A METHOD FOR USING ACTIVITY RECOGNITION TO IMPROVE ENSEMBLE FORECASTING FOR TRAFFIC SYSTEMS

by

James Howard

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Computer Science).

Golden, Colorado

Date _____

Signed: _____
James Howard

Signed: _____
Dr. William Hoff
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Big Boss
Professor and Head
Department of Electrical Engineering and Computer Science

ABSTRACT

Forecasting the occupancy of buildings can lead to significant improvement of smart heating and cooling systems. Using a sensor network of simple passive infrared motion sensors densely placed throughout a building, we perform data mining to forecast occupancy a short time (*i.e.*, up to 60 minutes) into the future. Our approach is to train a set of standard forecasting models to our time series data. Each model then forecasts occupancy a various horizons into the future. We combine these forecasts using a modified Bayesian combined forecasting approach. The method is demonstrated on two large building occupancy datasets, and shows promising results for forecasting horizons of up to 60 minutes. Because the two datasets have such different occupancy profiles, we compare our algorithms on each dataset to evaluate the performance of the forecasting algorithm for the different conditions.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES AND TABLES vii

LIST OF ABBREVIATIONS xii

ACKNOWLEDGMENTS xiii

DEDICATION xiv

CHAPTER 1 INTRODUCTION 1

1.1 Problem and objective 3

1.1.1 A motivating example 3

1.2 Approach 5

1.3 Contributions 6

1.4 Structure of the thesis 7

CHAPTER 2 MODELS 8

2.1 Notation 8

2.2 Forecast error metrics 9

2.3 Seasonal ARIMA model 12

2.3.1 Fitting a seasonal ARIMA 13

2.4 Historic average 19

2.5 Time delayed neural networks 19

2.6 Support vector regression 21

CHAPTER 3 DATASETS 23

3.1 Building datasets . . . 23
3.1.1 MERL dataset . . . 25
3.1.2 Colorado School of Mines dataset . . . 27
3.2 Traffic datasets . . . 29
3.2.1 Denver traffic dataset . . . 30
3.3 Additional notes on the datasets . . . 32
CHAPTER 4 A ENSEMBLE FORECASTER FOR LONG TERM FORECASTING . 34
4.1 Why ensemble forecasting? . . . 34
4.2 Bayesian combined forecasting . . . 35
4.3 BCF modifications . . . 37
4.3.1 Forecast δ time steps into the future . . . 37
4.3.2 Improving model error distributions . . . 38
4.3.3 Model selection thresholding . . . 39
4.4 Results of BCF and BCF-TS . . . 41
CHAPTER 5 ACTIVITY RECOGNITION ENSEMBLE FORECASTING . . . 46
5.1 The need for another ensemble forecaster . . . 46
5.2 Bayesian combined forecasting with activity and anomaly modeling (ABCF) . 48
5.3 Background literature on activity recognition and anomaly detection . . . 49
5.4 Anomaly extraction and representation . . . 54
5.5 Time series mixture of Gaussians derivation . . . 56
5.6 Selecting the number of clusters . . . 60
5.7 Sample representative clusters and a discussion about them . . . 62
5.8 ABCF derived . . . 65

5.9 Demonstration of ABCF 66

5.10 Results of ABCF 69

5.11 Future unexplored work on ABCF 74

CHAPTER 6 CONCLUSION 78

REFERENCES CITED 79

APPENDIX A - OTHER RESULTS 87

APPENDIX B - FIGURE REFERENCES 95

LIST OF FIGURES AND TABLES

Figure 1.1 Total number of cars passing major highway sensors in Denver on Sundays in September and October 2010 4

Figure 1.2 High level overview of our approach 6

Figure 2.1 Demonstration of the area measured by RMSE-ONAN. This data is two days worth of residual data from a Seasonal ARIMA model on Denver traffic data. The solid green areas are measured and squared towards RMSE-ONAN. All other areas will be given a value of zero. These solid green regions correspond with forecasting errors outside a prescribed boundary which in this case is one standard deviation from the mean for that given period of time. 10

Figure 2.2 Raw counts of Denver data for one sensor over a two week period. 14

Figure 2.3 Denver data: one week seasonal difference counts and autocorrelation over a two week period. 16

Figure 2.4 Denver data: Residual autocorrelation and partial autocorrelation results from fit seasonal ARIMA model. 17

Figure 2.5 One-step ahead prediction for a sample week. Black line is original data. Red line is forecasted data. Dotted box shows an example of mis-forecasting due to a Broncos game. 18

Figure 2.6 Architecture of a time delayed neural network with $m + 1$ inputs and J outputs 20

Figure 3.1 Passive infrared motion detector 25

Figure 3.2 Floor plan and sensor locations for the MERL office building dataset. The red rectangle in (a) and (b) is the location of our sensor for this work. 26

Figure 3.3 Scaled average readings for a given day and scaled readings for two days from the MERL dataset. 26

Figure 3.4 Sensor locations for the Colorado School of Mines Brown Building. The red rectangle is the location of the sensor used for our analysis. . . . 27

Figure 3.5 Sensor readings for two consecutive days from the Brown Dataset. 28

Figure 3.6 Average readings from Wednesdays and Thursdays for a given sensor from the CSMBB dataset. 29

Figure 3.7 City of Denver traffic sensors. Data analyzed for this work is from the sensor in red. 30

Figure 3.8 Average readings from Sunday and Monday for a given sensor from the Denver traffic dataset. 31

Figure 3.9 Scaled average readings for a given day and scaled readings for two days from the Denver traffic dataset. 32

Figure 4.1 Normalized posterior probabilities of component models on a section of MERL dataset. 36

Figure 4.2 Standard deviation of support vector machine residuals for all Wednesdays in MERL dataset. Time index represents 10 minute intervals from 6:00am to 7:00pm. 38

Figure 4.3 A comparison of forecasts at various horizons against real data for an sample time segment using BCF-TS. 41

Figure 4.4 A comparison of BCF-TS and SVM forecasts at horizon equal to three against real data. 42

Figure 4.5 Merl dataset: RMSE of forecasting for each model vs forecasting horizon. 43

Figure 4.6 Denver dataset: RMSE of forecasting for each model vs forecasting horizon. 44

Figure 4.7 Brown dataset: RMSE of forecasting for each model vs forecasting horizon. 45

Figure 4.8 Improvement percentage of RMSE values of BCF-TS compared to BCF. . 45

Figure 5.1 Two similar events occurring at different time in the same Denver residual dataset. 46

Figure 5.2 Scaled histogram of IBCF residual values for the Denver traffic dataset at various daily time steps. The red line is the corresponding best fit Normal distribution. Notice how in all plots there exist data points which exist after the tails of the Normal distributions have approached zero. 47

Figure 5.3 Extremely high level overview of our approach 48

Figure 5.4 Demonstration of a fixed width sliding window looking for locally maximal deviations from background behavior. 54

Figure 5.5 Extracted residuals from the MERL dataset using the sliding window extraction method with window length of 7 56

Figure 5.6 Demonstration of the elbow clustering selection [ref]. 60

Figure 5.7 Extracted residuals from the MERL dataset. Data was taken from the top 10% of the SVM forecaster's residuals. The red line in the center of the clusters represents the cluster average. 63

Figure 5.8 Extracted residuals from the Brown dataset. Data was taken from the top 10% of the ARIMA forecaster's residuals. The red line in the center of the clusters represents the cluster average. 63

Figure 5.9 Extracted residuals from the Denver dataset. Data was taken from the top 10% of the TDNN forecaster's residuals. 64

Figure 5.10 Sample residual and the respective probabilities of each time step for the base forecaster and the a single activity model. The teal line represents the likelihood of the background ARIMA model. The tan lines correspond to the likelihood each time step from one of the anomaly clusters. 67

Figure 5.11 Extracted sample clustered anomaly. 68

Figure 5.12 Two horizon time step results when applying ABCF to an ARIMA forecaster for a segment of the Denver dataset. 68

Figure 5.13 RMSE-ONAN vs forecasting horizon for Average model and Average + ABCF on the Denver dataset 69

Figure 5.14 RMSE-ONAN vs forecasting horizon for ARIMA model and ARIMA + ABCF on the MERL dataset 70

Figure 5.15 RMSE vs forecasting horizon for ARIMA model and ARIMA + ABCF on the MERL dataset 71

Figure 5.16 71

Figure 5.17 72

Figure 5.18 72

Figure 5.19 74

Figure 5.20 74

Figure 5.21 75

Figure 5.22 75

Figure 5.23 76

Figure 5.24 76

Figure A.1 Brown dataset: Root mean square error of forecasting for each model vs forecasting horizon. 87

Figure A.2 Merl dataset: root mean square error of forecasting for each model vs forecasting horizon. 88

Figure A.3 Denver dataset: root mean square error of forecasting for each model vs forecasting horizon. 88

Figure A.4 Improvement percentage of MASE values of BCF-TS compared to BCF. . 89

Figure A.5 Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm 89

Figure A.6 Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm 90

Figure A.7 Results for the four base forecasting algorithms for the Denver dataset and the improvements to SQEONAN from using our ABCF algorithm . . 90

Figure A.8 Results for the four base forecasting algorithms for the MERL dataset and the improvements to RMSE from using our ABCF algorithm 91

Figure A.9 Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to RMSE from using our ABCF algorithm 91

Figure A.10 Results for the four base forecasting algorithms for the Denver dataset and the improvements to RMSE from using our ABCF algorithm 92

Figure A.11 Results for the four base forecasting algorithms for the MERL dataset and the improvements to MASE from using our ABCF algorithm 92

Figure A.12 Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to MASE from using our ABCF algorithm 93

Figure A.13 Results for the four base forecasting algorithms for the Denver dataset and the improvements to MASE from using our ABCF algorithm 93

Figure A.14 Percentage improvement to RMSE due to application of ABCF 94

Table 2.1 The parameter values that were fit for MERL and CSMBB datasets for a Seasonal ARIMA model 18

Table 2.2 Number of delayed input nodes and hidden nodes for MERL and CSMBB datasets 21

LIST OF ABBREVIATIONS

1
for candidate data extraction55

Anomaly bayesian combined forecaster . . . . . ABCF

Bayesian combined forecaster . . . . . BCF

Seasonal auto regressive moving average . . . . . SARIMA

Support vector machine . . . . . SVM

Time delayed neural network . . . . . TDNN

ACKNOWLEDGMENTS

That you everyone

For those that shall follow after.

TODO Fill in a dedication

CHAPTER 1
INTRODUCTION

According to the U.S. Department of Energy [1], energy for heating and cooling accounts for approximately 35 - 45% of the total expenditure within a building. With such a large investment of energy being used to regulate the temperature of a building, possible areas of improvement are heavily sought after. Fully automated buildings with control systems to automatically heat and cool individual rooms or spaces have been designed [2, 3] to reduce building energy usage while maintaining proper temperatures throughout the building. These systems are complex and require knowledge of room usage, and in more complex systems room occupancy, as inputs into system control models.

While many factors such as ambient temperature, ventilation air flow, room volume, *etc.* affect the time it takes to heat or air condition a room, it still takes on the order of many minutes to bring a room to a stable desired temperature [4]. Due to this lag in changing the temperature of a room, it is not sufficient for control systems to begin air conditioning a room upon initial occupancy. Thus, to ensure that room temperatures are appropriate, smart building control systems typically rely on set schedules of occupancy. These systems may use scheduled occupancy up to 24 hours into the future to determine current heating and air conditioning control [5].

However, what happens to the system when building traffic deviates from its schedule? Let us consider a university classroom building. While not often, professors will occasionally cancel class. What should a smart control system do during this time? It does not make sense for the system to heat or cool the room as though it was occupied. The system should adapt to the changing environment. Similarly, what happens if snow has caused many of the students and faculty to stay at home? In this case, a cancelled class in the morning is likely correlated to class cancellations later in the day. Ideally the building control system

should identify situations where there is a significant deviation in the number of occupants within any part of the building and modify its heating or cooling schedule to account for such situations. In both of these scenarios, a set schedule is insufficient to produce an optimal heating or cooling schedule for the building.

As another example of the usage of complex control systems, consider the roadways of the United States. Optimal timing of traffic lights on major roadways across the United States could result in approximately a 22% reduction in emissions along with a 10% reduction in fuel consumption [6]. As of 2005 the total estimated fuel savings would amount to approximately 17 billions gallons of motor fuels annually. This traffic light timing does not only consider city lights, but also takes into account freeway onramp volume lights.

In the United States, traffic light timings are often determined by an individual from the local department of transportation standing near the light and manually determining a timing schedule, or in some cases multiple schedules to account for peak traffic times and non-peak times [7]. These schedules are then fixed and are changed either when roadways change to make new timings necessary or if petitioned by local citizens. These timings are then either set in local control box for that traffic light or by a central control system for the city.

As with the building scenario, what happens when the traffic deviates from normal? Fixed timings will not be able to account for changes in traffic. Inclement weather scenarios should likely require different timings then sunny days. Lights near large sporting events likely require different timings during those events than typical evenings. Even if schedules were made for such scenarios, there certainly exist scenarios for which schedules can not be made manually such as lane blockages due to accidents.

In both of the above environments, the control systems have to account for future occupancy of the environment. It is inadequate for these systems to use only current data to control the system. Accurate forecasts of the systems usage are necessary to produce optimal control systems.

1.1 Problem and objective

As alluded to with the above examples, our focus is on improving forecasting accuracy on human controlled traffic systems. By traffic, we mean the movement of vehicles along a road network, the movement of people in a building, or similar data derived from the actions of a group of people. Overall, our method is more general than just traffic systems and we believe it will work with any dataset which has rare repeated events that lead to similar effects in the system. For the rest of this paper, we define a time series dataset as $\{x_t^{(m)}\}$. Each x_t^m is an aggregate of the readings from sensor m reading at time block t. Our objective in this work is then:

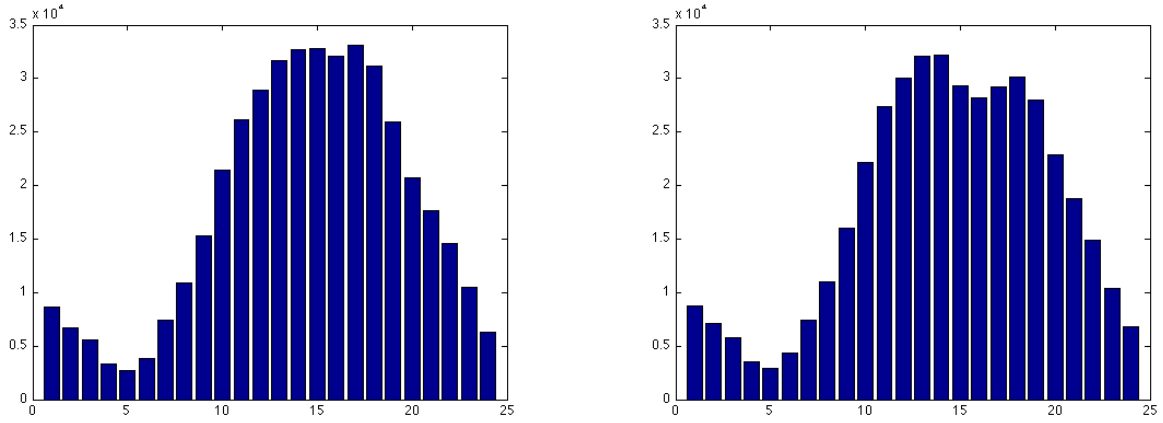
1. Produce an accurate forecast for $x_t^{(m)}$ δ time steps into the future.
2. Reduce worst case forecasts.
3. Keep approach unsupervised.

Because the literature on forecasting is so vast, we specifically focus only on human traffic systems. As we will demonstrate shortly, our approach will make some assumptions based on the type of data produced by these traffic systems which will As a result, we will demonstrate considerably better results that would likely be possible with data from other sources. Also, we will demonstrate our results empirically. Due our lack of theoretical constraints, our improved results may simply be a by product of our datasets and and not something that may be applicable to other human traffic systems. Of course, we do not believe this to be the case due to the improvement in vastly different datasets which we will demonstrate later.

1.1.1 A motivating example

To illustrate an example of the need to minimize worst case forecasts, we present the following example. The Denver Broncos, as with most American Football teams are a significant local attraction. In 2010 the team had an average attendance of 74,908 [8]. This

attendance, combined with additional fans flooding downtown to patronize bars and restaurants creates an interesting effect on freeway traffic patterns. Unsurprisingly, prior to the game there is an increase in total traffic volume along a freeway which is near the Broncos' Stadium. Also, there exists a nearly 20% drop in total traffic volume along the same stretch of freeway during the game. Figure 1.1 shows the total counts of Denver traffic by a single loop detector on a stretch of highway near the station for each hour of the day averaged for the first four Sunday Broncos home games and for the first four Sunday away games in 2010. Comparing Figure 1.1(a) with figure Figure 1.1(b) it is evident that a noticeable change in traffic patterns occur from approximately noon until approximately 6:00 pm. This traffic change corresponds with a 2:05pm kickoff time for the game.



(a) (b)

Figure 1.1: Total number of cars passing major highway sensors in Denver on Sundays in September and October 2010

Such a significant difference in traffic patterns should lead to a change in traffic light control. We have all likely encountered the frustrating scenario of attempting to leave one of these sporting events. Traffic lights are often still on predefined timings and the situation arises where a large number of vehicles attempt to get through a traffic light controlled intersection in one direction with almost no vehicles attempting to get through the intersection in the perpendicular direction. Ideally the light timings should be changed to fix this scenario

and increase the green light time in the direction with multiple cars. Of course, optimally other light timings will also need to be changed to account for the increase volume of traffic along certain paths within the city.

Traditional parametric forecasting models have difficulty accounting for these different traffic patterns and the problem becomes more difficult when when it is considered that the Broncos may play a Sunday night game or a Monday night game. Thus our another goal of our approach is to handle these significant deviations from normal traffic patterns which are typically the causes of worse case forecasting scenarios.

1.2 Approach

To satisfy our first objective of producing an accurate forecast for dataset $x_t^{(m)}$ δ time steps into the future, we have created an ensemble forecasting model based on the Bayesian combined forecaster (BCF) created by Petridis [9]. Chapter 4 discusses modifications we have made to BCF which improve its performance on human controlled traffic systems.

To satisfy our second objective of reducing worst case forecasts, i.e. the sporting event scenario, we introduce a novel forecasting technique based on anomaly detection and modeling. Empirically we have found that for human controlled traffic systems large deviations in forecasting accuracy often coincided with human controlled activities. For the purposes of this paper, we frequently interchange the terms activities and anomalies. The reason for this is as a matter of perspective. To a forecasting model, an infrequently occurring human activity will likely produce anomalous results. We describe any event which causes large prediction errors for our forecasting system as either an anomaly or an activity. For vehicle traffic systems these activities might be sporting events, road closures, or accidents. In the case of building models, such activities might be large infrequent meetings, cancelled classes, or early closure due to holidays.

Because such activities can overlap or occur at different times with varying amount of background noise present, it is a difficult task for one parametric model to accurately encapsulate all of the activities which occur within the system. Our approach is to split

the problem of forecasting into two parts: development of a background model and the development of a set of activity models. Our background model is represented by any forecasting model. In practice we have computed our results using traditional forecasting models such as Time Delayed Neural Networks, Auto Regressive Models or Support Vector Machines.

To model activities, we propose comparing different models from the activity recognition literature along with a new model which we propose here. Forecasting is then performed using an ensemble predictor defined in Chapter 5, taking outputs from all trained activity models and the background model. Figure 1.2 shows the general structure of our approach. We combine classic forecasting models with activity models in a novel ensemble model to improve overall forecasting accuracy.

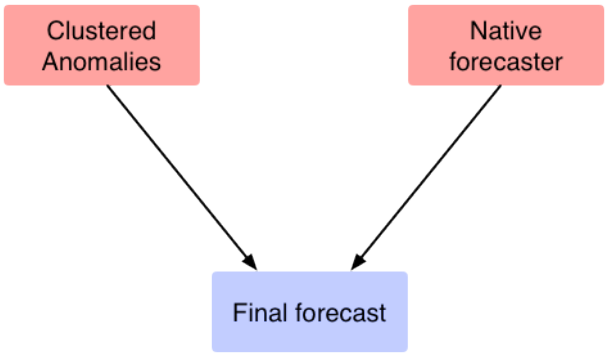


Figure 1.2: High level overview of our approach

Finally, in satisfying our third objective, we keep everything unsupervised. Nothing in our approach prevents the use of supervised modeling of activities and incorporating a trained activity model into our final forecast, but such supervised information is not necessary to achieve improved forecasting results.

1.3 Contributions

The contributions to the field of unsupervised traffic forecasting from this work are:

- Extraction and modeling of forecasting activity models using residual errors from another forecasting model.
- An improvement to a combined Bayesian prediction model to improve forecasting accuracy
- A combined prediction model using an ensemble forecaster and activity models to improve short term forecasts during the presence of anomalous activities

1.4 Structure of the thesis

The remainder of this thesis is outlined as follows: Chapter 2 defines typical forecasting models which are to be used for comparison and as base native forecasters for our final ensemble forecaster. This chapter also introduces some notation used through this document. Chapter 3 discusses the datasets used therein along with specifics details on any necessary changes to make the datasets compatible with our forecasters. Chapter 4 discusses a classic ensemble forecaster which we apply to our datasets along with introducing some improvements on this forecaster which greatly improve its long term forecasting performance. Finally Chapter 5 discusses our novel approach using activity modeling to improve the short term accuracy of any of our base forecaster for our datasets. This section also includes a brief discussion of previous work in the realm of anomaly detection.

