by multiplying weight matrices with hidden variables, i.e.,

$$\hat{z}(t+h_f) = f(z(t))$$

$$x(t+h_f) = \hat{z}_{1?}(t+h_f)\mathbf{w}_1(t) + \ldots + \hat{z}_k(t+h_f)\mathbf{w}_k(t)$$

where f (*t*) is the forecasting method for hidden variables z(t), and $x(t+h_f)$ is the forecasting speed at time *t*. We update the weight matrix W for each time step so that the current state can be kept updated, contributing to more accurate forecasting.

We also try to find the best parameters for our forecasting algorithm. As discussed in the last section, *k* = 1 is enough to give the satisfactory results. $h_p$= 3, *knn* = 5 can provide robust results.

## 4.2 Suggestions for Future Work

In the future, our work will focus on the following aspects:

• Large scale. In this thesis, we test the Baltimore County, which has 1751 road segments. Our next step is to forecast the speeds for the whole Maryland, involving tens of thousands road segments.

• Spatial correlation. The road segments we test are close to each other. This is why we can only use one hidden variable to forecast the whole road segments. However, we also need to explore the scattered road segments and see how to reduce the data dimensionality efficiently.

• Improved KNN. There is an improved KNN method has been proposed, which takes advantage of temporal-spatial correlation. We will try to apply it and if its performance can beat the original KNN.

• Performance measurement. In our experiment, we use MSE as our performance measurement. There are many other standards as well. We will try them to further validate our algorithm.

• Travel time prediction. Our method can forecast the speeds for the next hour ($h_f$ = 60). Thus, we can use our algorithm to find the best way from the starting point to the destination. This application is very important to our daily life.

※ In our study, The traffic flow data used to do our research work are from Maryland State. It is collected by the Vehicle Probe Project(VPP) started in July 2008. Here, we express our sincere thanks to VPP . A brief introduction of VPP is given in an appendix.

# Appendix: The VPP Data Set

The Vehicle Probe Project started in July 2008 and its original goal was to enable a wide-variety of transportation operations and planning applications that require a high-quality data source. VPP data are generated by Global Positioning System (GPS) devices in vehicles. Multiple readings on a segment during a reporting period are aggregated to compute average travel speeds for that reporting period.

The VPP, archived in the VPP Suite maintained by CATT Laboratory at University of Maryland, College Park, is facilitated by the I-95 Corridor Coalition and is a continuous feed of probe-based traffic data acquired from the private sector through contracts awarded since January 2008. The VPP contractually reports traffic conditions on over 7,000 miles of freeways and 32,000 miles of arteries.

The data file downloaded from VPP Suite contains three speed values: current, average or historic and reference. The current speed is the speed observed on the given segment of road. The average or historic speed is an average of all the readings captured at the reporting instance over the past several weeks.

A widely adopted road segmentation scheme called Traffic Message Channels (TMC). TMC divides roads from one break in access to another, and uses a unique code for each segment. All VPP data files are accompanied by a TMC identification file, which contains the location information of the TMC codes. Probe data is generally aggregated to TMC codes, and reported against a TMC code for reporting periods.

# Glossary

If needed.

# Bibliography

[1]. Liao T W. Clustering of time series data—A survey. Pattern Recognition, 2005, 38(11): 1857-1874.

[2]. William B M. Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process. [Dissertation]. University of Virginia, Charlottesville, USA, 1999.

[3]. Zhang H S, Zhang Y,LI ZH, et al. Spatial-temporal traffic data analysis based on global data management using MAS. IEEE Transaction on Intelligent Transportation Sys., 2004, 5(4)：268-275.

[4]. Kohonen T. Self-Organizing Maps. Berlin Heidelberg: Springer-Verlag, 1995.

[5]. Kohonen T. Self-Organizing Maps. 2nd ed. Berlin Heidelberg: Springer-Verlag, 1997.

[6] Y. Chen, Y. Zhang, and J. Hu, "Multidimensional traffic flow time series analysis with self-organizing maps," Tsinghua Science & Technology,vol.13, no.2, pp.220–228, 2008.

[7]. Ahmed, M.S., Cook, A.R., 1979. Analysis of freeway traffic time-series data by using Box-Jenkins techniques. Transportation Research Record 722, 1–9.

[8]. Wang, Y., Papageorgiou, M., 2005. Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. Transportation Research Part B: Methodological 39 (2), 141–167.

[9]. Yuan Y., Van Lint, J.W.C., Wilson, R.E., van Wageningen-Kessels, F., Hoogendoorn, S.P., 2012. Real-time Lagrangian traffic state estimator for freeways.

IEEE Transactions on Intelligent Transportation Systems 13 (1), 59–70.

[10]. Treiber, M., Kesting, A., 2012. Validation of traffic flow models with respect to the spatiotemporal evolution of congested traffic patterns. Transportation Research Part C: Emerging Technologies 21 (1), 31–41.

[11]. Fowe, A. J., Chan, Y., 2013. A microstate spatial-inference model for network-traffic estimation. Transportation Research Part C: Emerging Technologies 36, 245–260.