CHAPTER 2

MODELS

With this chapter we explore some of the most common forecasting models for traffic datasets. These models are meant to be used as a baseline from which to compare our results. In Chapter 4 the models presented here are incorporated into an ensemble forecasting model. The models selected for this work are:

- Seasonal Auto Regressive Integrated Moving Average (ARIMA)
- Historic Average
- Time Delayed Neural Network (TDNN)
- Support Vector Regression (SVM from the well known Support Vector Machine)

Also in this chapter, we introduce the metrics which will be used to empirically demonstrate the various forecasting capabilities of these established models. Some of the discussions of models within this chapter make mention of our datasets. These datasets are described in detail in Chapter 3. We chose to describe the specifics of our forecasting models prior to our datasources because we believe this makes for a more coherent paper.

2.1 Notation

As already stated, we define a time series dataset used within as $\{x_t^{(m)}\}$. Each x_t^m is an aggregate of the readings from sensor m reading at time block t. In total the number of time blocks are represented by N.

Forecasts for a given model k from the set of all models K are represented by

$$y_{t+1}^{k,m} = f(x_t, ..., x_1; \theta_k). \tag{2.1}$$

Thus the forecast of x_{t+1} is a function of all past data and some trained parameterization θ_k for that model. For this work we forecast a model for each individual sensor and for convenience often drop the m and k from our forecast notation. Also, in this work we need to forecast more than one time step into the future. Future forecasts are performed through iterative one step ahead forecasts. An example of a forecast two time steps ahead of current time t is given by

$$y_{t+2} = f(y_{t+1}, x_t, ..., x_1; \theta_k). \tag{2.2}$$

Such a forecast is simply the forecast for one time step into the future but now with the forecasted value of y_{t+1} used as the most recent datapoint to forecast y_{t+2}. Forecasting in this nature allows for forecasts any number of time steps into the future.

Another useful time series used in this work is the residual dataset defined as

$$r_{t,\delta} = x_{t+\delta} - y_{t+\delta}. \tag{2.3}$$

This set of residuals is the difference between the raw data and a forecasting function operating δ time steps into the future.

2.2 Forecast error metrics

There are a multitude of various forecasting error measurements that have been used to assess forecast accuracy. Expert recommendation of which forecast error measurement to use for what types of data has changed over time. Armstrong [10, 11] found that root mean squared error (RMSE) was by far the most popular error measurement in 1981 among academics. However, by 1995 RMSE preference by academics had dropped by roughly half and mean absolute percentage error (MAPE) was the preferred error measurement.

For this work, to coincide with expert opinion on forecast error measurements, we use both mean absolute scale error (MASE) [12, 13] and RMSE as cost functions to compare our forecasting accuracy. RMSE was selected due to its simplicity and ubiquity within the forecasting community. MASE was selected for it being both a modern derivation of MAPE and for its accuracy in places where MAPE and RMSE does not provide adequate results.

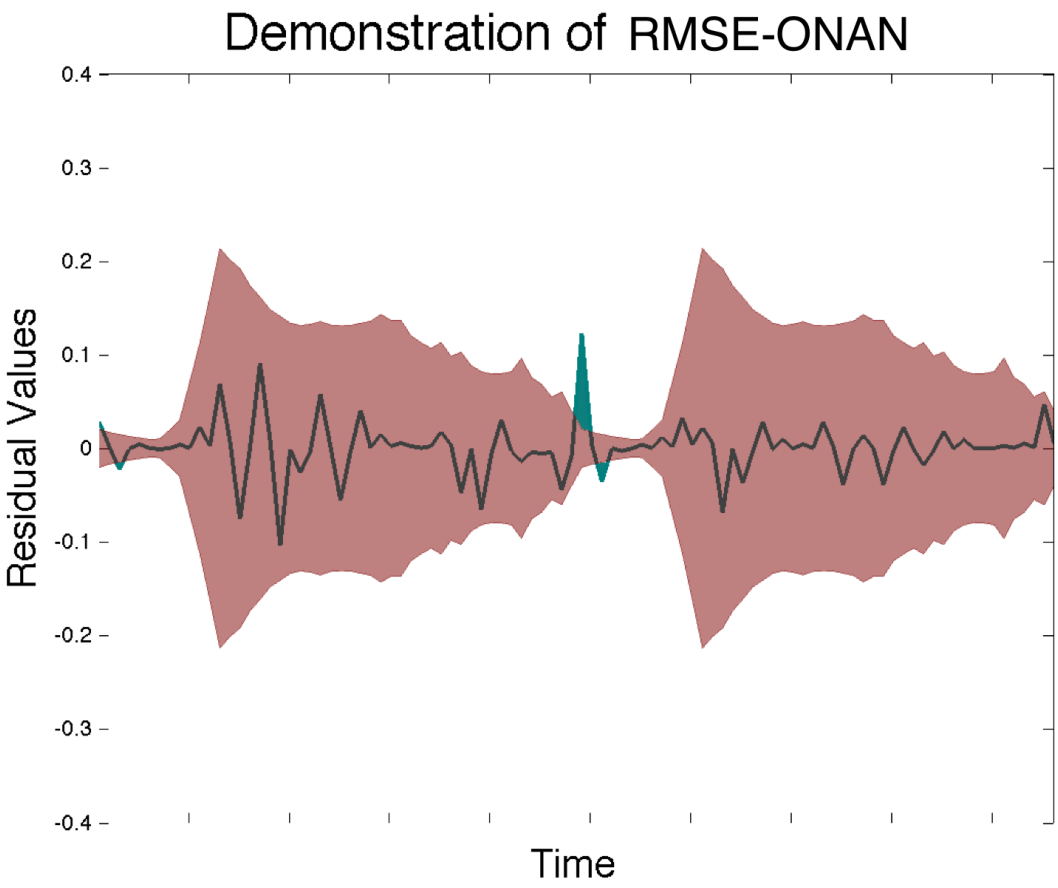


Figure 2.1: Demonstration of the area measured by RMSE-ONAN. This data is two days worth of residual data from a Seasonal ARIMA model on Denver traffic data. The solid green areas are measured and squared towards RMSE-ONAN. All other areas will be given a value of zero. These solid green regions correspond with forecasting errors outside a prescribed boundary which in this case is one standard deviation from the mean for that given period of time.

Also, because much of the focus of our work is interested in reducing inaccurate forecasts during the presence of some anomalous event, we introduce another forecasting measurement which we call "root mean squared error outside of noise against naive" (RMSE-ONAN). This metric provides a convenient way to measure the improvement our approach yields which is not easily captured by RMSE and MASE. The following is a detailed description of our metrics:

RMSE

RMSE is one of the most common error functions used to determine the quality of a set of forecasts. It measures the average difference between two time series. Due to its squared error term, this measurement is not a good indicator of error between distinctly different datasets. For our work comparing the input series x and the forecast series y RMSE performs adequately.

$$RMSE_\delta = \sqrt{\frac{\sum_{t=1}^{N-\delta} (x_{t+\delta} - y_{t+\delta})^2}{N - \delta}} \tag{2.4}$$

MASE

Mean absolute scaled error was developed to be a generally applicable measurement of forecast error. The metric is especially useful for datasets with intermittent demand unlike the commonly used MAPE score because MASE will never results in infinite or undefined values unless all historic data is equal (*i.e.* $\sum_{i=2}^{N} |x_i - x_{i-1}| = 0$)[14]. We thought this metric pertinent to include because both roadway and building traffic will likely have times with little to no traffic.

$$MASE_\delta = \frac{\sum_{t=1}^{N-\delta} r_{t,\delta}}{\frac{N}{N-1} \sum_{i=2}^{N} |x_i - x_{i-1}|} \tag{2.5}$$

RMSE-ONAN

Root mean squared error outside noise against naive measures a forecast's accuracy during its worst case scenarios. This is performed by measuring the sum of squared errors of forecasts outside a prescribed boundary. We compute this boundary from the forecasting accuracy of a naive forecaster. For our work, we consider the naive forecaster to be the historical average of the data for a given time.

Most of the results of this work deal with boundaries set by one standard deviation of the residual dataset formed by the historic average (naive) model. RMSE-ONAN is defined

as

$$RMSE - ONAN_{\delta,\sigma} = \frac{\sqrt{\sum_{t=1}^{N-\delta} A(r_{t,\delta}; \sigma_t)^2}}{N - \delta}. \tag{2.6}$$

We define $A(r_\delta; \sigma)$ as

$$A(r_{t,\delta}; \sigma_t) = \begin{cases} r_{t,\delta} - \sigma_t & \text{If } r_{t,\delta} \geq \sigma_t \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

σ_t is the standard deviation of the data at that time step for that given day. For certain comparisons, it is useful to use $n * \sigma$ with values of $n > 1$.

Notice that this function is a RMSE average of all values of $A(r_\delta)$ instead of simply the average RMSE of values where $A(r_\delta) > 0$. This is because RMSE-ONAN is used to compare between multiple forecasting algorithms of the same dataset the total effect of sufficiently inaccurate forecasts (by sufficiently inaccurate, we mean any forecast outside a defined accuracy boundary). If we were to measure only values where $A(r_\delta) > 0$, then an average value is not able to distinguish between forecasting models that leave few sufficiently inaccurate forecasts and many inaccurate forecasts assuming they are of similar value.

An example demonstrating the regions summed by RMSE-ONAN is given by Figure 2.1. In this image, only the area of the green spaces are squared and summed towards RMSE-ONAN. All other regions are zero. The large salmon colored region is the area that is one standard deviation for all days at that time for the residual dataset.

2.3 Seasonal ARIMA model

The Auto Regressive Moving Average Model (ARMA) or derivations on its form (Auto Regressive Integrated Moving Average, Seasonal Auto Regressive Moving Average, *etc*) have been used in numerous forecasting applications from economics to vehicle traffic systems. While we have been unable to find ARMA based models used on building occupancy data directly, we have found it used to forecast building energy usage and vehicle occupancy [15–19]. Due to ARIMA models having strong forecasting accuracy and frequent academic use, we believe it serves as an excellent baseline of comparison for a forecasting problem.

Our building and traffic data has periodic trends and a non stationary mean, thus from the class of ARIMA models, we believe a seasonal ARIMA model is best suited to fit our data. The seasonal ARIMA model as defined by Box and Jenkins [20] is:

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla_s^D T_t = \theta_q(B)\Theta_Q(B^s)e_t \tag{2.8}$$

where $\{T_t\}$ is our observed time series and $\{e_t\}$ represents an unobserved white noise series ($e_t \sim N(0, \sigma^2)$) the values of which are computed through model training and are not known a priori. B is the backshift operator which is a function that allows access to older time readings. For example $BT_t = T_{t-1}$ and $B^5T_t = T_{t-5}$. ∇_s^D is the seasonal difference operator ($\nabla_s^D T_t = (1 - B^s)^D T_t$) and $\phi,\ \Phi,\ \theta,\ \Theta$ are trainable parameters.

Seasonal ARIMA models are notated as

$$ARIMA(p, d, q)(P, D, Q)_s \tag{2.9}$$

where p is the number of autoregressive terms, d is the number of differences and q is the number of moving average terms. P, D, and Q all correspond to the seasonal equivalents of p, d, and q. The parameter s is the seasonality of the model. For a full discussion of seasonal ARIMA models see Box and Jenkins [20].

Forecasting from this model is performed by iteratively forward feeding values of the model into itself. Since the set of residuals e from a properly trained seasonal ARIMA model is described by a white noise Gaussian distribution $N(0, \sigma^2)$, we can take the expected value of the residual at time e_{t+1} to be 0. This leaves us with the following forecasting equation:

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla_s^D T_{t+1} = \theta_{q-1}(B)\Theta_{Q-1}(B^s)e_t \tag{2.10}$$

2.3.1 Fitting a seasonal ARIMA

Finding the correct values of p, d, q, P, D, Q, s is traditionally a hard problem. To fit our parameters we use a method recommended by Robert Nau from his lecture notes on optimally fitting ARIMA models [21]. Nau first recommends estimating the order of differencing on the data to achieve stationarity. After estimating the proper order of differencing, the

autoregressive and moving average terms can be identified through visual inspection of the autocorrelation and partial autocorrelation plots.

Achieving stationarity

Accurately identifying the required order of differencing will lead to a stationary dataset and thus one which can be forecasted by an ARIMA model. For a time series to be weakly stationary two conditions must be satisfied: The expected value of $x^{(t)}$ is the same for all t and the covariance between any two observations depends only on the lag.

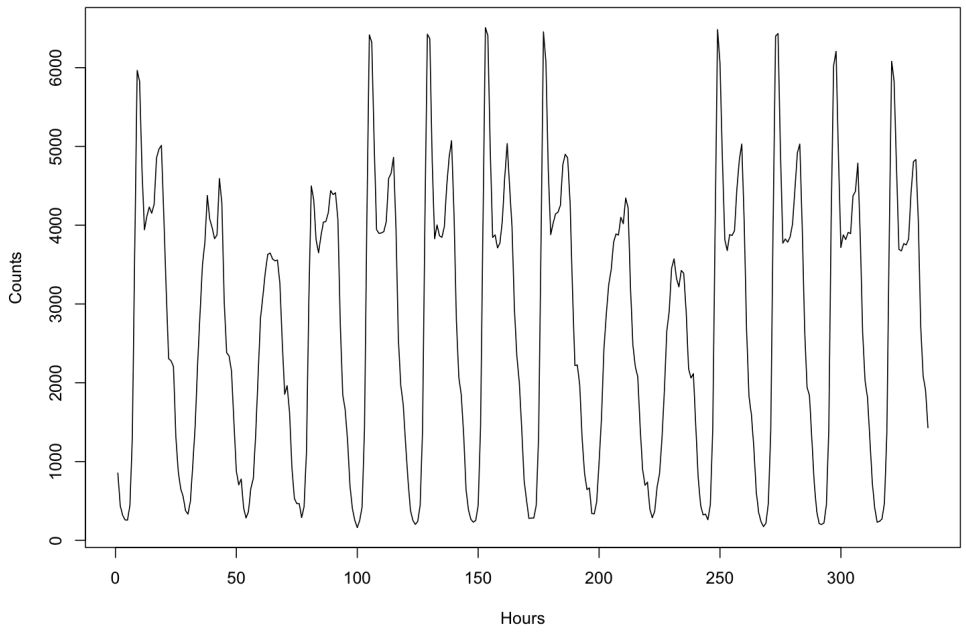


Figure 2.2: Raw counts of Denver data for one sensor over a two week period.

In general it is difficult to prove stationarity, but there exists a number of methods which assist in determining if a time series is close enough to stationary to be modeled by an ARMA model. Visual inspection of both the raw data and the autocorrelation function is a useful tool to test for stationarity. Figure 2.2 shows the raw counts values at hourly lags of vehicle counts for one sensor over a two week period from the Denver dataset. The data shows no constant mean and thus can not be stationary. The graph of autocorrelation values shows

local peaks every 24 hours with a significant peak at one week of lag (168 hours).

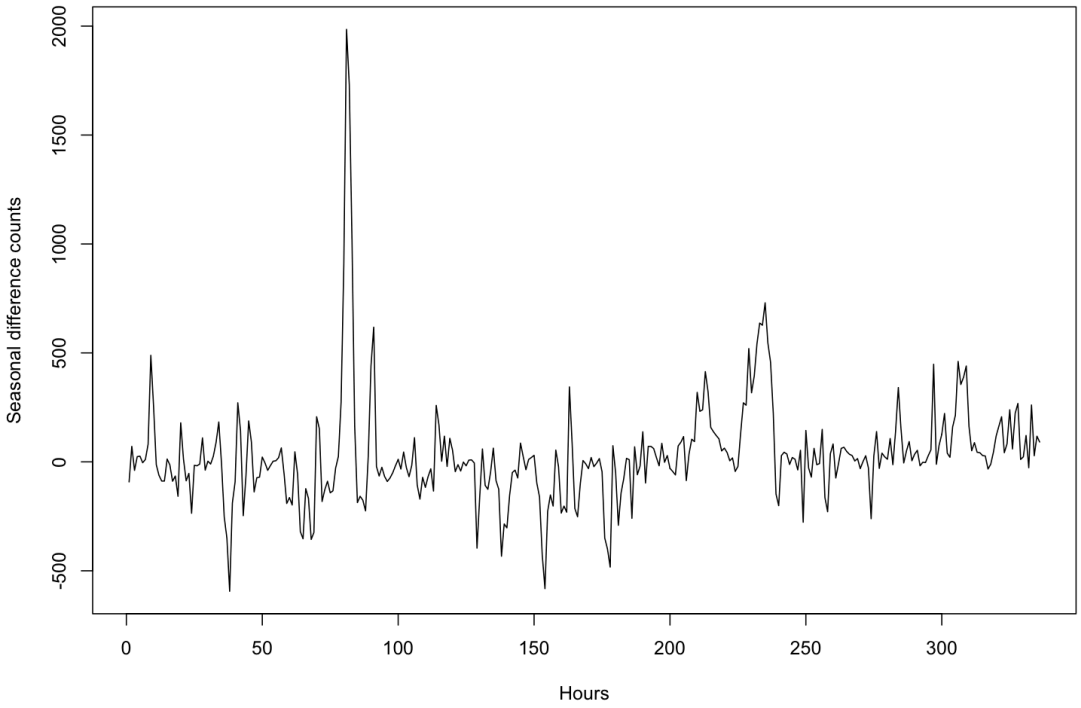
Intuitively a one week seasonal difference should yield a stationary time series and visually, outside of an anomalous reading, Figure 2.3 shows such stationarity. Applying the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) [22] test for stationarity on the seasonally differenced data confirms the visual inspection. Using R's implementation of KPSS gives a p-value greater than 0.1. This is significantly higher than the standard value to reject the stationarity hypothesis of 0.05.

As a verification of our model, we applied the LJung-Box test [23] on our set of residual data for each model. This tests if any of the auto correlation values on the residual dataset are significantly different from 0. To be valid, the LJung-Box test should return a value of $p > 0.05$. All of our residual sets passed: $p = 0.9964$ for MERL and $p = 0.1072$ for CSMBB and $p = 0.1266$ for Denver.

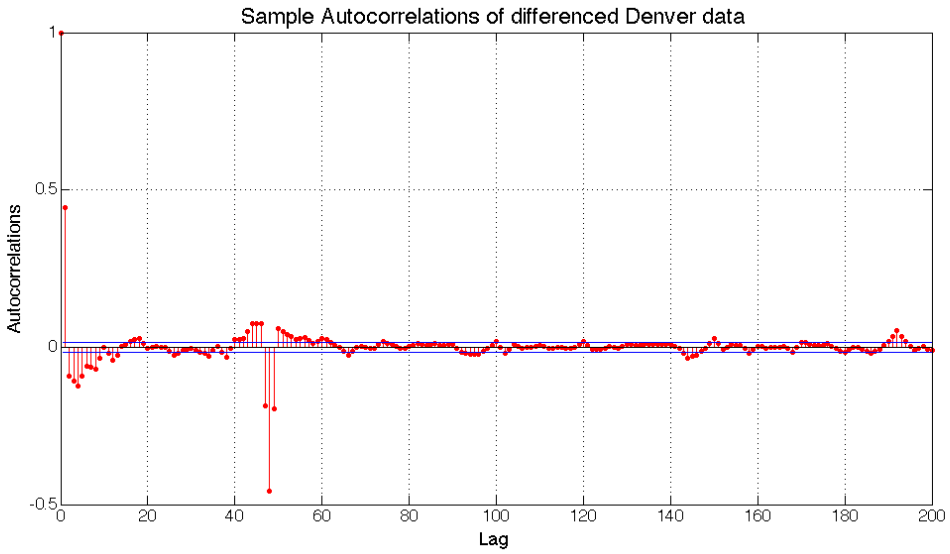
Determining auto regressive and moving average parameters

The autoregressive, moving average, differencing parameter and their respective seasonal parameter values for ARIMA models tend to be 0, 1, 2, or 3 [20]. Due to this small range of input values the total input space is relatively small (six parameters with four possible values equates to $4^6 = 4096$) allowing us to apply a brute-force search for the best model. Model performance is determined by the Akaike information criterion (AIC) [24]. Our optimal model is a seasonal ARIMA $(1, 0, 1)(0, 1, 1)_{168}$. This model is very similar to [15] who also fit a seasonal ARIMA model to traffic data.