[12]. Kerner, B.S., Klenov, S.L., Hermanns, G., Schreckenberg, M., 2013. Effect of driver over-acceleration on traffic breakdown in three-phase cellular automaton traffic flow models. Physica A: Statistical Mechanics and its Applications 392 (18), 4083–4105.

[13]. Karlaftis, M.G., Vlahogianni, E.I., 2011. Statistics versus neural networks in transportation research: differences, similarities and some insights. Transportation Research Part C Emerging Technologies 19 (3), 387–399.

[14]. S. Papadimitriou, J. Sun, and C. Faloutsos, “Streaming pattern discovery in multiple time-series,” in Proceedings of the 31st international conference on Very large data bases, pp. 697–708, VLDB Endowment, 2005.

[15]. Y. Chen, Y. Zhang, and J. Hu, “Multidimensional traffic flow time series analysis with self-organizing maps,” Tsinghua Science&Technology,vol.13, no.2, pp.220– 228, 2008.

[16]. D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” the Journal of machine Learning research, vol. 3, pp. 993– 1022, 2003.

[17]. X. Wei, J. Sun, and X. Wang, “Dynamic mixture models for multiple time-series.,” in IJCAI, vol. 7, pp. 2909–2914, 2007.

[18]. P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun, “A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting,” Transportation Research Part C: Emerging Technologies, vol. 62, pp. 21–34, 2016.

[19]. B. M. Williams and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results,” Journal of transportation engineering, vol. 129, no. 6, pp. 664–672, 2003.

[20]. S. Lee and D. Fambro, “Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting,” Transportation Research Record: Journal of the Transportation Research Board, no. 1678, pp. 179–188, 1999.

[21]. J. Wang, W. Deng, and Y. Guo, “New bayesian combination method for short-term traffic flow forecasting,” Transportation Research Part C: Emerging Technologies, vol. 43, pp. 79–94, 2014.

[22]. J. Guo, W. Huang, and B. M. Williams, “Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification,” Transportation Research Part C: Emerging Technologies, vol. 43, pp. 50–64, 2014.

[23]. Y. Hu, C. Wu, and H. Liu, “Prediction of passenger flow on the highway based on the least square support vector machine,” Transport, vol. 26, no. 2, pp. 197–203, 2011.

[24]. X. Jiang and H. Adeli, “Dynamic wavelet neural network model for traffic flow forecasting,” Journal of transportation engineering, vol. 131, no. 10, pp. 771–779, 2005.

[25]. B. L. Smith and M. J. Demetsky, “Short-term traffic flow prediction: neural net work approach,” Transportation Research Record, no. 1453, 1994.

[26]. S. Sun, R. Huang, and Y. Gao, “Network scale traffic modeling and forecasting with graphical lasso and neural networks,” Journal of Transportation Engineering, vol. 138, no. 11, pp. 1358–1367, 2012.

[27]. A. Stathopoulos, L. Dimitriou, and T. Tsekeris, “Fuzzy modeling approach for combined forecasting of urban traffic flow,” Computer-Aided Civil and Infrastructure Engineering, vol. 23, no. 7, pp. 521–535, 2008.

[28] M. Van Der Voort, M. Dougherty, and S. Watson, “Combining kohonen maps with arima time series models to forecast traffic flow,” Transportation Research Part C: EmergingTechnologies,vol.4,no.5,pp.307– 318, 1996.

[29] I. Okutani and Y. J. Stephanedes, “Dynamic prediction of traffic volume through kalman filtering theory,” Transportation Research Part B: Methodological, vol. 18, no. 1, pp. 1–11, 1984.

[30] S. Papadimitriou, J. Sun, and C. Faloutsos, “Streaming pattern discovery in multiple time-series,” in Proceedings of the 31st international conference on Very large data bases, pp. 697–708, VLDB Endowment, 2005.

[31] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams,” in Proceedings of the 29th international conference on Very large data bases-Volume 29, pp. 81–92, VLDB Endowment, 2003.

[32] Y. Sakurai, S. Papadimitriou, and C. Faloutsos, “Braid: Stream mining through group lag correlations,” in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 599–610, ACM, 2005.

[33] Y. Zhu and D. Shasha, “Statstream: Statistical monitoring of thousands of data streams in real time,” in Proceedings of the 28th international conference on Very Large Data Bases, pp. 358–369, VLDB Endowment, 2002.

[34] B. Yang, “Projection approximation subspace tracking,” Signal Processing, IEEE Transactions on, vol. 43, no. 1, pp. 95–107, 1995.

[35] J. Van Lint and C. Van Hinsbergen, “Short-term traffic and travel time prediction models,” Artificial Intelligence Applications to Critical Transportation Issues, vol. 22, pp. 22–41, 2012.

[36] B. L. Smith, “Forecasting freeway traffic flow for intelligent transportation systems application.,” Transportation Research Part A, vol. 1, no. 31, p. 61, 1997.

[37] E. I. Vlahogianni, M. G. Karlaftis, and J.C.Golias, “Short-term traffic forecasting: Where we are and where we are going,” Transportation Research Part C: Emerging Technologies, vol. 43, pp. 3–19, 2014.

[38] B. Yang, “Projection approximation subspace tracking,” IEEE Transactions on Signal processing, vol. 43, no. 1, pp. 95–107, 1995.