For datasets where such brute force methods are not possible, a visual analysis of the residual plots is suggested. This analysis relies on looking a various residual autocorrelation and partial autocorrelation plots. Figure 2.4 displays the plots for our fit seasonal ARIMA model. Ideally these plots should so no correlation between the lags of the data. Our plots display a strong correlation on lag 48. This correlation is due daily seasonality (one day is similar to another day). However this correlation was weaker than our weekly seasonality and thus our introduced seasonal difference was for one week instead of one day. Because

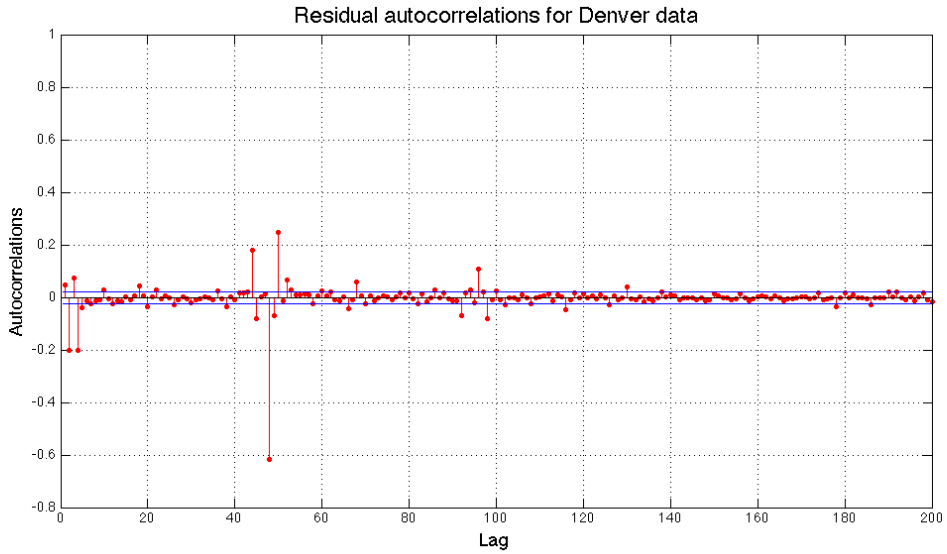


(a) Seasonal difference counts

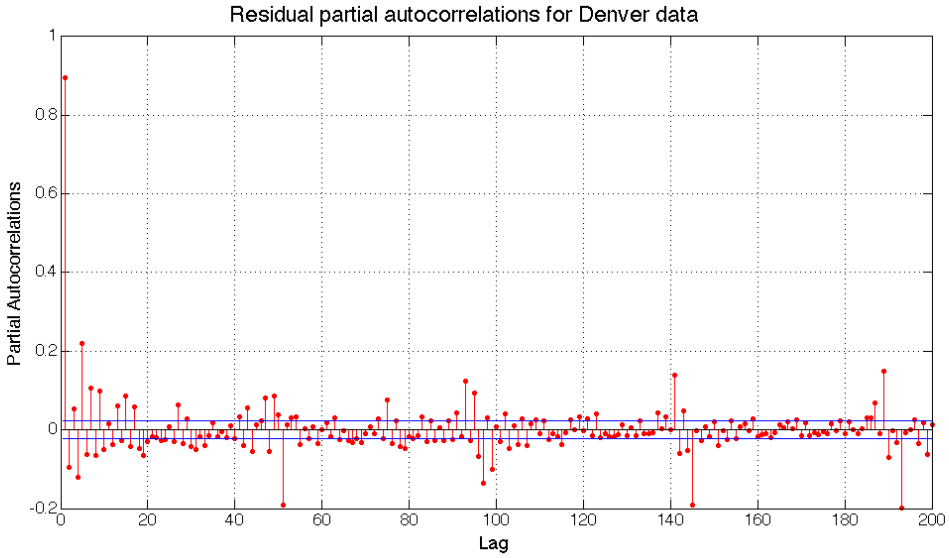


(b) Autocorrelation of seasonal difference counts

Figure 2.3: Denver data: one week seasonal difference counts and autocorrelation over a two week period.



(a) Residual autocorrelations for Seasonal ARIMA model.



(b) Residual partial autocorrelations for Seasonal ARIMA model

Figure 2.4: Denver data: Residual autocorrelation and partial autocorrelation results from fit seasonal ARIMA model.

our data exhibited little lag correlation with one large blip at lag 48 instead of the common slow decay of lag values, we believe this fit to be sufficient.

To verify our seasonal ARIMA fit is accurate Figure 2.5 shows an example of one-step ahead prediction performed on a sample week of test data. The mean absolute percentage

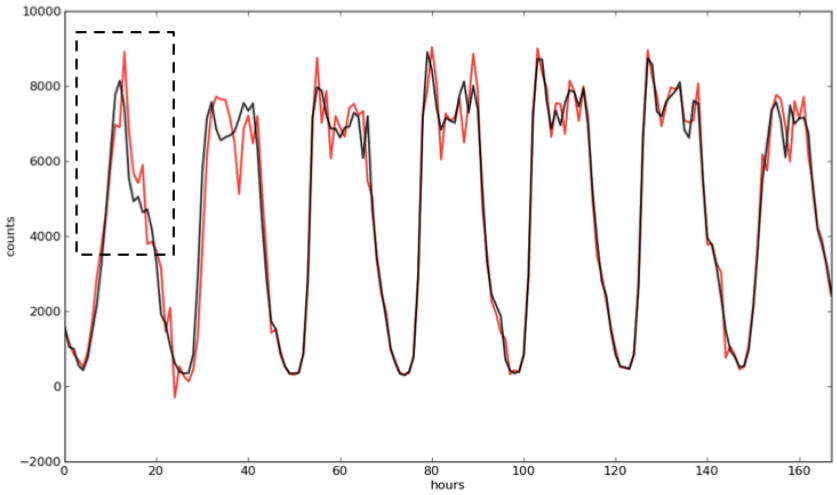


Figure 2.5: One-step ahead prediction for a sample week. Black line is original data. Red line is forecasted data. Dotted box shows an example of mis-forecasting due to a Broncos game.

error for this week was approximately 8.2%. This error score is close to the results from other authors on other vehicle traffic datasets [15, 25]. Also, for Figure 2.5 the dotted line boxes a time when a Broncos game was occurring. Forecasting during the Broncos game was initially low while traffic was unusually high as people were traveling to the game and then too high for much of the duration of the game. We discuss our approach to solving such misforecasts in Chapter 5.

Table 2.1: The parameter values that were fit for MERL and CSMBB datasets for a Seasonal ARIMA model

Dataset	p	d	q	P	D	Q	s
MERL	0	0	1	0	1	5	78
CSMBB	0	1	1	0	1	3	72
DENVER	1	0	1	0	1	1	168

Our final model parameters can be seen in Table 2.1. Each dataset we have for analysis has a strong weekly trend and a weaker daily trend, *i.e.* Mondays from one week should be similar to Mondays from another week, but Mondays may not necessarily be similar to

Tuesdays, Wednesdays or Fridays. We see that the best fit ARIMA models for each dataset find this weekly trend with the season parameter (s). Note, the season is different for each model due to a difference in number of readings for each day that we extracted data.

2.4 Historic average

This model is simply the per day average of readings at each time step. For certain types of data this model is has been shown to be more accurate than seasonal ARIMA forecasting [17], specifically when the data has a strong historic correlation. Average forecasts have the advantage of being extremely computationally fast and having a forecast accuracy that does not depend on the forecasting horizon. This result will be shown later.

TODO Explain this math quickly

$$Enterequationhere \tag{2.11}$$

$$Thiswilltakeabitmoreexplanationhere \tag{2.12}$$

2.5 Time delayed neural networks

Time delayed neural networks (TDNN) are a special subset of regression neural networks where the input data is a the previous Δ time steps of data from the time series. This special class of neural networks have been used frequently in vehicle traffic forecasting literature [27, 28]. Commonly the output is a single point forecast from that same time series at some point $t + \delta$ in the future. The form of our 1 hidden layer time delayed neural network is:

$$y_{t+1} = \phi\{\sum_{j=1}^{J} w_j \psi_j \left[\sum_{l=0}^{m} w_{ji} x_{t-l\delta} + w_{j0} \right] + w_0\} \tag{2.13}$$

where $\phi\{\bullet\}$ is a linear activation function on the output layer and $\psi[\bullet]$ is the standard sigmoid function. A visual representation of the node architecture of a time delayed neural network is displayed in Figure 2.6.

Forecasting is performed by computing the output for a $m + 1$ length window of time and then iteratively forecasting a set of time steps in the future by using forecast data as

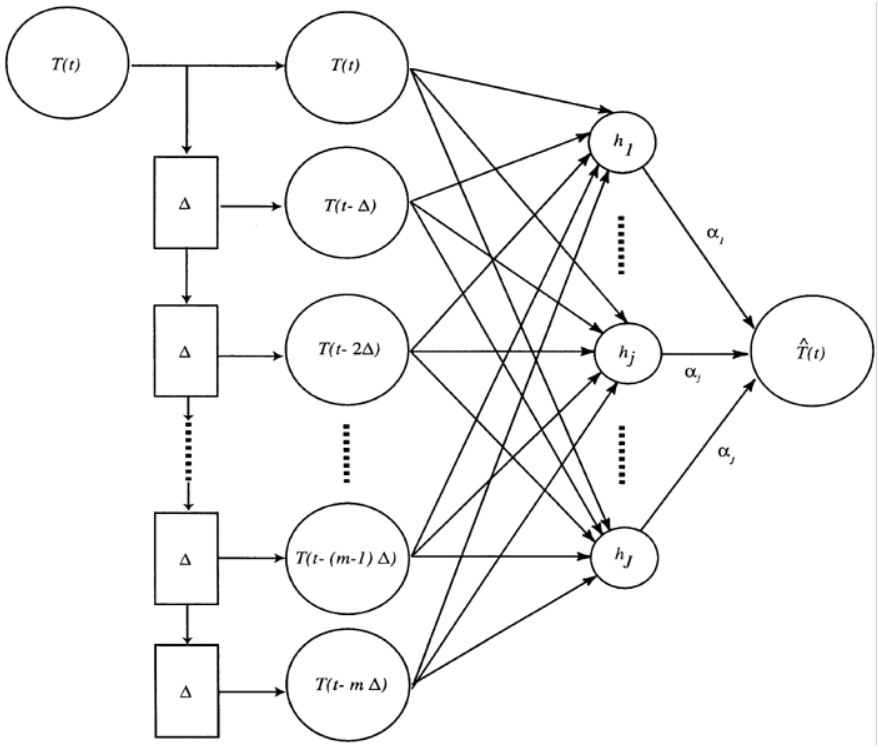


Figure 2.6: Architecture of a time delayed neural network with $m + 1$ inputs and J outputs [26].

inputs into the next forecast.

The number of input nodes and hidden nodes for each dataset is given in Table 2.2.

Table 2.2: Number of delayed input nodes and hidden nodes for MERL and CSMBB datasets

Dataset	Delayed input nodes	Hidden nodes
MERL	15	8
CSMBB	12	8
DENVER	6	6

2.6 Support vector regression

Support vector regression (SVR) is a universal learning method which offers a powerful way to forecast time series. It has been used in the past successfully to forecast travel times and vehicle counts for roadway traffic systems [18, 29–31]. Many of these empirical studies have demonstrated SVR as outperforming other forecasting techniques including ARIMA and TDNN.

Support vector machines were developed by Vapnik in 1993 [32]. SVM's use structural risk minimization to find an minimal upper bound on some expected risk. Training support vector machines to be used for regression is not done in the same way as other time series models. We first had to transform our dataset to a series of examples with a fixed length window. For a fixed window of size w, training input data is of the form $\{x_t, x_{t-1}, ..., x_{t-w+1}\}$. Target data is of the form x_{t+1}. Thus the training examples which we provided to our SVM was $\{x_{t+1}, [x_t, x_{t-1}, ..., x_{t-w+1}]\}$.

The general formulation for support vector regression is to minimize the following [33]:

$$\underset{w}{\text{minimize}} \quad \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n}(\epsilon_i + \epsilon_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - f(x_i, w) \leq \gamma + \epsilon_i^* \\ f(x_i, w) - y_i \leq \gamma + \epsilon_i \\ \epsilon_i, \epsilon_i^* \geq 0, i = 1, \ldots, n \end{cases}$$

This formulation is analogous to common "soft margin" formulation of the SVM. ϵ and ϵ^* are the slack variables which measure the deviation of training samples. The function $f()$ is the regression function and w are the parameters of the regression function $f()$. γ is loss function sensitivity and finally the constant $c > 0$ determines the trade-off between selection of the function f and the amount the slack variables ϵ and ϵ^* are tolerated.

To perform SVR training we used the popular *libsvm* package [34]. Because parameter selection is a notoriously difficult problem for SVR, we followed the guidelines as outlined by Hsu, Chih-Chang and Lin, who are the creators of the libsvm package. We first scaled the data by normalizing it between $[0, 1]$. Then we searched for our best values of C, ϵ and γ using the root mean squared error of the validation set factor to determine performance of those parameters.

For all of our datasets we used a historic window equal to length five. This happened to be the same window length used by [29] in past vehicle forecasting work. Also, we empirically tried four common kernel functions: linear, polynomial, radial basis and sigmoid. We had best for all three datasets with radial basis kernels, but a linear kernel also produced good results.

CHAPTER 3

DATASETS

This chapter contains information pertaining to the datasets used for our work. Our datasets are from three different sources. We have two building datasets and one freeway traffic dataset. The traffic dataset is a count of all the vehicles on a particular stretch of road aggregated over 60 minutes. Because most of these road stretches are on major highways, this tends to be a high volume dataset. The building datasets are much lower volume, but of vastly different types. One building is a college research and classroom building. The other is a typical office building. When considered as a whole, we believe these buildings represent a wide corpus of potential human traffic systems.

Because some of our forecasting models work better for scaled data, we first normalize the data to values between 0 and 1. To do this we performed the common scaling function:

$$x_t := \frac{x_t - x_{min}}{x_{max} - x_{min}}. \tag{3.1}$$

x_{min} and x_{max} represent the minimum and maximum values that x takes on within the time series.

3.1 Building datasets

While there has been considerable work in estimating building occupancy values, due to difficulties with acquiring accurate ground truth occupancy values, such datasets are rare. The problem in acquiring these occupancy values is that many buildings do not have sufficient infrastructure to accurately sense people throughout a building. One approach used by researchers is to use simulated models of occupancy [35, 36]. These agent based models have the potential for significant accuracy, but tend not to scale well to large buildings where the large number of agents, rooms and interactions lead to non-trivial solutions. Due to these problems with simulation approaches, we prefer to estimate occupancy from sensor

data rather than simulated data.

To estimate building occupancy from sensor data, numerous techniques have been developed. A common system uses a combination of simple sensors and wireless motes. Agarwal, et. al [37] created motes using a combination of IR sensors and reed switches placed on doors to determine the likelihood that a room is occupied. The focus was not on estimating the number of occupants, but instead to determine if the room was occupied at all.

Mamidi [38] and the University of Southern California have developed a building-level energy management system using a combination of motion detectors and environmental sensors to estimate the occupancy of rooms with multiple individuals present. Ground truth was collected and used as the basis for target values. These values, coupled with raw sensors were then used to train a machine learning algorithm that was implemented on the motes to estimate occupancy.

Meyn, et al [39] created a multiple mote occupancy estimation system using passive infrared sensors, carbon dioxide sensors and coupled this information with building badge counters and video cameras. In all of the above occupancy estimation system, the researchers were only concerned with estimation and not forecasting. We were unable to acquire the datasets for testing our approach. Also, due either to the low volume of occupancy or short duration it is likely the datasets produced by these researchers would not be sufficient for our work.

Our building datasets come from two sources. The first is a combined research and office building from Mitsubishi's Electronic Research Lab (MERL) dataset [40]. To our knowledge, this dataset is the closest to a research standard building occupancy dataset that we could find. The second is a classroom and office building from the Colorado School of Mines (CSMBB) [18, 41]. Both datasets use passive infrared sensors (Figure 3.1) to estimate motion in an area.

Due to the nature of IR sensors, we are only able to detect motion instead of actual occupancy; for example, a group of three people would occur as one reading in both systems.

Figure 3.1: Passive infrared motion detector

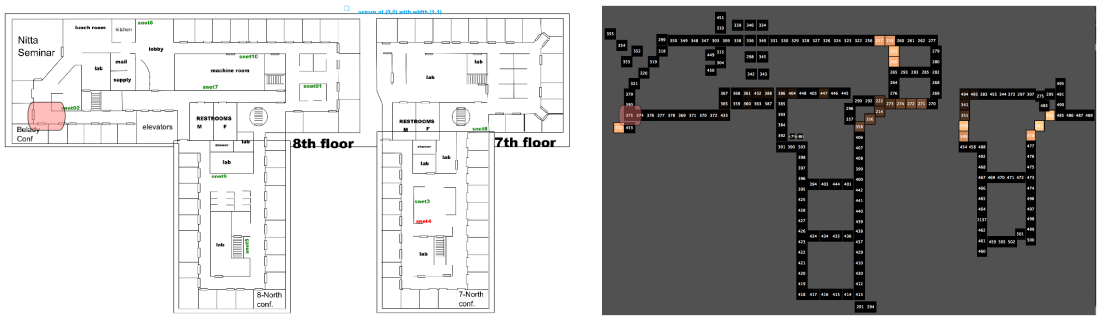
Despite this drawback, real occupancy data would likely be similar to our data, but with higher variance and higher means. As the range of occupancy estimates in our two datasets are quite different and we are able to achieve accurate estimates in both scenarios, we do not foresee problems when applying our forecasting techniques to more accurate estimated values. We thus believe this data sufficient to test our occupancy estimation algorithms.

3.1.1 MERL dataset

The Mitsubishi Electronic Research Labs dataset is derived from a collection of over 200 passive infrared sensors place densely throughout the 7th and 8th floor of a research office building. This dataset has been used as the basis for multiple areas of research [40, 42–47]. However, none of this research to our knowledge focused on building occupancy forecasting. The closest related work focused on tracking individuals within the building.

The sensors are placed roughly two meters apart on the ceilings, creating a dense sensing area with little non-sensed space. Readings are taken at the millisecond level, but due to the sensors' settling times the inter-detection time of motion is approximately 1.5 seconds. A representation of this floor plan is given in Figure 3.2.

The data was collected from March 2006 through March 2008 and there are roughly 53 million sensor readings. This building is similar to most office buildings with a number of personal offices along with labs and conference rooms. Employees have roughly set schedules

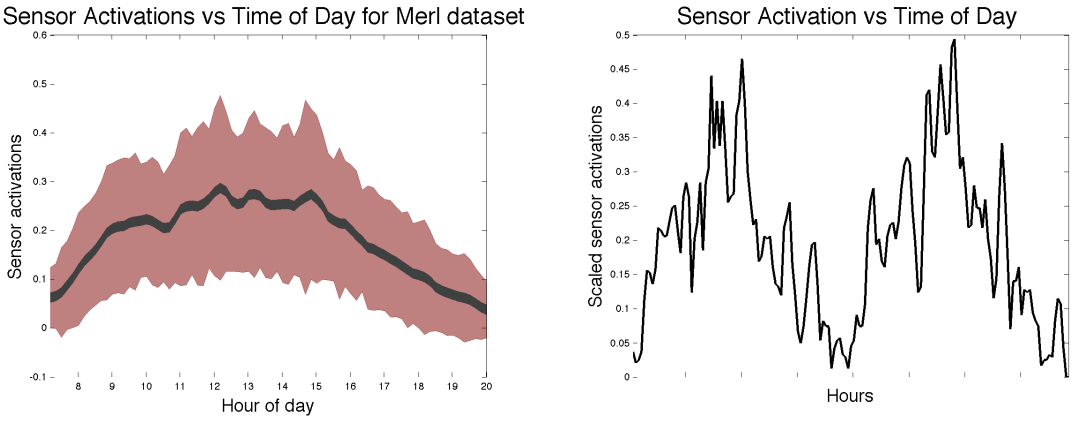


(a) (b)

Figure 3.2: Floor plan and sensor locations for the MERL office building dataset. The red rectangle in (a) and (b) is the location of our sensor for this work.

and holidays are observed as normal. We chose a sensor near a commonly used conference room to give us a significant number of daily readings. Many of the other sensors throughout the building had few readings as they covered areas with little traffic near employee office areas.

The counts of sensor activations have been aggregated every 10 minutes. Because of the lack of significant motion at night, we look only at activations that occur between 6:00am and 7:00pm daily. A plot of the average activations of all days for a single sensor along with a range of one standard deviation is given in Figure 3.3(a).



(a) (b)

Figure 3.3: Scaled average readings for a given day and scaled readings for two days from the MERL dataset.

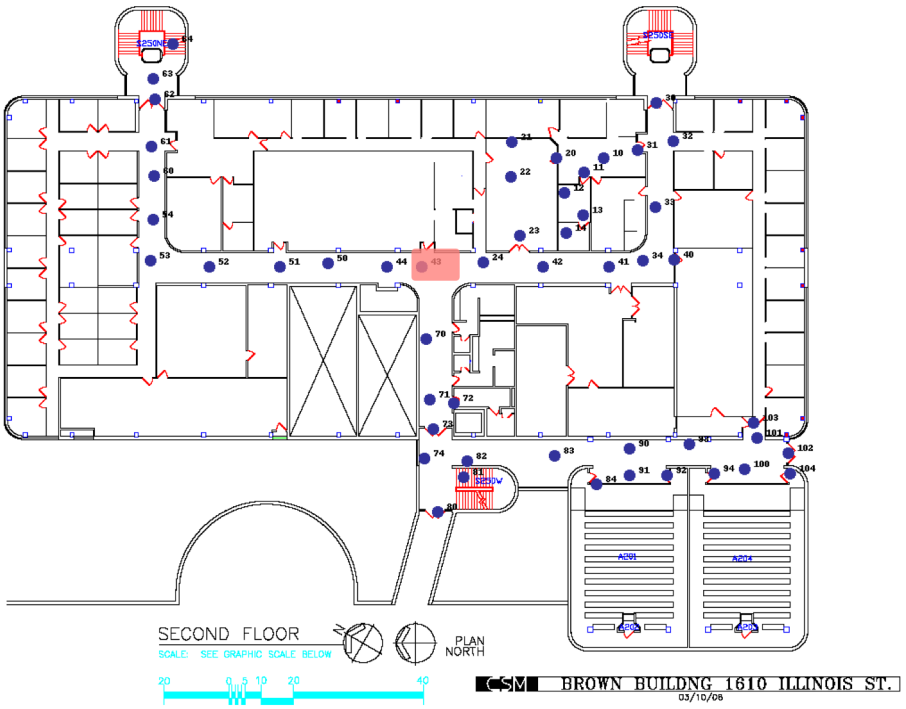


Figure 3.4: Sensor locations for the Colorado School of Mines Brown Building. The red rectangle is the location of the sensor used for our analysis.

Peak motion unsurprisingly occurs during the middle of the day corresponding to lunch time. Peak standard deviation appears to occur near noon and 4pm. The noon variation is unsurprising as people will commonly skip lunch for various reasons. The 3pm variation is more surprising. This seems earlier than expected for daily departure, but perhaps it is at the early window of when people will begin to depart for the day.

An image representing sample values throughout two days is given in Figure 3.3(b). Despite the mean appearing smooth in Figure 3.3(a), we see how volatile sensor activation readings are though out the day.

3.1.2 Colorado School of Mines dataset

The Colorado School of Mines dataset [18, 41] is a collection of 50 passive infrared sensors mounted on the ceiling of the second floor of a class and office room building. The density of the sensor placement depends on the location within the building. Outside the auditorium in

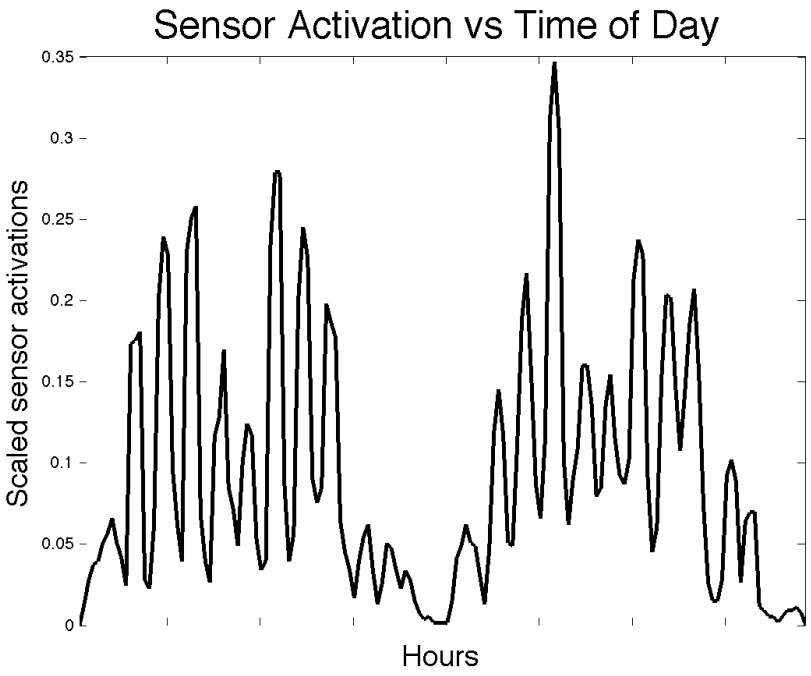


Figure 3.5: Sensor readings for two consecutive days from the Brown Dataset.

the lower right of Figure 3.4 is a dense collection of sensors placed approximately every few meters. Throughout the rest of the building the sensors are placed roughly every 5 meters. Data was collected for one academic school year from 2008 to 2009 and there are more than 23 million sensor readings. To acquire readings, the sensors were polled every second and recorded data if motion was detected.

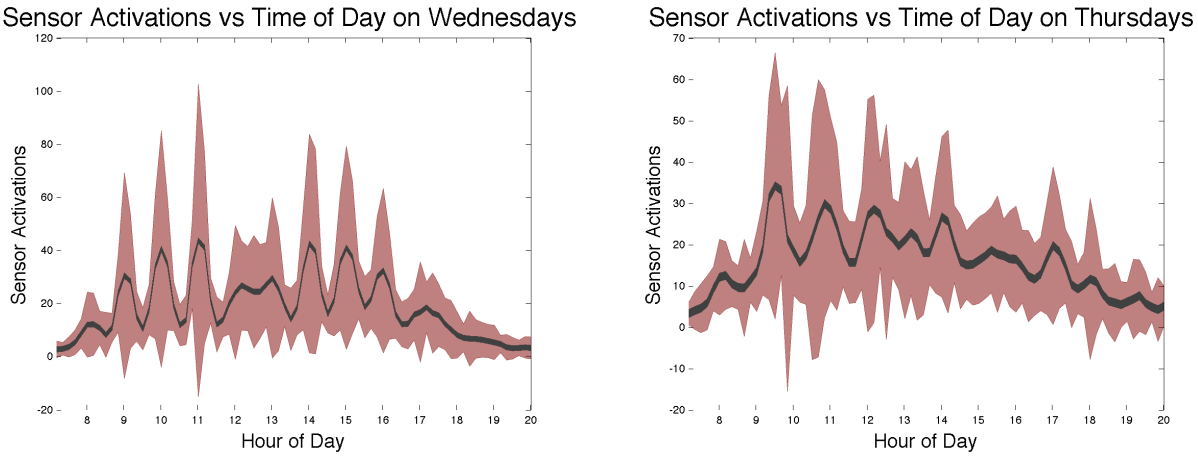
This dataset is much different than the MERL dataset as classes typically provide activity on a rigid schedule during the day. Also as students have exams and projects, late night motion is sporadic based on the time of year. The counts of sensor activations have been aggregated over every 10 minutes. Despite occasional late night motion during exam time, most nights have no significant motion. For this reason we focus on data between 7:00am and 7:00pm daily. Typical daily class periods are seen in Figure 3.5. This image shows two consecutive days. The class passing times are easily visible with increased traffic happening roughly every hour.

Also, different days of the week not only have different schedules, but have different class lengths. Classes at the school typically fall into two different schedules; either a class meets on Monday, Wednesday and Friday where the classes are 50 minutes in length or a class meets on Tuesday and Thursday where the classes are 75 minutes in length. A plot of the average activations of all Wednesdays and Thursdays for a single sensor along with a range of one standard deviation is given in Figure 3.6. The defined peaks in the dataset correlate to class start and end times when most students will be in the hallways of the building. Notice how the peak times differ from Wednesday to Thursday.

For this work, we focus on the Monday, Wednesday and Friday class schedules.

3.2 Traffic datasets

In contrast to building occupancy datasets used for forecasting, there exist numerous traffic datasets which are used in the time series forecasting literature [15, 48–50]. Vehicular traffic datasets are an excellent test for forecasting algorithms as they tend to be highly repetitive with a significant seasonal component.



(a) (b)

Figure 3.6: Average readings from Wednesdays and Thursdays for a given sensor from the CSMBB dataset.

3.2.1 Denver traffic dataset

The Denver traffic dataset is collected from many doppler radar vehicle counters which count the number of vehicles passing through a small region of road throughout the day. Data is available from 2008 to 2013 on most days of the week. The data is nearly 1 million readings for the sensors we extracted and approximately 33,000 readings for any one sensor.

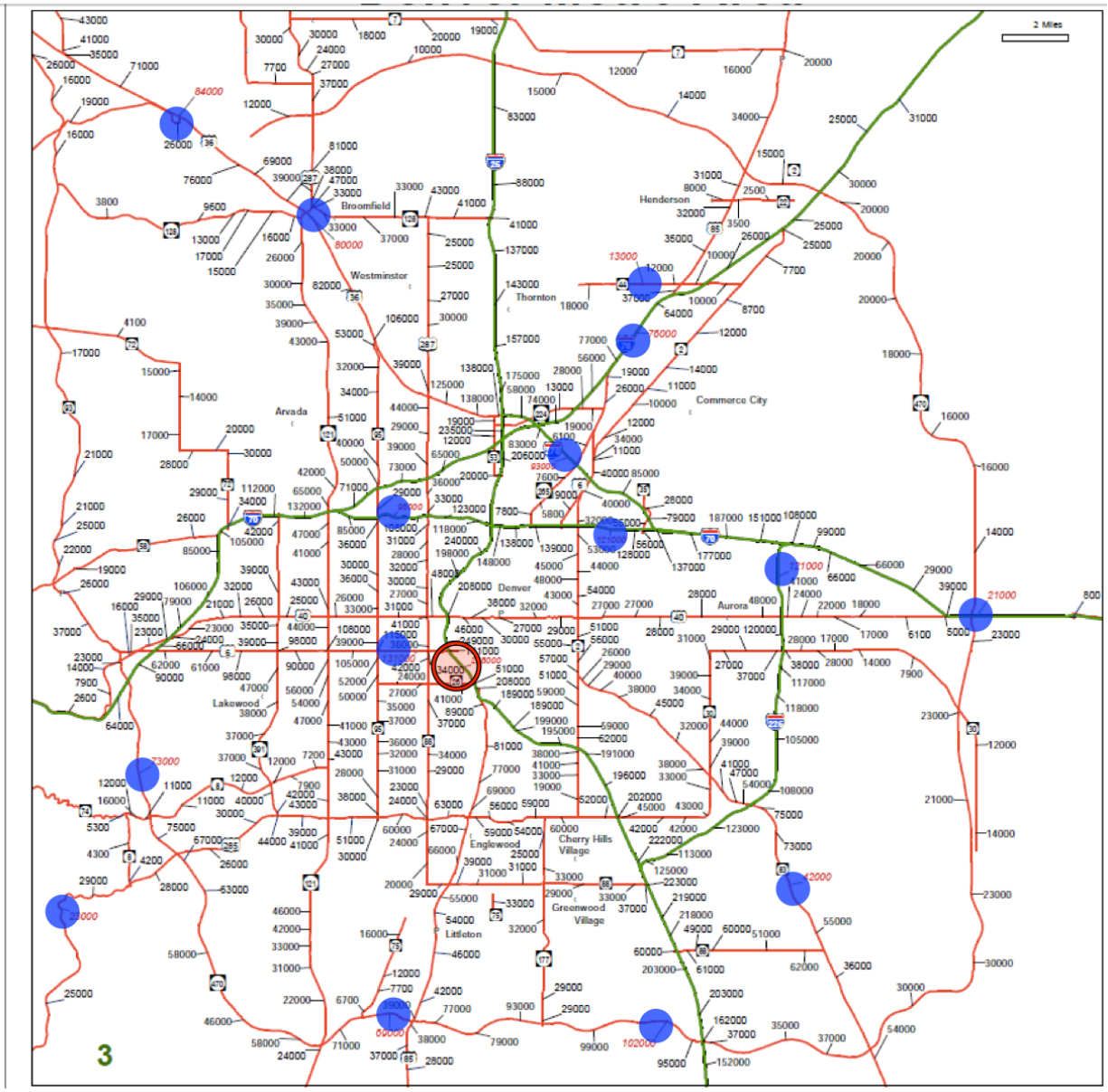
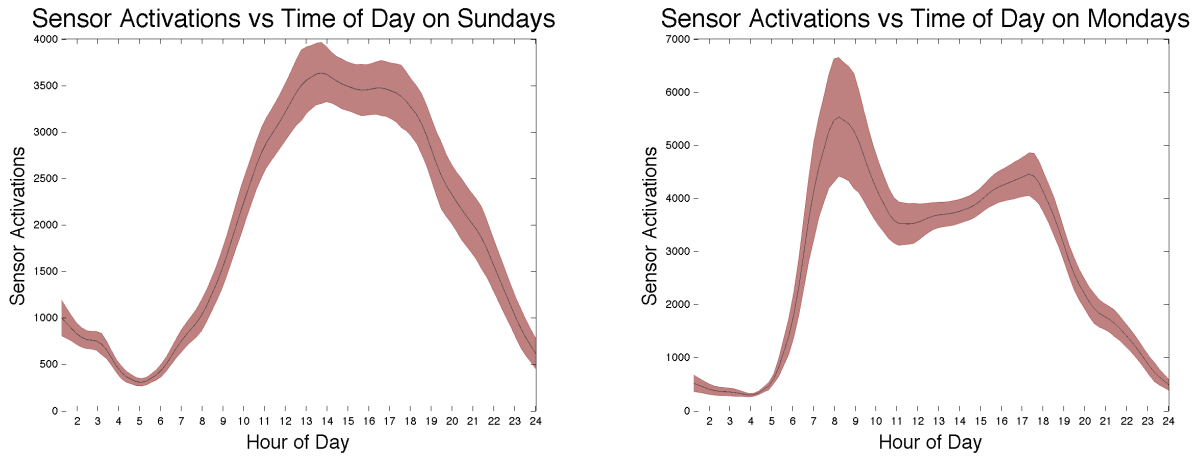


Figure 3.7: City of Denver traffic sensors. Data analyzed for this work is from the sensor in red.



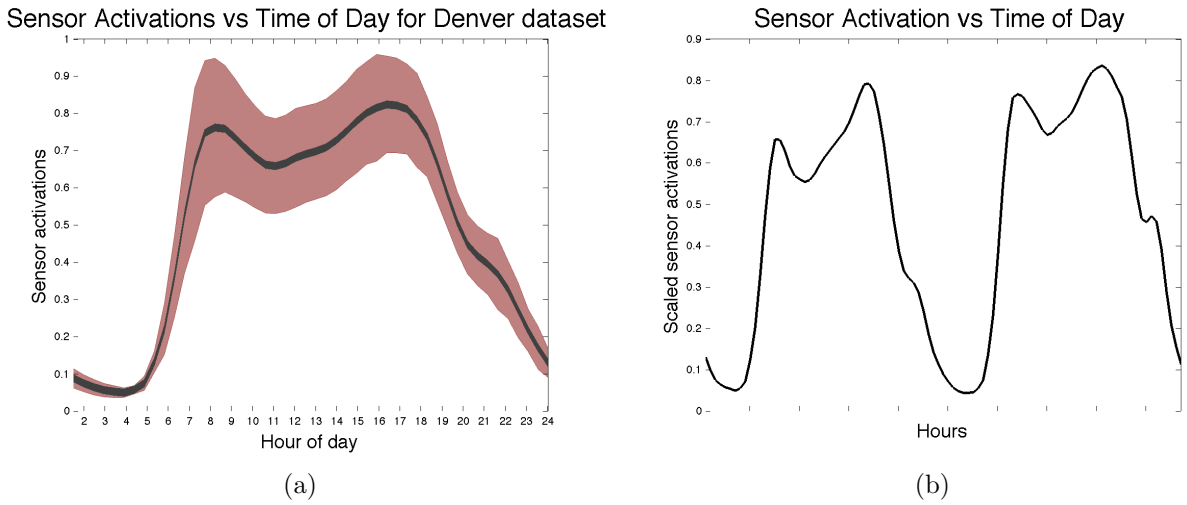
(a) (b)

Figure 3.8: Average readings from Sunday and Monday for a given sensor from the Denver traffic dataset.

For this work we use a freeway sensor near downtown Denver, closely located to Mile High Stadium; the home of the Denver Broncos. The sensor is located on Figure 3.7 represented by a red rectangle.

Count data is aggregated per hour for each direction of traffic at every sensor location. This data is highly repetitive as Monday through Thursday have approximately the same daily traffic patterns. Friday behaves much like the rest of the weekdays with the differences being that evening rush hour happens about an hour earlier and their is an increase in night activity. Example averages for a Sunday and Monday are represented in Figure 3.8. This figure clearly shows how Monday has a peak time during morning rush hour and a small peak again during afternoon rush hour. Sunday shows no such pattern. Instead Sunday shows more slow increase in activity throughout the day. Also of note is the large amount of sensor activations per hour compared to the CSMBB and MERL datasets. The counts for Denver traffic are thousands of times larger than our building datasets.

The scaled dataset used for this work is shown in Figure 3.9(a). This is an average of all Monday through Thursdays for the sensor indicated earlier. Also, unlike in the other datasets, we do not cut the data down to specific times of day. This is done, both because



(a) (b)

Figure 3.9: Scaled average readings for a given day and scaled readings for two days from the Denver traffic dataset.

there is more traffic during the early morning and late night on this dataset compared to the others and because the data is aggregated only every hour, and we do not want to remove more readings from the dataset. Finally Figure 3.9(b) shows an example of two days of scaled data. Due to the 60 minute readings instead of 10 minute readings, the data appears much smoother.

3.3 Additional notes on the datasets

Variation in datasets

These three datasets are distinct in many ways. They differ significantly in total volume of sensor activations, levels of noise in the systems and types of patterns within the datasets. We believe this variation sufficient to demonstrate the efficacy of our approach to forecasting time series for human controlled environments.

One sensor vs Multiple sensors - why we use a univariate dataset

For this work, we use only one sensor to perform our forecasts. While intuition may imply that a multivariate yields greater forecasting accuracy when compared to a univariate ap-

proach; Kamarianakis and Prastacos [48] demonstrated this is not necessarily true. They studied the problem of forecasting vehicular traffic flow and found similar results using both multivariate and univariate approaches.

We believe the same results hold when it comes to building datasets instead of vehicle datasets. This is primarily motivated by the geographic size of our data and the time of reading aggregation. For our building datasets, the data is accumulated in 10 minute intervals. That resolution of detail was empirically selected because it smooths the data sufficiently to allow for accurate forecasting while still being short enough to provide meaningful forecasts. 10 minutes is however, longer than it takes an individual to walk from one end of our buildings to the other. Therefore at a scale of 10 minutes, knowledge of occupancy in one sensor does not seem to imply occupancy in another sensor. This same scaling problem exists for the Denver dataset as well. Unless traffic is exceptionally bad, the travel time across all of Denver is less than one hour.

Missing values in the dataset

Due to data collection problems with sensors, the raw data from all of our datasets all had segments with missing values. These missing values were replaced with the historic average value for that sensor at that time on that day. For example a missing value at 3pm on a Tuesday in the Denver dataset would be replaced by the average value of all Tuesdays at 3pm.

CHAPTER 4

A ENSEMBLE FORECASTER FOR LONG TERM FORECASTING

Previous chapters introduced both a set of common forecasting models and the some traffic datasets. Here we look towards an ensemble to outperform our set of previously introduced forecasters. This chapter details our work on modifying the Bayesian combined forecaster and applying it to our datasets. At the end of the chapter we present results on how this ensemble forecaster works compared to the more common forecasters presented earlier.

4.1 Why ensemble forecasting?

Instead of simply using a single model, ensemble forecasting uses a combination of multiple models to perform a single forecast. These models may all be from the same class of model (i.e. a neural network), but trained with different different data or parameters or they may be models from a mix of classes. Later in this chapter we explore a mixed model ensemble further.

Because ensemble forecasting accounts for the uncertainties of its component models, there are a number of advantages that present themselves over a single model forecaster. For example, [51, 52] present an ensemble forecaster which can reduce nonlinear error growth for future forecasts by averaging out unpredictable components. It can also make predictions on the skill of individual forecasters and on its own accuracy by looking at how much agreement exists among the forecasts. A stronger forecast agreement likely yields more accurate forecasts. Also many ensemble forecasters stem from a strong statistical basis giving forecasts as a likelihood weighted outcome instead of a hard value. This weighted likelihood is computed by assessing the residual error of individual forecasting models. Thus even forecasters without any statistical basis may be imbued with stochastic properties.

4.2 Bayesian combined forecasting

The BCF approach [9] is one of several types of methods which attempt to combine other forecasting models for time series. We selected this forecasting method over other multiple model forecasting methods (such as mixture of experts or ensembles of neural networks) due to its modularity and strong statistical backing. BCF is modular in that it allows for the component forecasting models to come from any trained forecaster with a well defined distribution of the forecaster's mis-forecasts. Its statistical backing comes from its direct derivation from Bayes' rule.

To derive BCF we first assume the existence of K models. From these K models, we want to create a probability distribution on a new random variable z that is used to determine if model k is the correct model from which to forecast at time t. To do this we use the notation of Petridis [9] and define p_t^k as follows

$$p_t^k = p(z = k|x_t, ..., x_1). \tag{4.1}$$

From here we apply Bayes rule and get

$$p_t^k = \frac{p(x_t|z = k, x_{t-1}, ..., x_1) \cdot p(z = k|x_{t-1}, ..., x_1)}{p(x_t, ..., x_1)}. \tag{4.2}$$

Notice that $p(z = k|x_{t-1}, ..., x_1) = p_{t-1}^k$. Using p_{t-1}^k we can create a recursive estimation based on prior p_t^k.

With recursive values for p_t^k and replacing $p(x_t, ..., x_1)$ with a sum of conditional probabilities on z we get

$$p_t^k = \frac{p(x_t|z = k, x_{t-1}, ..., x_1) \cdot p_{t-1}^k}{\sum_{j=1}^{K} p(x_t|z = j, x_{t-1}, ..., x_1) \cdot p_{t-1}^j}. \tag{4.3}$$

We use the empirically observed forecasting error for each model to estimate $p(x_t|z = k, x_{t-1}, ..., x_1)$. The forecasting error for a given model at time t is

$$e_t^k = y_t^k - x_t. \tag{4.4}$$

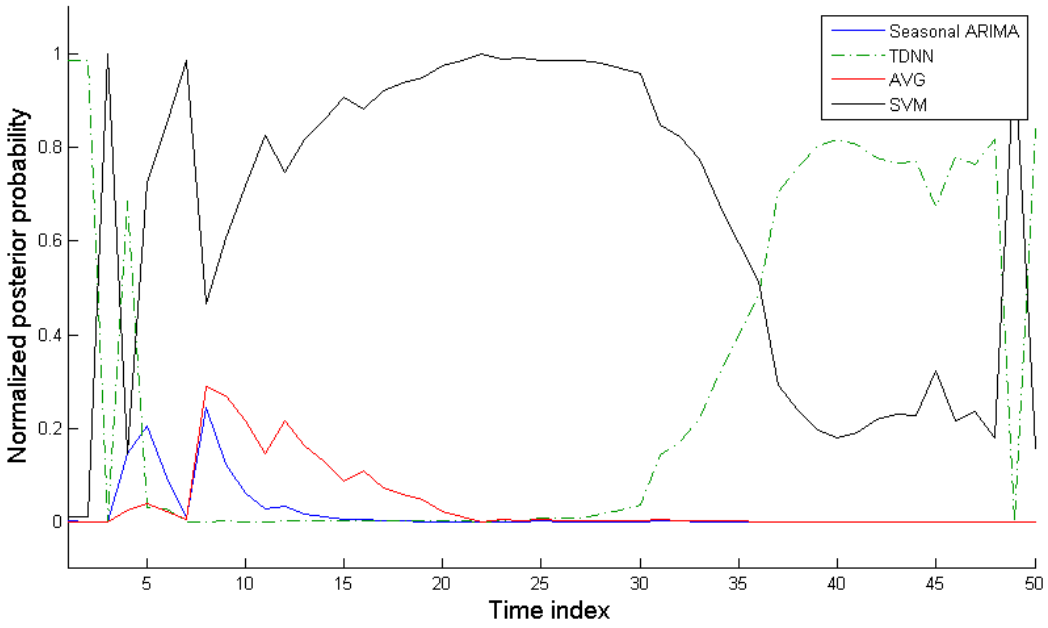


Figure 4.1: Normalized posterior probabilities of component models on a section of MERL dataset.

We can use these forecasting errors to estimate a probability distribution for each model on the random variable e_t^k. We allow this distribution to take on a set of parameters represented by ω_k. Thus for each model the probability error distribution function on the model error random variable is given by $q(e_t^k; \omega_k)$. In practice, this error is typically modeled as a white noise zero mean Gaussian process. For our work, we also assume Gaussian noise and hence our parameterization we are searching for is simply the mean and standard deviation of our noise.

The final equation for the posterior probability of a given model k is

$$p_t^k = p(z = k|x_t, ..., x_1) = \frac{p_{t-1}^K \cdot q(x_t - y_t^k; \omega_k)}{\sum_{j=1}^K p_{t-1}^j \cdot q(x_t^j - y_t^j; \omega_j)}. \tag{4.5}$$

Forecasting using BCF is done by either computing a weighted forecast δ time steps into the future for each forecasting model or by simply selecting the model with the highest likelihood. For this paper we forecast using a weighted forecast of all models. The forecasting

equation is

$$x_{t+\delta}^{ALL} = \sum_{k=1}^{K} p_t^k \cdot y_{t+\delta}^k. \tag{4.6}$$

An example of these changing normalized posterior probabilities for a small section of the MERL dataset is shown in Figure 4.1. In this figure, we perform a forecast using four base models: Seasonal ARIMA, TDNN, SVM and Historic Average. Our BCF forecaster is forecasting one time step into the future for each time index 1 to 50. This figure demonstrates that each of the component models applies a weighted forecast to the final forecast. This forecast changes over time according to how accurate that individual model was in a recent set of forecasts.

4.3 BCF modifications

In this subsection we discuss a number of modifications to maximize the effectiveness of BCF for our data. We refer to the modified BCF algorithm as Bayesian Combined Forecasting for multiple Time Steps or BCF-TS for short. These modifications enable BCF to work with forecasting horizons greater than one in the future.

4.3.1 Forecast δ time steps into the future

Traditional implementations of BCF in other domains [9, 53] are interested only in 1 time step ahead forecasts. For our work we require forecasts that are δ steps ahead which requires a small change to the BCF method. Recall from our notation section that a forecast for dataset x is represented as $y_t = f(x_{t-1}..., x_1; \theta_k)$, where θ_k is the parameterization of the forecasting function for model k and y_t is the forecast for time t. Using this notation, to forecast δ time steps into the future, we need to determine an estimate of the error of such a future forecast. Thus instead of generating a model's error distribution from

$$e_t^k = y_t^k - x_t = f(x_{t-1}, ..., x_1; \theta_k) - x_t. \tag{4.7}$$

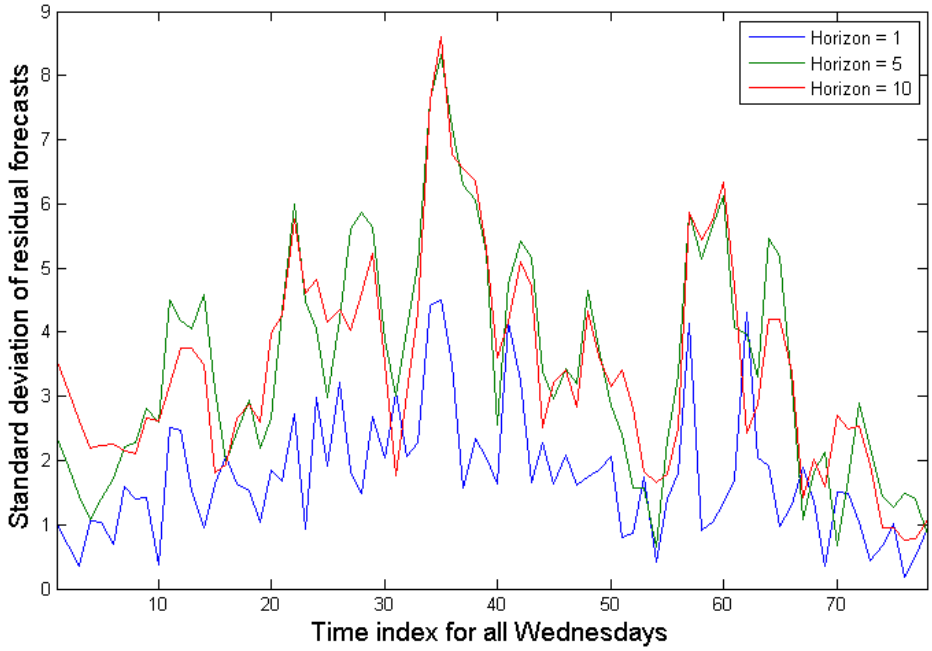


Figure 4.2: Standard deviation of support vector machine residuals for all Wednesdays in MERL dataset. Time index represents 10 minute intervals from 6:00am to 7:00pm.

The error distribution is instead generated from

$$e_t^k = f(y_{t-1}, ..., y_{t-\delta+1}, x_{t-\delta}, ..., x_1; \theta_k) - x_t. \tag{4.8}$$

The reason for this change is due to the assumption that our error distribution is an accurate representation of forecasting accuracy. Notice that the primary difference between these equations is that the output of $f()$ is fed back into the function to "feed forward" ahead further forecasts. The forecasting error distribution for models at 1 time step into the future is not necessarily the same as models at δ time steps. To account for this we compute a different error distribution for each forecast time step.

4.3.2 Improving model error distributions

The standard implementation of BCF assumes a single fixed error distribution for each component model. An inspection of the residuals of each model demonstrates some clear daily and weekly trends. See Figure 4.2 for an example of how the forecasting error distribu-

tion for a trained support vector regression model on the MERL dataset depends on the time and on the forecasting horizon. From this figure, it is clear that we get peaks of forecasting error that occurs at roughly time index 12 (8:00 am) and another at about time index 35 (11:50 am) and finally one at about time index 60 (4:00 pm). These residual peaks persist at horizons of 1 and 5, but significantly drop off by horizon 10. For a short data horizon, it is clear that the residual error distribution is not fixed.

To represent a more realistic error distribution instead of a fixed white noise Gaussian that is commonly used in the literature, we fit a Gaussian for each slice of a day. For example, the data from the MERL dataset was used from 6:00am to 7:00pm. The thirteen hours of data used per day represent 78 time slices. These Gaussians are computed from a validation set representing 20% of our data. It is from this set of model error distributions that we compute BCF.

As a possible improvement to this set of error distributions, we note that using a generalized autoregressive conditional heteroskedastic (GARCH) model [20] or some other appropriate model to forecast future variance based on local and historic changes in variance would likely outperform our time based average Gaussian models. GARCH models are similar to seasonal autoregressive moving average models which we use as one of our component forecasting models except that they are regressive on the local residual terms of the model.

4.3.3 Model selection thresholding

Diebold [54] cautions against the use of forecasting using a Bayesian combination of models in all cases. Diebold points out that under certain situations a convex combination of forecasts for models may not be optimal, and cases exist where taking negative likelihood weighting may be optimal. These conditions are likely to arise during instances where the data may not be accurately described by any of the forecasting models.

Furthermore, due to noise in the data, there exist short windows of the data where no model is able to provide a sufficiently accurate forecast. During these windows of time, forecasts will often come from the historically worst model. The reason for this phenomenon

is due to the way in which BCF performs model selection. The forecasting residuals of each model is trained by the BCF forecast for each base model. The model with the worst forecasting performance will often have a large range of residuals and thus have the highest forecasting variance. Because the this variance is much higher, the likelihood of the worst historic model is typically higher than any of the more accurate models with a lower forecasting variance. This leads to the worst forecasters doing most of the forecasting during noisy events.

The behavior of the worst forecasters having the largest forecasting weight during noisy events is not problematic for that individual forecast from which a noisy event happened, as no forecaster is likely to forecast accurately a single blip of noise. The problem is that each forecaster's posterior probability is recursive and uses the likelihood of prior forecasts to determine the likelihood of a future forecast. This means that each time there is a blip in the data, the worst forecasters will typically be given the most weight for the next few forecasts until enough time has passed for the posterior probability of each forecasting model to return to typical ranges.

To combat this situation, we have implemented a model selection threshold h_k. If the likelihood of all component models is below h_k, then we forecast from only the model which is historically the most accurate based on our validation set.

The threshold is different for each model, and should depend on the error distribution of the model. In practice we have found that 2σ serves as a good threshold. Basing the threshold on σ is useful as it provides a threshold value which does not depend on e_t^k. For a zero mean Gaussian the probability of the 2σ threshold is

$$p(2\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-2}. \tag{4.9}$$

Because the Bayesian combined forecasting approach is iterative, it is possible that a long section of forecasts that indicate one model correct or incorrect can lead to likelihood underflow. Due to this problem we adjust our normalized likelihoods so that no model may reach a value below 0.001. This empirically chosen value is low enough to not have a great impact

on forecasts while still being high enough to allow model likelihoods to change quickly.

4.4 Results of BCF and BCF-TS

BCF and BCF-TS (BCF with our specific set of modifications) were trained and tested using all component models described above. All of the models were trained on 60% of the total datasets. Another 20% was used for model validation and the final 20% used for testing. The results shown below are on the test set only. Figure 4.3 shows a sample section of test data from the MERL dataset along with BCF-TS forecasts for horizons of 1 and 5. As expected as the forecasting horizon increases the forecasts become less accurate.

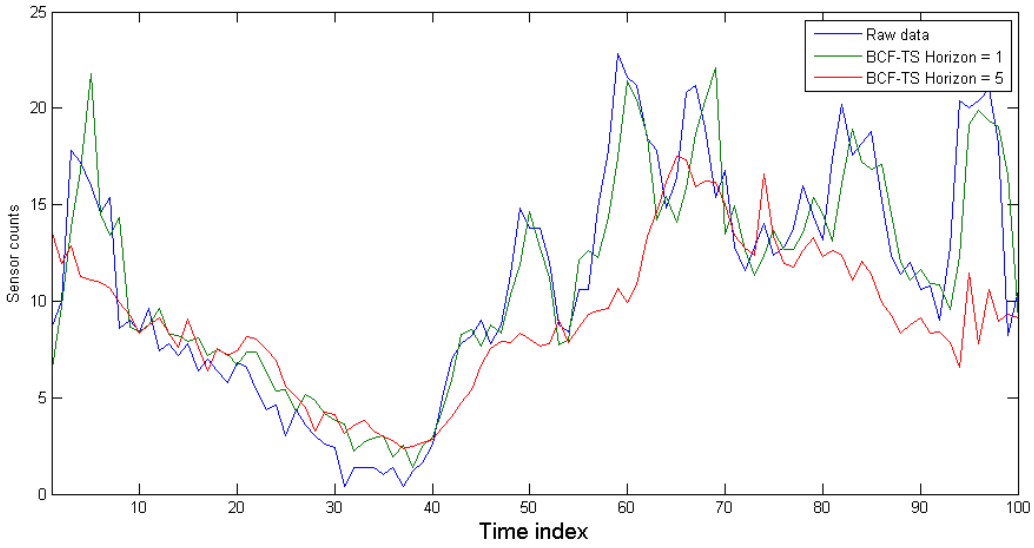


Figure 4.3: A comparison of forecasts at various horizons against real data for an sample time segment using BCF-TS.

It is common for one model's normalized posterior probability to be near one when that model is currently accurate. Figure 4.4 shows as example of this behavior. From time index 1 to 8, the SVM component model has a posterior probability near 1.0 and as a result BCF-TS forecasts nearly completely from this model. Then from time index 9 to index 35 the SVM model's posterior probability is lower and as a result BCF-TS uses other models for its combined forecast. This figure demonstrates the advantage of component model switching on

the final forecast. During areas of poor forecasting performance by one model, it is possible that another models will accurately forecast.

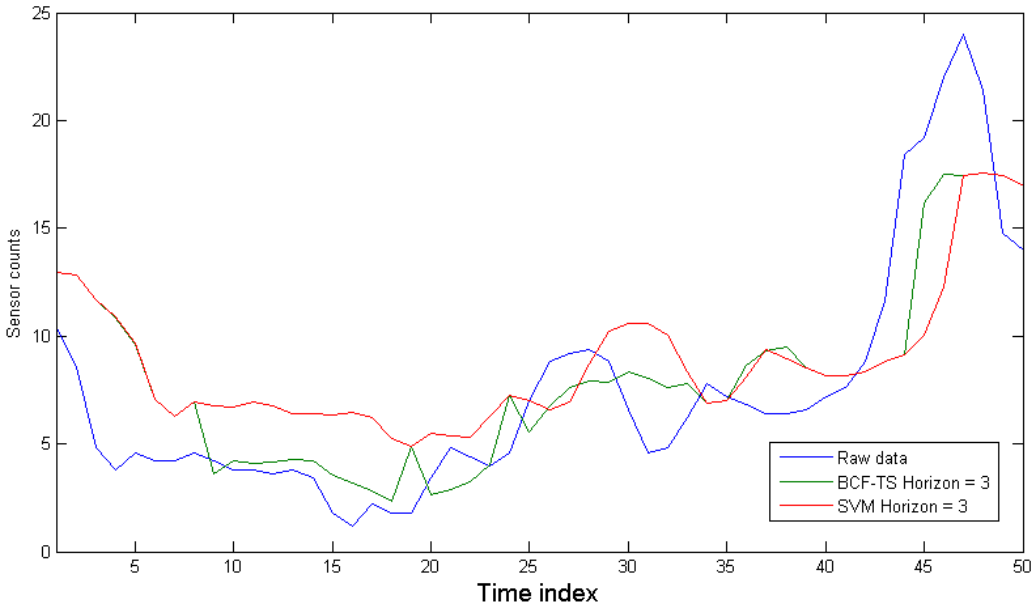


Figure 4.4: A comparison of BCF-TS and SVM forecasts at horizon equal to three against real data.

Figure 4.6, Figure 4.5 and Figure A.1 show the results of the RMSE and MASE results of forecasts across a forecast horizon up to 15 time steps into the future for each model. These plots show that BCF-TS generally has the lowest error for forecasts more than a couple of time steps into the future. However, the average model shows itself to be a strong indicator of future activity for forecasts beyond 6 to 8 time steps into the future.

In the CSM Brown building dataset the Seasonal ARIMA model was a good forecaster of future activity while in the MERL set it performed significantly worse than even the average model on all nearly all forecasting horizons. This is likely due to a stronger seasonal component to the Brown building dataset due class schedules. Instead, on the MERL dataset there is little seasonal correlation and thus natural variance from a prior season may incorrectly affect current forecasts. This result is similar to that of other papers that use seasonal ARIMA models [17]; where in the case of strong seasonal data, results are better for short

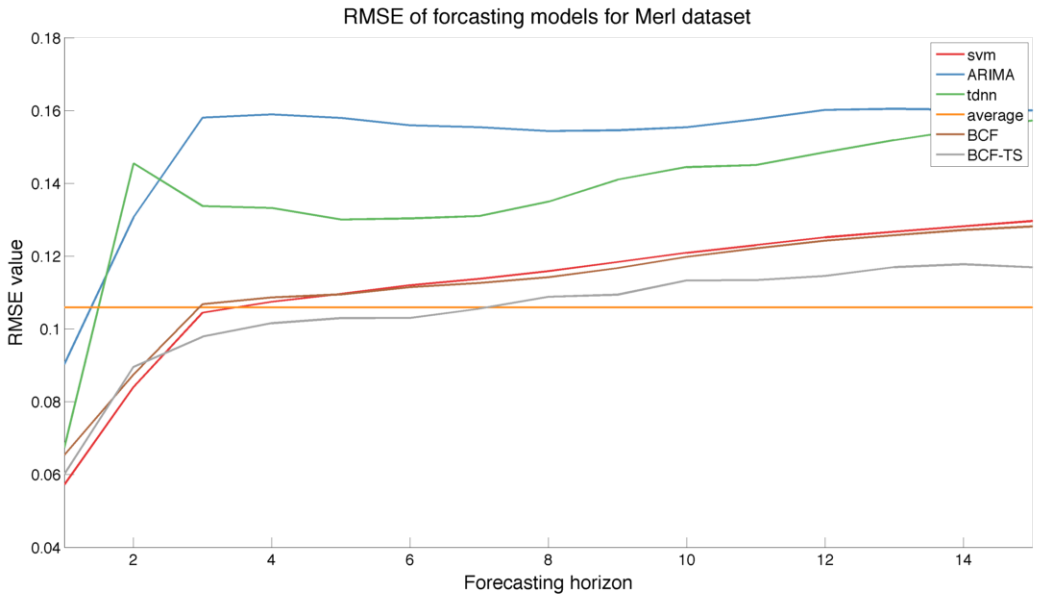


Figure 4.5: Merl dataset: RMSE of forecasting for each model vs forecasting horizon.

horizon forecasts, but longer forecasts favor historic averages. Scenarios such as the one demonstrated here where one forecaster is better suited for a short forecasting horizon and another forecaster is for long forecasting horizon are excellent provide excellent candidates for Bayesian combined forecasting.

In the Brown hall dataset **??** BCF and BCF-TS produce very similar results, with BCF-TS only demonstrating modest improvement in most time steps. This is likely due to forecasts by both SVM and ARIMA component models remaining generally in agreement for throughout our forecasting horizon. Due to this agreement BCF is able to remain an accurate forecaster and is not subject to the typical underflow problems discussed earlier. Despite this strong forecasting from BCF, our changes still allowed BCF-TS to outperform BCF in most cases.

In each of our data sets, BCF and BCF-TS generally outperformed the component models given all time steps. In some cases for certain time steps, a component model may be more accurate, but generally our ensemble forecasters were more accurate. Figure 4.8 shows the percent improvement of BCF-TS over BCF on each of the datasets. In general, our

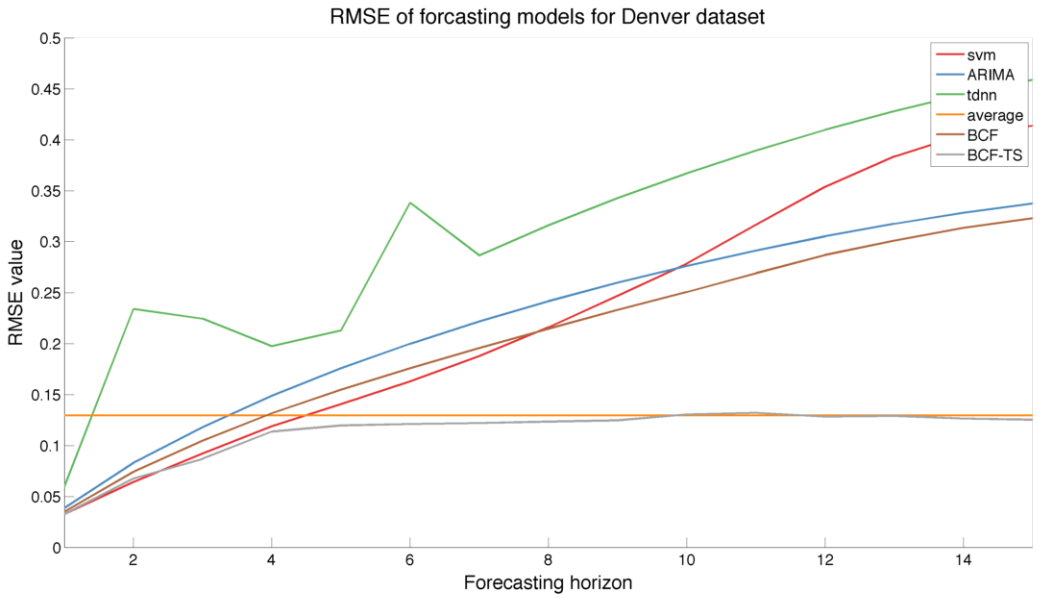


Figure 4.6: Denver dataset: RMSE of forecasting for each model vs forecasting horizon.

modifications to BCF allow BCF-TS to perform better in almost every forecasting horizon.

Since our changes to BCF-TS deal with multi-step forecasts, it is not surprising that most of our improvement over BCF occurs multiple time steps into the future and generally remains multiple time steps into the horizon. We especially focus on forecasting horizons 3 to 6 which are generally 5 to 10% for both Brown and MERL datasets with the Denver dataset much higher. Improvement can be easily seen in Figure 4.6 as BCF-TS trends to the best model (historic average) when all other models begin to forecast poorly.

We feel BCF-TS demonstrates empirically a general improvement over BCF. In our dataset it almost always improves the forecasting accuracy with no additional expert knowledge required for training.

Plots for the results of MASE values for all datasets along with a percent improvement plot for MASE values similar to Figure 4.8 are included in the appendix (**??**, Figure A.3, Figure A.2 and, Figure A.4).

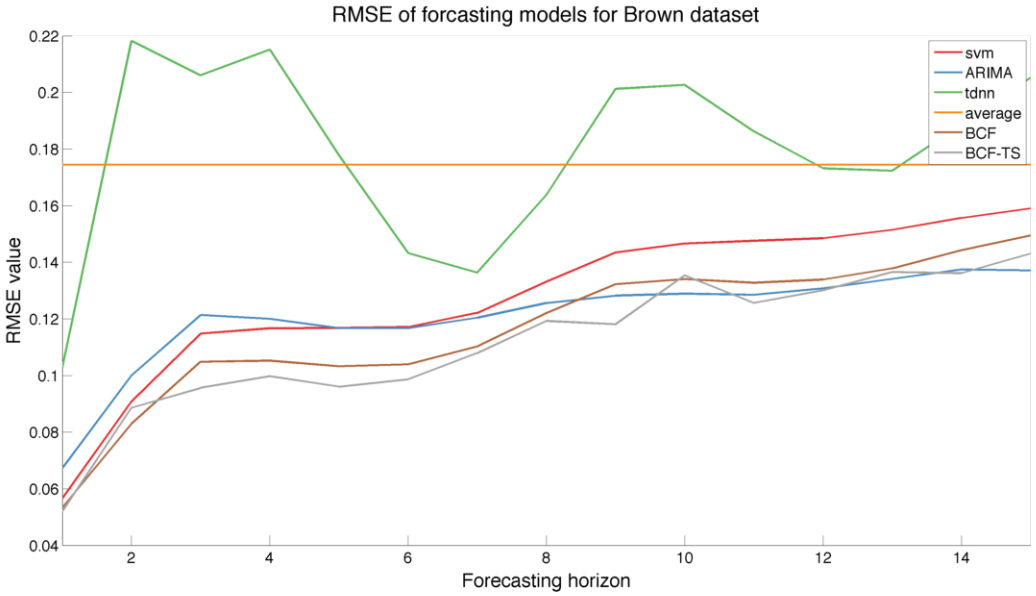


Figure 4.7: Brown dataset: RMSE of forecasting for each model vs forecasting horizon.

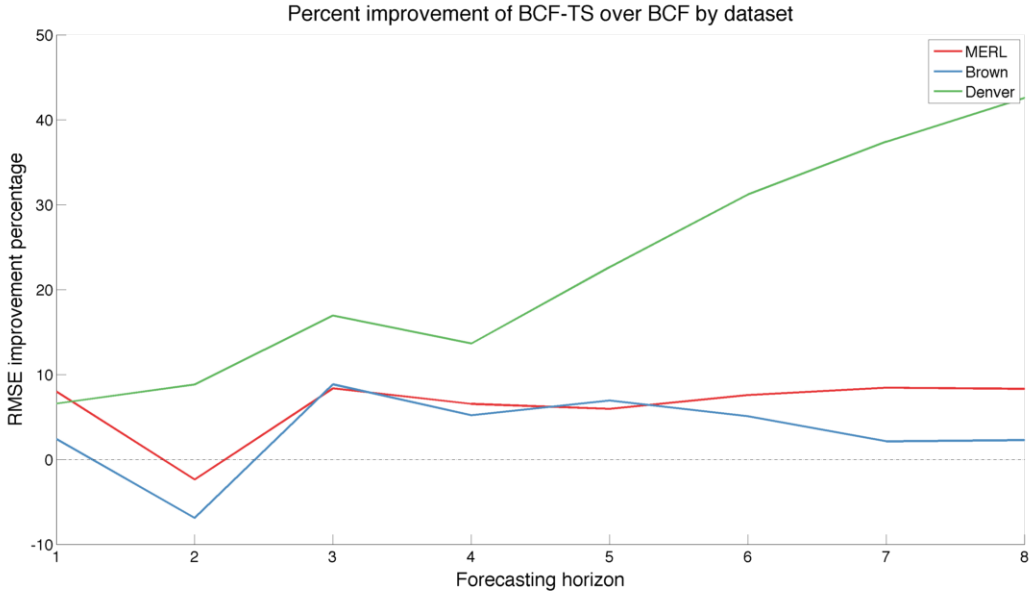


Figure 4.8: Improvement percentage of RMSE values of BCF-TS compared to BCF.

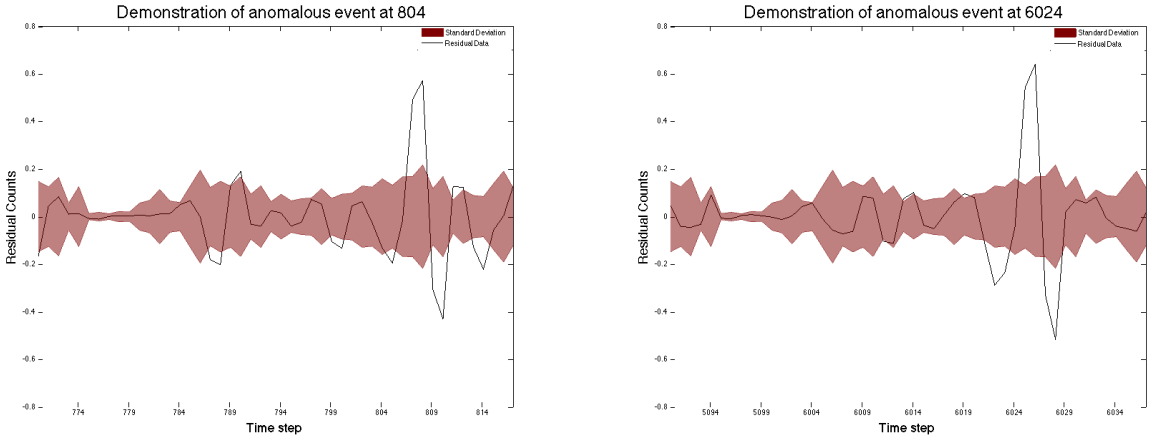
CHAPTER 5

ACTIVITY RECOGNITION ENSEMBLE FORECASTING

In the past chapter, we showed that a few changes can lead to some significant improvements in multi-step forecasting accuracy when ensemble Bayesian combined forecasting is applied to our traffic datasets. This chapter initially investigates BCF further by exposing some potential problem scenarios. We look at some related work into potentially solving these problems through the identification and modeling of activities and finally introduce a new ensemble forecasting technique which solves many of these problems.

5.1 The need for another ensemble forecaster

When analyzing the residual datasets of these ensemble forecasts, we noticed a fundamental problem with our forecaster. In the residual forecasts of BCF and many of the forecasting algorithms, the resulting residual forecast would still have many repeated misforecasts that could not be explained by random noise. As alluded to in Chapter 1, these repetitive mis-



(a) (b)

Figure 5.1: Two similar events occurring at different time in the same Denver residual dataset.

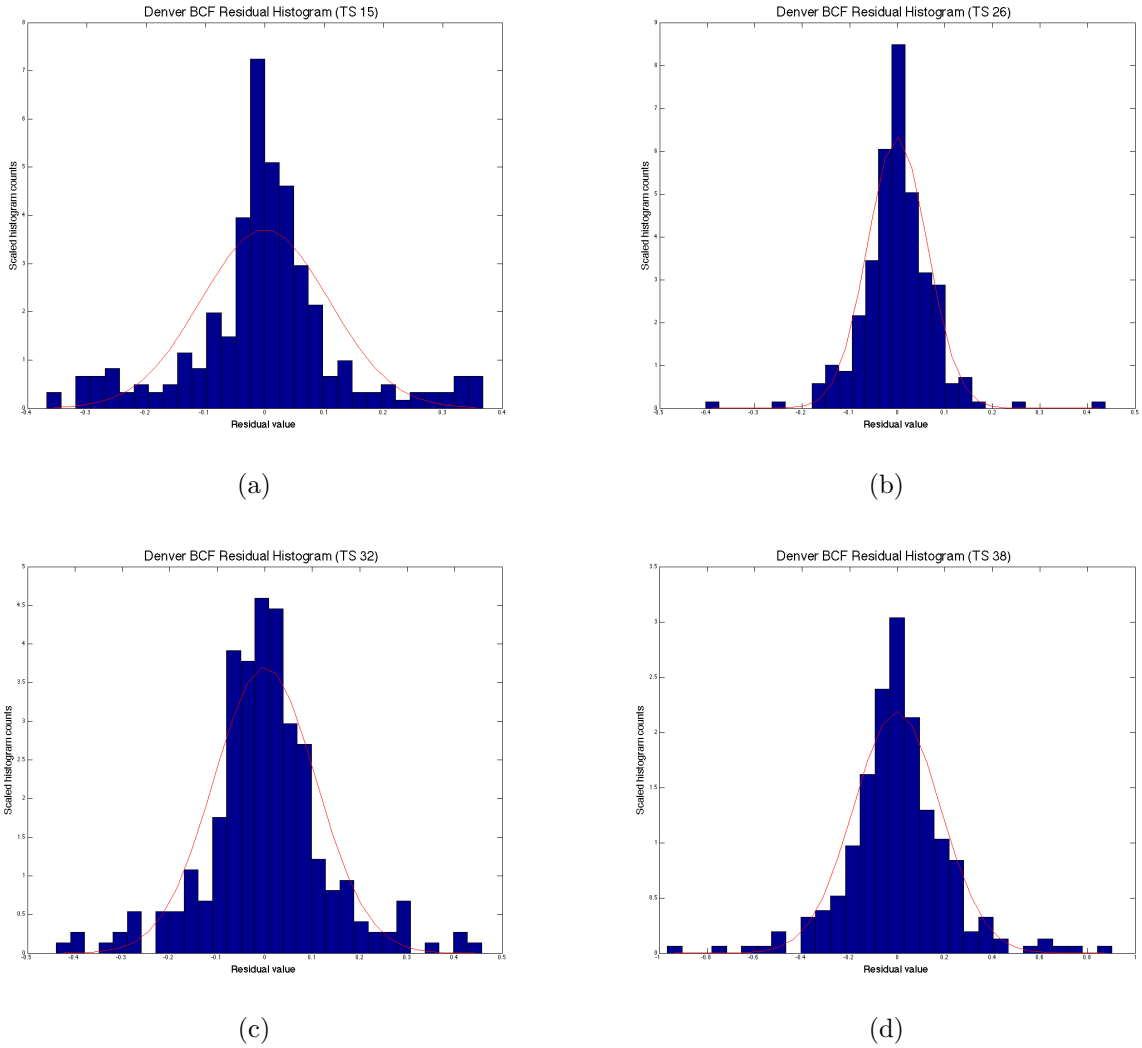


Figure 5.2: Scaled histogram of IBCF residual values for the Denver traffic dataset at various daily time steps. The red line is the corresponding best fit Normal distribution. Notice how in all plots there exist data points which exist after the tails of the Normal distributions have approached zero.

forecasts may come from large human controlled scheduled events (such as sporting events, public celebrations, or in the case of buildings - meetings) or they may be from uncontrolled and unscheduled events (such as weather or traffic accidents).

Figure 5.1 shows two of these events occurring. Such an event clearly occurs outside the normal behavior noise of our data. The light red region in this image represents the one standard deviation boundary for the residual data. From Chapter 4, we know that BCF

residual data tends to be normally distributed per daily time step. The residual datasets from each time step pass the one-sample Kolmogorov-Smirnov [55, 56] test for normality ($p \geq 0.1$). Assuming this data is truly normally distributed then the odds of getting even one such datapoint outside the $\pm 3\sigma$ should occur roughly 1 in every million data points. Yet, in our Denver traffic data, we had 22 such instances in the testing dataset alone ($N \approx 12000$).

Figure 5.2 the distribution of our datasets graphically. In this figure, a histogram of the data from four different fixed daily time steps from the Denver traffic dataset, overlaid with the corresponding best fit normal distribution. In each image the same pattern appears. Residual values around the mean and on the tails tend to occur with a greater frequency than would be suggested by a normal distribution. Values around the mean are not the problem. These values correspond to accurate forecasts. Values around the tail however are problematic and are further evidence of large events being one of the causes of noise in our residual data.

The unlikely existence of these tail events combined with evidence that the overall residual time series behaves as a white noise Gaussian process provides evidence that these events are likely not due to natural noise and instead due to other factors. The remainder of this chapter discusses ways to identify and model these anomalous events along with providing an algorithm for using these events to improve our forecasting algorithm.

5.2 Bayesian combined forecasting with activity and anomaly modeling (ABCF)

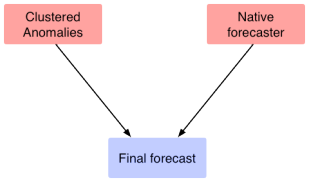


Figure 5.3: Extremely high level overview of our approach

As outlined in the introduction, we return here to a brief discussion of our approach. The structure of non-Gaussian anomalies in the residual datasets of our forecasts leads us to explore if models of these anomalies can be used to improve our forecasts. We propose a hybrid ensemble approach which can be applied to any forecaster and may lead to improved results.

A high level view of our approach is detailed in Figure 5.3. Given any trained forecaster, we extract a set of residual data. Searching this residual

dataset, we next attempt to extract and model the most representative set of anomalous events. From Figure 5.1, we know that for our traffic systems - and we believe for many other traffic systems with repeated events - these events are not unique and are likely to repeat with similar behavior. These anomalous events are then used as the basis of an ensemble forecaster combined with a zero mean background model which represents the native forecaster.

In Section 5.8 we introduce a recursive ensemble forecaster which uses clustered anomalies as inputs and works on the residual dataset of any given native forecaster. Finally, using both the anomaly ensemble inputs and the native forecast, we combine the results to produce a final forecast.

5.3 Background literature on activity recognition and anomaly detection

To construct our approach we need to identify and model these anomalous events. Considerable work has been done in the field of time series anomaly detection; however this work is typically limited to anomaly detection and classification. We were unable to find many academic references using anomaly modeling techniques for improved forecasting. We believe that this limited amount of anomaly modeling work is due to the nature of anomalies; either they are random and difficult to predict as is the case with stock market anomalies [57, 58] or they require some immediate attention and thus forecasting is an inappropriate response to the anomaly. Such a situation arises frequently in network data monitoring. Detection and classification of network anomalies help to determine potential network attacks [59, 60] and allow for preventive measures to take place, but forecasting network traffic during these anomalies is of little utility.

Our anomalies tend to be repetitive and multiple time steps in length. Data of this form is very closely related to another field of time series data analysis - that of activity recognition. Due to the broad nature of this term and the breadth of research on anomaly detection, we briefly discuss some of the ways in which time series anomaly detection and activity recognition has been utilized in the past as a way to familiarize the reader with a

discussion of the literature. This review is not meant to be a complete list of all works in these fields, but instead gives an overview of the types of work done in this field so that we may better contrast previous work with our approach.

Anomaly Detection

From Eamon Keogh, a prominent researcher in time-series anomaly detection, a reading or series of readings is anomalous in a time-series if the

> "frequency of occurrences differed substantially from that expected, given previously seen data. [61]"

A common method of anomaly detection is through the use of tools to assist in visual identification [62–64]. Tools have been extensively developed for network anomaly detection. Such tools allow network administrators and researchers to quickly identify and potentially classify anomalies in the form of certain network attacks. This is done by giving providing graphs, and overlay visualizations which make the identification of patterns more apparent. Through the use of these tools, network administrators are able to quickly respond to various attacks and minimize the potential damage to the network. Visual assistance tools for anomaly detection do not provide modeling and forecasting of anomalies and thus are of little utility for utilizing anomalies to improve traffic system forecasting.

Another common technique for anomaly detection within time series data is known as change point (or step detection). Originally developed for statistical quality control, the cumulative sum control chart (CUSUM) [65] is a classic algorithm for detecting changes in the mean of a time series. It involves the calculation of a cumulative sum of the weighted observations. When this sum exceeds a certain threshold value, a change in value is declared.

The field of change point detection is quite heavily researched. Researchers have developed algorithms for most types of data and computational scenarios. There are algorithms to detect changes in time series mean, changes in variance and changes in distribution from the exponential family [66]. These algorithms can operate offline, online [59], top down,

bottom up and globally. Excellent summaries of current state of the art in change point detection spanning all major formulations (Bayesian, minimax, and generalized Bayesian) are found in [67, 68].

For some time after the original CUSUM change point method, research moved to more parametric stochastic models. Researchers would look for statistical changes in a time series that fit another distribution or family of distributions, for example Adams and MacKay's paper on Bayesian Online Change Point Detection [69]. Such methods are powerful when the change point distributions are known a priori, however we do not assume such knowledge.

Another more recent paper from Liu et al. [70] discusses an entirely different change point detection technique which relies on the ratio between the distribution of some samples at time t and some further samples at time $t+n$. Through ratio estimation, researchers can use a dissimilarity measurement on non parametric models of the data to estimate change points. The power of this technique comes from directly estimation the ratio of probability densities and not the densities themselves. From this density estimation, the rational is that knowledge of the two densities implies the ratio, but the ratio does not uniquely imply the densities and thus the ratio may be easier to estimate while still giving knowledge of the change point.

The problem with general change point techniques for our data is that our data distributions change constantly throughout the day. Our MERL dataset shows large amounts of activity during lunch time and work start and end times. We are interested in those instances where activities are anomalous. Our technique uses a combination of peak finding and time of day residual CUSUM. A detailed description of our technique and results will be shown later.

Individual Activity Modeling and Recognition

Work in activity recognition has focused on recognizing either individual activities or group activities. In this section we describe many of the individual activity recognition techniques. One common type of individual activity recognition is from wearable sensors such as ac-

celerometers or RFID tag readers. This type of work is almost always supervised and the goal is to map sensor readings to a comprehensible activity such as dish washing or tooth brushing [71, 72]. While some of this has potential applications to our goals, much of it is not applicable as the focus is typically on recognizing activities from fully labeled datasets. Authors from this field have used many of the standard machine learning models: decision trees [72], support vector machines [72–74], naive Bayes [72, 74], nearest neighbor [72, 74], and hidden Markov Models (HMM) [71, 75]. Comparisons amongst models have shown that performance is data dependent and that no one model appears to be best for all types of activities [72, 74]

Huynh [76] used a naive Bayes classifier in a different way for wearable sensor individual activity recognition. Instead of using it to describe activity, it was used as a dimensionality reduction technique; the results of which were the basis for a dictionary in latent Dirichlet allocation [77]. The topics generated from latent Dirichlet allocation are then clustered using k-means. Each of these clusters represents a single activity. This clustering approach proved effective for the recognition of repeated activities throughout the day, but due to its reliance of a fixed ratio of latent Dirichlet allocation projected topics, it is likely that recognizing combinations of activities will prove problematic.

To account for activities of varying time lengths, probabilistic suffix trees [78] have shown to be an effective model for activities. Trees are trained using all sequential subsets of an input sequence and a total model is then created from the set of trees using AdaBoost [79]. The performance level of suffix trees seems to be highly noise dependent. [80] compared HMMs with suffix trees and found that suffix trees out performed HMMs when the data was without noise, but as the noise increased HMMs performed increasingly better, eventually surpassing the performance of suffix trees.

Group Activity Modeling and Recognition

There are a limited number of publications that exist on group activity modeling recognition using a large number of sensors. Within this problem domain the challenges to solve are

different due to the type of data collected and due to group activities typically occurring over multiple sensors. The work within this domain that is most similar to our work is from Mitsubishi Electronic Research Laboratories (the creators of the MERL dataset) [42, 47]. The goal of such group activity work is to utilize a set of sensors to describe activities or events which effect many individuals within the environment. As expected, these group event models are typically used to describe larger or longer movements within the environment. For example, such modeled events may be holiday movement vs workday movement or general flow through out a building based on sensor topology [41, 43].

HMMs have been used as a model for learned activities. These models are used to build a tree [46, 47] with each level described by a model with a different number of hidden nodes, meaning that model accuracy is roughly correlated to tree depth. At the top of the tree are simple models used to describe gross activities. The leaves of the tree are highly complex models describing specific activities. This tree structure has the advantage of being computationally efficient while maintaining accuracy on par with other techniques based on clustering of HMMs [81].

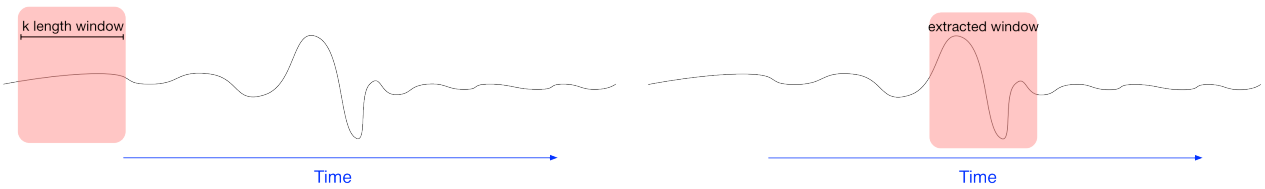
In a method similar to the HMM tree, work has been done to create a hierarchy of fixed filters based on possible sensor topologies [43]. At each level of the hierarchy, the number of sensors and the amount of time history increases. The probability of occurrence of each fixed arrangement is then computed when all levels are created, the resulting model represents the total classifier. The usage of fixed sensor topologies is highly environment dependent and the fixed time lengths with each level of the hierarchy are likely too restraining for the types of activities we expect to observe.

From the results of all activity recognition papers, it appears that approaches which show the most promise tend to use a model which allows comparison of inputs with various time lengths. Also, hierarchical techniques tend to perform better than multiple models of equal complexity. In defense of these general observations is the work of Huynh [82] who found that empirically for his problem, there is not a single feature or time window of past history

that will perform best for all activities. Instead each activity is best modeled by a set of features and length of time unique to that activity. Huynh postulates that his finding is true of most activity recognition problems. Our approach, which will be described in detail in the next section is to represent activities as a mixture of Gaussians. While for this work we use fixed length activities, our ensemble forecaster (described in Section 5.7) is derived to work with activities each which may have any length independent of the other modeled activities. Modeling mixed length activities is recommended for future work.

5.4 Anomaly extraction and representation

As we showed earlier in this section with Figure 5.2, the tails of the best fit Gaussian for the BCF residual random variable tend to have larger than expected likelihoods. These unlikely residuals do not appear to be completely random. From a review of the literature, we know that there are numerous methods to extract and model this data. For our work, we propose a simple extraction technique and then model the data according to a time-series mixture of Gaussians. This method allows us to impart a probabilistic behavior to our anomaly models. Such behavior is essential to our eventual forecasting technique.



(a) (b)

Figure 5.4: Demonstration of a fixed width sliding window looking for locally maximal deviations from background behavior.

Sliding window data extraction

Given a time series of residual data, we extract the top $k\%$ of the maximum residual data as measured by a fixed length sliding window. Figure 5.4 demonstrates visually our method of potential candidate residual data segments to model. In this example, the fixed length

window is slid along a time series and windows with large total residual deviation are selected. These windows are then centered around the largest peak within the window. This is done to ensure an easier time clustering as similar extracted residual segments will typically line up with each other.

Algorithm 1 Algorithm for candidate data extraction

```
#assume we have access to residual_data as a one dimensional array

extracted_lengths = []
final_data = []

for t = 0; t <= residual_data.length - window_width; t+ = window_width do
    window_sum = sum(abs(residual_data[t:t+ window_width]))
    extracted_lengths.append((window_sum, t))
end for

#sort the extracted lengths by the window_sum dimension descending
extracted_lengths = sort(extracted_lengths, dim1, order = "descending")
num_to_extract = ceil(k * extracted_lengths.length)

#Center all the extracted residuals by local peaks
for t = 0; t < num_to_extract; t + + do
    dind = extracted_lengths[t][1]
    #Max peak on local window
    mp = find_max_peak(abs(residual_data[dind:dind + window_width]))

    #Recenter the data around the local peak
    final_data.append(residual_data[mp - window_width / 2: mp + window_width / 2])
end for
```

Algorithm 1 precisely describes our approach. The input parameter k is chosen empirically for each dataset and underlying forecasting model. The effects of this parameter k are discussed later. A sample of the top 10% extracted residuals for the MERL dataset is displayed in Figure 5.5.

On additional possible technique for data extraction is sparse dictionary coding. This technique finds a set of primitives (in this case small representations of the time series residual data) which can be used to best approximate the original dataset. We believe it likely that primitives with the largest residual values would most likely correspond to the anomalies we

would like to find. This technique has been extensively to find primitives activations on time series EMG data [83] and for image reconstructions [84].

For this work however, we believe that our constant length anomaly extraction is sufficient to show the strength of our combined forecasting approach.

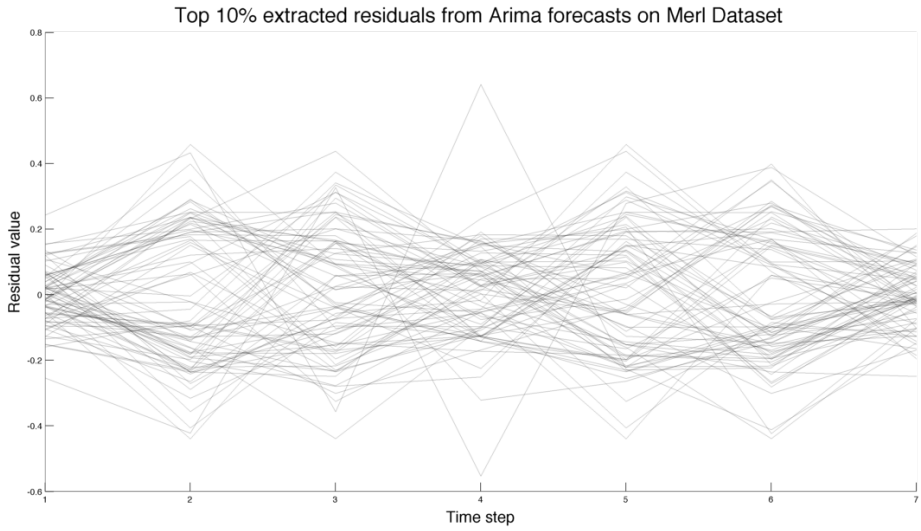


Figure 5.5: Extracted residuals from the MERL dataset using the sliding window extraction method with window length of 7

5.5 Time series mixture of Gaussians derivation

A mixture of Gaussians is a strongly supported stochastic data clustering technique used in activity recognition. This technique models a set of data by positing that each data point (x_i) is generated from one of K Gaussians. Borrowing partially the notation of Andrew Ng [85], a mixture of Gaussians is described mathematically by specifying a joint distribution $p(x_i, z_i) = p(x_i|z_i)p(z_i)$. z_i is multinomial on parameter ϕ ($\phi_j \geq 0$, $\sum_{j=1}^{K} \phi_j = 1$ and $p(z_i = j)$). Each $x_i|z_i = j \sim \mathcal{N}(\mu_j, \Sigma_j)$. From this derivation, the z_i's are latent variables which makes the estimation of the Gaussian models more difficult. The result of mixture of Gaussian training is to produce a soft clustering of every datapoint x_i to each trained Gaussian distribution.

Traditionally, a mixture of Gaussians is for data vectors with no time element, but some research has been done on using a mixture of Gaussians for time series data clustering. In one paper [86], the authors create a design matrix to construct a delayed embedding of the original time series. Each row in the matrix corresponds to a fixed length vector selected from the input time series. All of the rows together contain all data from the time series with each row over containing overlapping data with one new time series element. From this embedding, the authors compute a mixture of Gaussians to approximately reconstruct the time series. In another paper [87] Gaussian mixture models are used to mix the weights of an autoregressive model in an attempt to reconstruct the original time series.

Here we introduce another technique for modeling time series data were we assume the presence of multiple short (typically fewer than 10 time steps) residual time series. These time series are assumed to be derived from a set of K unique time series. In each of these time series, we model the time steps as Gaussians, each with their own unique mean and covariance. For our work, the input time series are those candidate extracted anomalies from the previous section (i.e. we are attempting to model and cluster data from Figure 5.5). The following is our derivation of this approach.

The goal of mixture of Gaussians is to find a set of models which will maximize the log likelihood of the parameters of some models to the dataset. Given dataset (in this case a dataset of residuals) $\{r^{(i)}\}$ we maximize

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{M} log\{p(r^{(i)}; \phi, \mu, \Sigma)\} \tag{5.1}$$

where M is the total number of time series instances. In this equation, the $r^{(i)}$'s are each time series slices from the candidate residuals. The (i) represents the ith residual candidate time series - we use the (i) notation to distinguish from the exponent operator. For this work, each $r^{(i)}$ is of fixed length N. ϕ, μ, and Σ are the parameters to optimize for each time series Gaussian. Each $r^{(i)}$ is roughly analogous to x_i in the typical derivation of a mixture of gaussians with the difference here that each $r^{(i)}$ is N readings in length.

As in traditional mixture of Gaussians formulation, we introduce a latent variable $z^{(i)}$ which is multinomial on parameter ϕ ($\phi_j \geq 0$, $\sum_{j=1}^{K} \phi_j = 1$ and $p(z_i = j)$). With this latent variable $z^{(i)}$ we now maximize the likelihood of the joint distribution of r and z. This equation is given as

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{M} \log \sum_{k=1}^{K} p(r^{(i)}, z^{(i)} = k; \phi, \mu, \Sigma) \tag{5.2}$$

In the traditional mixture of Gaussians algorithm each model is ostensibly a Gaussian which may be multivariate on $x^{(i)}$. One option to model time series data using a mixture of Gaussians is to use each residual input window as multi dimensional training data. A set of K multi variate Gaussians would then be used to fit the training set. This type of model does not work with our activity time series forecaster introduced later. To make this algorithm work with time series data, we define the models instead by

$$p(r^{(i)}|z^{(i)} = k; \mu, \Sigma) = \prod_{n=1}^{N} p(r_n^{(i)}; \mu_n, \Sigma_n). \tag{5.3}$$

In this equation, N is the length of each time series instance. Each $p(r_n^{(i)}; \mu_n, \Sigma_n)$ is computed by a different Gaussian at time offset n. Thus our model for each time series is N independent Gaussians.

Finally, to make our notation a bit easier we define a variable $w_k^{(i)}$.

$$w_k^{(i)} = p(z^{(i)} = k|r^{(i)}; \mu, \Sigma) \tag{5.4}$$

$w_k^{(i)}$ represents the probability of latent variable $z^{(i)}$ taking the value j from our set of models defined in equation 5.3

Expectation maximization (EM) is a common technique to solve maximization problems of this kind. Here we briefly derive the EM algorithm for our time series derivation of mixture of Gaussians. Combining our equations so far, we can write the likelihood of the time series model as

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{M} \log \sum_{k=1}^{K} w_k^{(i)} \frac{p(r^{(i)}, z^{(i)} = k; \phi, \mu, \Sigma)}{w_k^{(i)}} \tag{5.5}$$

This is the equation we want to maximize. The EM algorithm is computed in two steps, the E step which gives a lower bound on the equation we are trying to maximize and then the M step which maximizes for the current lower bound.

E-Step

The E-step hardly changes from the traditional EM mixture of Gaussians algorithm. We simply need to calculate

$$w_k^{(i)} = p(z^{(i)} = k|r^{(i)}). \tag{5.6}$$

This can be calculated directly from the data by applying Bayes rule.

$$w_k^{(i)} = p(z^{(i)} = k|r^{(i)}) = \frac{p(r^{(i)}|z^{(i)} = k)p(z^{(i)} = k)}{\sum_{k=1}^{K} p(r^{(i)}|z^{(i)} = k)p(z^{(i)} = k)} \tag{5.7}$$

M-Step

For the maximization step it is assumed that we know the values of $w_k^{(i)}$. Thus, we need to maximize equation 5.5 with respect to μ, Σ, and ϕ. However there is one more simplifying step which needs to be made prior to direct maximization. This can be made according to Jensen's inequality, that is equation 5.5 may be written as

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{M} \sum_{k=1}^{K} w_k^{(i)} \log \frac{p(r^{(i)}|z^{(i)} = k; \mu, \Sigma)p(z^{(i)} = k; \phi)}{w_k^{(i)}} \tag{5.8}$$

and still converge using the EM algorithm.

Combining equations 5.8 and 5.3 gives the following log likelihood

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{M} \sum_{k=1}^{K} w_k^{(i)} [\log \frac{p(z^{(i)} = k)}{w_k} + \sum_{n=1}^{N} \log \frac{p(r^{(i)}; \mu_{k,n}, \Sigma_{k,n})}{w_k}] \tag{5.9}$$

Maximizing 5.9 with respect to ϕ_k, $\mu_{k,n}$ and $\Sigma_{k,n}$ yields the following update rules for each parameter in the M step.

$$\phi_k = \frac{1}{M} \sum_{i=1}^{M} w_k^{(i)} \tag{5.10}$$

$$\mu_{k,n} = \frac{\sum_{i=1}^{M} w_k^{(i)} r_n^{(i)}}{\sum_{i=1}^{M} w_k^{(i)}} \tag{5.11}$$

$$\Sigma_{k,n} = \frac{\sum_{i=1}^{M} w_k^{(i)} (r^{(i)} - \mu_{k,n})(r^{(i)} - \mu_{k,n})^{\mathrm{T}}}{\sum_{i=1}^{M} w_k^{(i)}} \tag{5.12}$$

5.6 Selecting the number of clusters

Determining an appropriate number of clusters is an important task in cluster analysis. Because of its importance for any clustering technique, the literature on this subject is extensive. While there is no universal definition for determining the optimal number of clusters, there is a general similarity for most cluster analysis techniques. That is, grouping objects (activity residuals for our dataset) which are as similar as possible into sets, while keeping the model representing the clusters as dissimilar as possible. A full discussion of all of these approaches is too broad for our work. Here we will discuss a few common approaches and describe the approach we take to clustering.

Elbow

Developed by Robert Thorndike [88] in 1953, this approach looks to plot the inter-cluster sum of squared error verses the number of clusters. In many clustering applications, the average inter-cluster error will begin to drop significantly and then begin to asymptotically decrease. The number of clusters is chosen at this "elbow." Figure 5.6 demonstrates a this elbow. In the image, the number of clusters selected is 2. A common problem with this approach is that the elbow can not always be unambiguously identified.

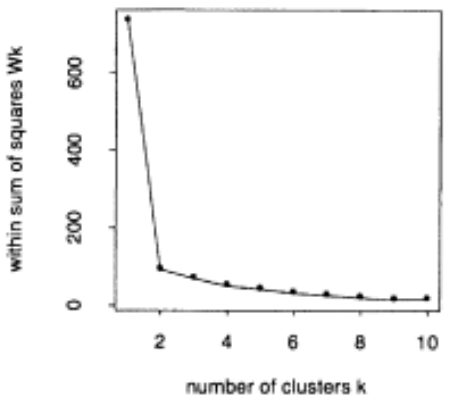


Figure 5.6: Demonstration of the elbow clustering selection [ref].

Information criterion

An information based approach to selecting the number of clusters is to use either the Akaike information criterion [24] or the Bayesian information criterion [89]. There exist

other information criterions, but they do not have the ubiquity of the prior two criterions. Both the Akaike and the Bayesian criterion seek to determine the number of clusters by imposing a tradeoff between the goodness of fit and the number of model parameters. For our work, goodness of fit can be measured by the log likelihood of the data to the Gaussian clusters and the number of model parameters is simply the number of models.

Cross-validation

TODO WRITE ON CROSS-VALIDATION METHOD

Silhouette

Silhouette [90] scores are another method of selecting the number of clusters. A silhouette score balances the tradeoff between the average dissimilarity between all data points for a given cluster and the average dissimilarity to the data points of the next nearest cluster. The silhouette score is defined as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}. \tag{5.13}$$

$a(i)$ is the average dissimilarity between data point i and all other points within the cluster i is assigned. $b(i)$ is the average dissimilarity between data point i and all other points within the nearest cluster that does not contain i. The optimal number of clusters is then chosen by finding the maximum silhouette score amongst multiple possible values of K. Because mixture of Gaussians is a soft cluster assignment algorithm (each point has a likelihood to be assigned to each cluster instead of belonging to one single cluster), we consider the assignment of each point to be only the Gaussian from which it is most likely modeled.

The best method for selecting the optimal number of clusters is an open problem. Different datasets and different clustering methods may require different criterions to determine the optimal number of clusters. Since our approach is mostly unsupervised, we do not have the luxury of determining what criterion works best for our data.

The doctoral thesis [91] of Yan broached the topic of best clustering criterion and showed significant variation between clustering criterions. Of these criterions, silhouette based cluster selection was consistently one of the best selection criterions. Given the variability in selection techniques and combined with Yan's findings, we feel confident that Silhouette scores provide an adequate cluster selection criterion.

5.7 Sample representative clusters and a discussion about them

This section visually demonstrates a few of the clusters extracted from our datasets using the time series mixture of Gaussian's approach. These residual clusters were chosen from each dataset and from different base forecasting approaches. The images shown here are from a mix of base forecasters without regard to the forecaster's accuracy. Such clusters have been generated for every combination of dataset and forecaster residuals.

Figure 5.7 is a set of clusters from a residual time series of MERL dataset extracted using the SVM forecaster. The residuals are from a time series forecasting horizon of 20 minutes (two time steps). The extracted window length is 70 minutes (seven time steps). These clusters were taken from the top 10% of the MERL data residuals.

Figure 5.8 is the set of clusters from the Brown dataset extracted from the ARIMA forecaster. The clusters from the dataset were taken from the top 10% of the Brown data residuals. Notice how these clusters, unlike the clusters from Figure 5.7 do not begin nor end with a residual value of zero. This likely means that the true residuals are longer in length than those displayed here. However, our method of silhouette score maximization returned this set as the best. Most likely this is due to non-consistent residual behavior if the extracted residual window is increased.

Figure 5.9 displays the set of modeled clusters from the Denver dataset extracted from the TDNN forecaster. These clusters were taken from the top 10% of the forecaster's residuals. The time series within the Denver dataset correspond to 30 minute intervals. Thus, the clusters shown in Figure 5.9 demonstrate events that occur over at least 3 hours (six time steps). Also, some of the clusters begin or end with a 0 residual, but others drift higher or

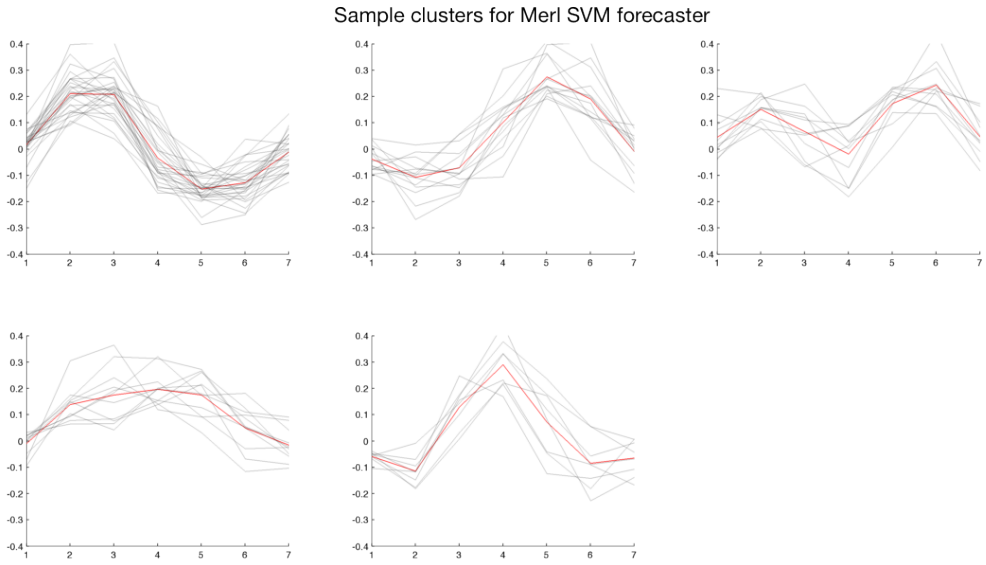


Figure 5.7: Extracted residuals from the MERL dataset. Data was taken from the top 10% of the SVM forecaster's residuals. The red line in the center of the clusters represents the cluster average.

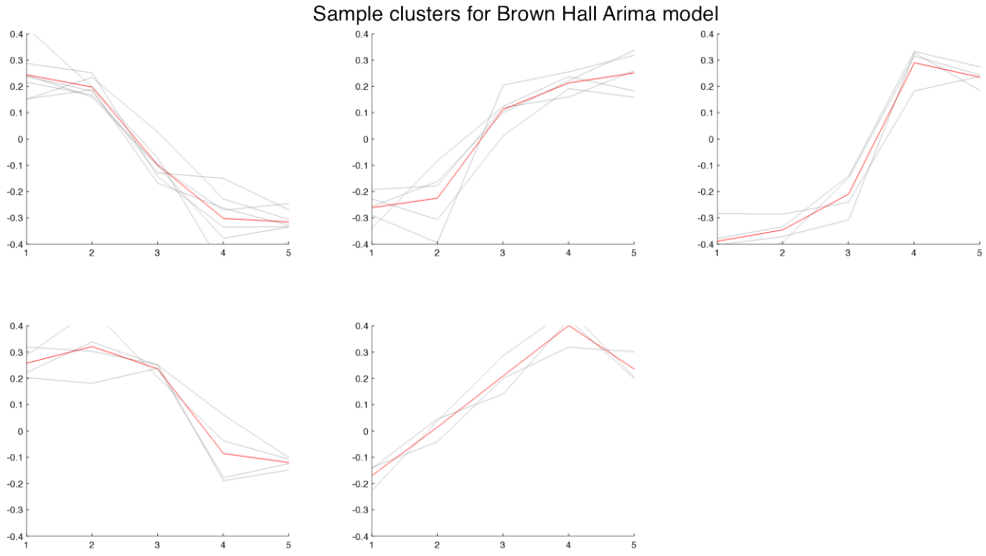


Figure 5.8: Extracted residuals from the Brown dataset. Data was taken from the top 10% of the ARIMA forecaster's residuals. The red line in the center of the clusters represents the cluster average.

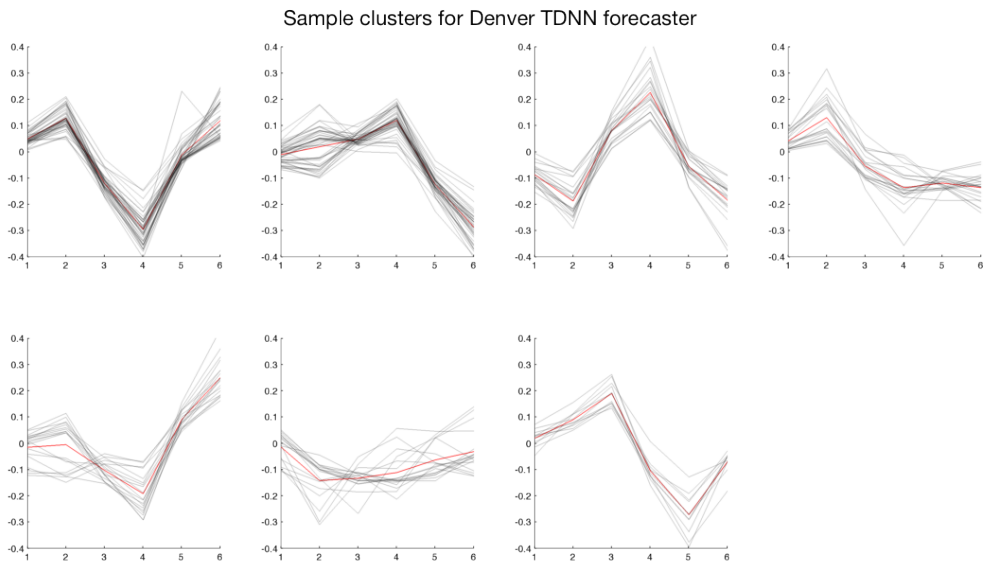


Figure 5.9: Extracted residuals from the Denver dataset. Data was taken from the top 10% of the TDNN forecaster's residuals.

lower. We believe this demonstrates events which may not have finished during the duration of the clusters demonstrated here. Ideally we would like to have clusters long enough to fit the entire event, but short enough that they don't contain "normal" behavior. Clusterings such as this demonstrate a scenario where modeling these clusters through a non-fixed length technique may lead to better results.

Also, notice how some of the clusters above show certain member activities which are somewhat dissimilar from the red cluster line. While we didn't implement it in this work, it is certainly feasible to extract outlier (or dissimilar) activities from the potential pool of activities and rerun the mixture of Gaussians algorithm. This sort of outlier extraction would likely lead to slightly better anomaly models.

Finally, we believe that the structure demonstrated from these clusters again leads further credibility to the residual data containing some information about the activity or anomaly that generated the data. These residual clusters do not appear to be simply the product of random noise.

5.8 ABCF derived

Here we present the math behind our Anomaly Bayesian combined forecasting algorithm. Similar to the Bayesian combined forecaster introduced in Chapter 4, our ABCF algorithm is recursive and computationally efficient.

We present the algorithm using two anomaly event models a and b with a background model c. Extending the algorithm to work for any number of algorithms is apparent from this presentation. For our work, the models a and b are modeled anomalies from our time series mixture of Gaussians; however this derivation allows the models to be any of a stochastic time series model such that it is possible to compute $p(x_t|a_t^{(i)})$. Also, the model c is assumed to be a background model which allows for the computation of $p(c_t|x_t)$.

Let $a_t^{(i)}$ = the event that model b is active at time t at index i and $b_t^{(i)}$ = the event that model b is active at time t at index i. At any time t we have one of $a_t^{(i)}$, $b_t^{(i)}$ or c. From this we know that

$$\sum_{i}^{len(a)} p(a_t^{(i)}|x_t,\ldots,x_1) + \sum_{i}^{len(b)} p(b_t^{(i)}|x_t,\ldots,x_1) + p(c_t|x_t,\ldots,x_1). \tag{5.14}$$

The goal of this forecaster is then to find each of $p(a_{t+1}^{(i+1)})$, $p(a_{t+1}^{(i+1)})$, and $p(c_{t+1})$. Given these model likelihoods, we can then produce a final forecast in a way similar to BCF in chapter 4.

We know from Bayes rule

$$p(a_{t+1}^{(i+1)}|x_{t+1},\ldots,x_1) = \frac{p(x_{t+1}|a_{t+1}^{(i+1)},x_t,\ldots,x_1)p(a_{t+1}^{(i+1)}|x_t,\ldots,x_1)}{p(x_{t+1})} \tag{5.15}$$

Thus to find $p(a_{t+1}^{(i+1)}|x_{t+1},\ldots,x_1)$ we need to compute:

- $p(x_{t+1}|a_{t+1}^{(i+1)},x_t,\ldots,x_1)$
- $p(a_{t+1}^{(i+1)}|x_t,\ldots,x_1)$
- $p(x_{t+1})$

Computing $p(x_{t+1})$ is straight forward from the law of total probability

$$\begin{aligned} p(x_{t+1}) = & \sum_{i}^{len(a)-1} p(x_{t+1}|a_{t+1}^{(i+1)}, x_t, \ldots, x_1)p(a_{t+1}^{(i+1)}|x_t, \ldots, x_1)+ \\ & \sum_{i}^{len(b)-1} p(x_{t+1}|b_{t+1}^{(i+1)}, x_t, \ldots, x_1)p(b_{t+1}^{(i+1)}|x_t, \ldots, x_1)+ \\ & p(x_{t+1}|c_{t+1}, x_t, \ldots, x_1)p(c_{t+1}|x_t, \ldots, x_1). \end{aligned} \tag{5.16}$$

The computation for $p(x_{t+1}|a_{t+1}^{(i+1)}, x_t, \ldots, x_1)$ is performed directly by computing the likelihood of a new data point x_{t+1} for model a at offset $i+1$.

Finally, the calculation of $p(a_{t+1}^{(i+1)}|x_t, \ldots, x_1)$ is from an earlier forecast and is what gives our algorithm its recursive nature. We pass the posterior of all values of a_t to the priors of a_{t+1}. Thus $p(a_{t+1}^{(i+1)}|x_t, \ldots, x_1)$ is simply a passed value from the last time step in our algorithm. Specifically we define

$$p(a_{t+1}^{(i+1)}|x_t, \ldots, x_1) = \begin{cases} p(a_t^{(i)}|x_t, \ldots, x_1) & : i < len(a) \\ 0 & : otherwise. \end{cases} \tag{5.17}$$

Thus in the present of an event we pass along the likelihood of that even to the next time step and recompute the posterior of that event. The prior for the start of each event is a user tunable parameter.

$$\begin{aligned} p(a_{t+1}^{(1)}|x_t, \ldots, x_1) = p(a_{t+1}^{(1)}) = \theta_a \\ p(b_{t+1}^{(1)}|x_t, \ldots, x_1) = p(b_{t+1}^{(1)}) = \theta_b \end{aligned} \tag{5.18}$$

5.9 Demonstration of ABCF

Figure 5.10 is a demonstration of ABCF. This figure displays a sample time frame (in blue) from the Denver dataset along with the corresponding region for one standard deviation of noise from a best fit ARIMA forecaster with a horizon of two time steps. The teal line below the graph is the likelihood of the background ARIMA model accurately representing the residual data from the set of clustered anomalies. The remaining tan lines are the

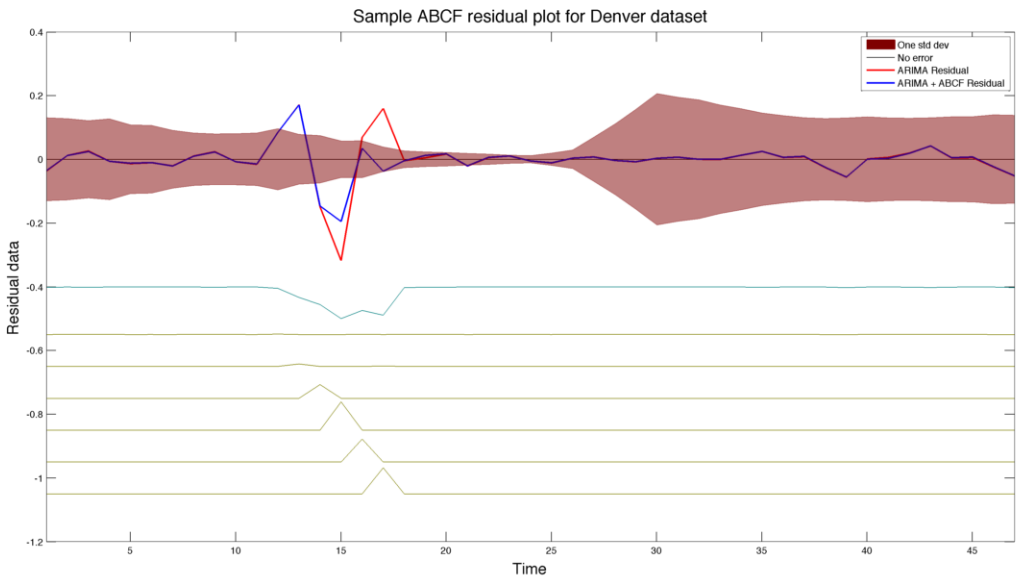


Figure 5.10: Sample residual and the respective probabilities of each time step for the base forecaster and the a single activity model. The teal line represents the likelihood of the background ARIMA model. The tan lines correspond to the likelihood each time step from one of the anomaly clusters.

individual likelihoods of each time step of a given clustered anomaly shown in Figure 5.11

The red line is the resulting ABCF forecast to the residual. Ideally the residual should be completely zero, thus resulting in no forecasting error. At approximately time 10, an anomalous event begins to occur. The ABCF algorithm takes a few time steps to identify this anomaly and determine that the background ARIMA forecaster is no longer accurate. We see this effect by noticing the teal line's reduction in likelihood from approximately time step 12 to time step 17. During this time all other extracted clusters are attempting to fit the anomaly.

The recursive nature of our likelihood time step propagation in the ABCF algorithm is apparent in the tan lines of this sample figure. At time step 13, there is a slight perturbation in the second tan line. This perturbation continues to rise both in value (indicating an increase in likelihood that this cluster is active) and time along the anomaly. Finally at about time step 18, the model is finished. At this point the likelihood of the background

model begins to rise again and eventually the background model (the teal line) is once again the most prevalent model amongst all the extracted anomalies.

Figure 5.10 displays a significant improvement in RMSE-ONAN. While the early stages of the anomaly show no improvement, the residual error during the middle of the anomaly is roughly halved and the residual error at the end of the anomaly is nearly zero. Figure 5.12 shows the final result of this forecast window. The ARIMA forecaster is delayed in its forecast response to the anomaly, but because such an anomaly was commonly seen within this dataset, the ABCF algorithm is able to estimate the anomaly and improve forecasting accuracy during the remainder of the anomaly.

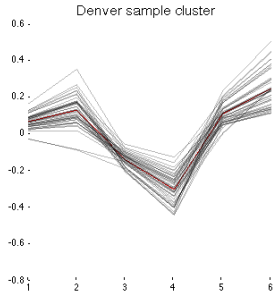


Figure 5.11: Extracted sample clustered anomaly.

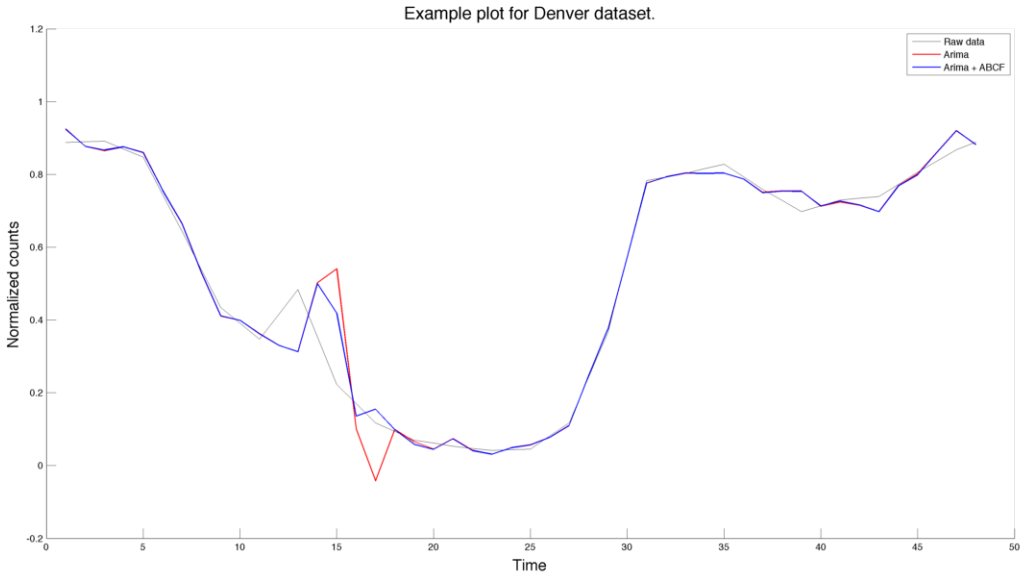


Figure 5.12: Two horizon time step results when applying ABCF to an ARIMA forecaster for a segment of the Denver dataset.

5.10 Results of ABCF

Since the number of potential images to display about the improvements due to ABCF is quite large we provide a few images here to demonstrate our results and put the remaining images in Appendix A for the interested reader.

Figure 5.13 demonstrates the improvements to the RMSE-ONAN metric of applying our ABCF algorithm to the average forecasting algorithm for the Denver dataset. Unsurprisingly we found our largest improvements came with the historic average model. The reason for this is that any activities or consistent deviations from normal behavior show up in the average model. These behaviors are well represented in the residuals and thus make good candidate models for our ABCF model.

Figure 5.14 and Figure 5.15 display the improvement to both the RMSE-ONAN metric and RMSE metric when applying ABCF to an ARIMA forecaster on the MERL dataset. Notice that while the improvement to the RMSE-ONAN metric is significant in Figure 5.14, the improvement to RMSE considerably smaller. This result is expected. ABCF improves fore-

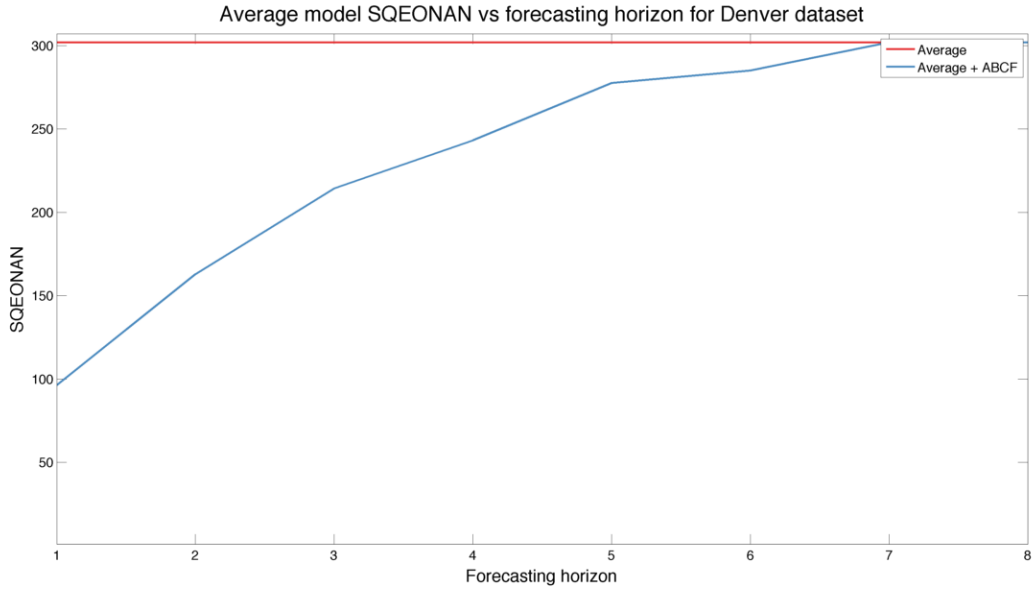


Figure 5.13: RMSE-ONAN vs forecasting horizon for Average model and Average + ABCF on the Denver dataset

casting during the presence of anomalies. Due to the nature of anomalies being sufficiently rare, ABCF may not improve overall forecasting accuracy by a large margin.

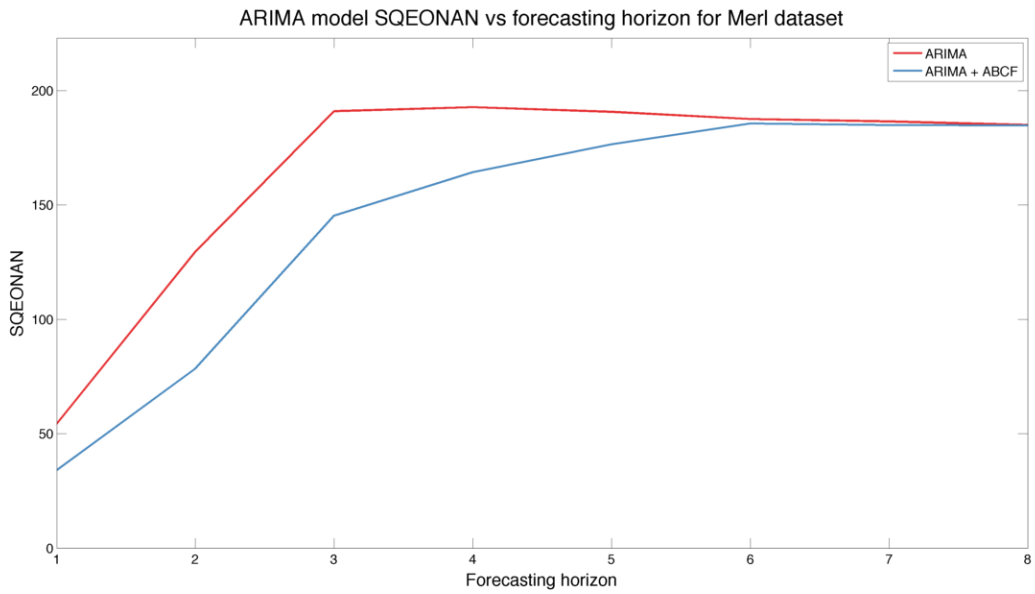


Figure 5.14: RMSE-ONAN vs forecasting horizon for ARIMA model and ARIMA + ABCF on the MERL dataset

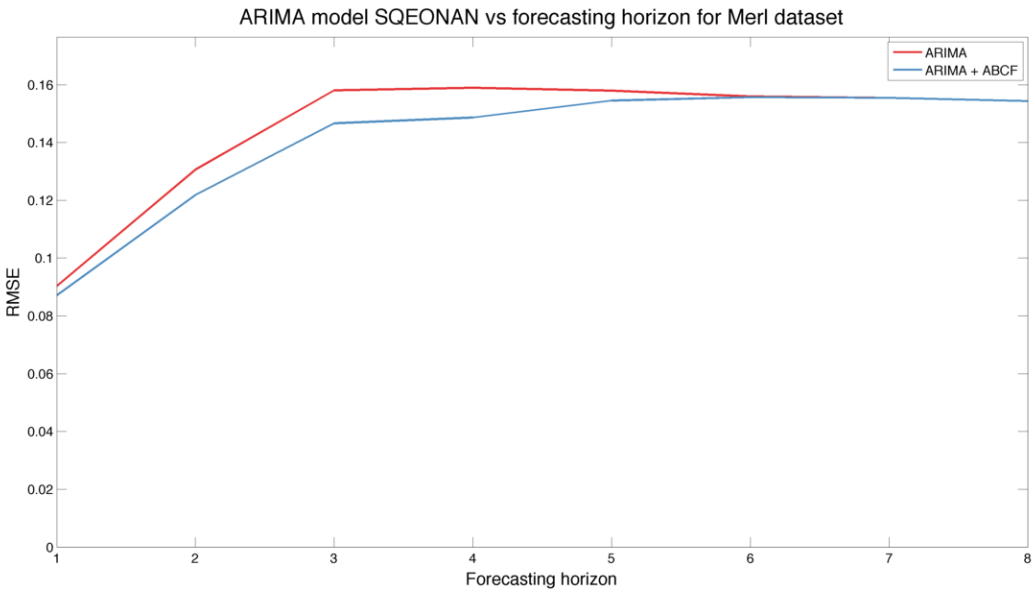


Figure 5.15: RMSE vs forecasting horizon for ARIMA model and ARIMA + ABCF on the MERL dataset

Aggregated Results

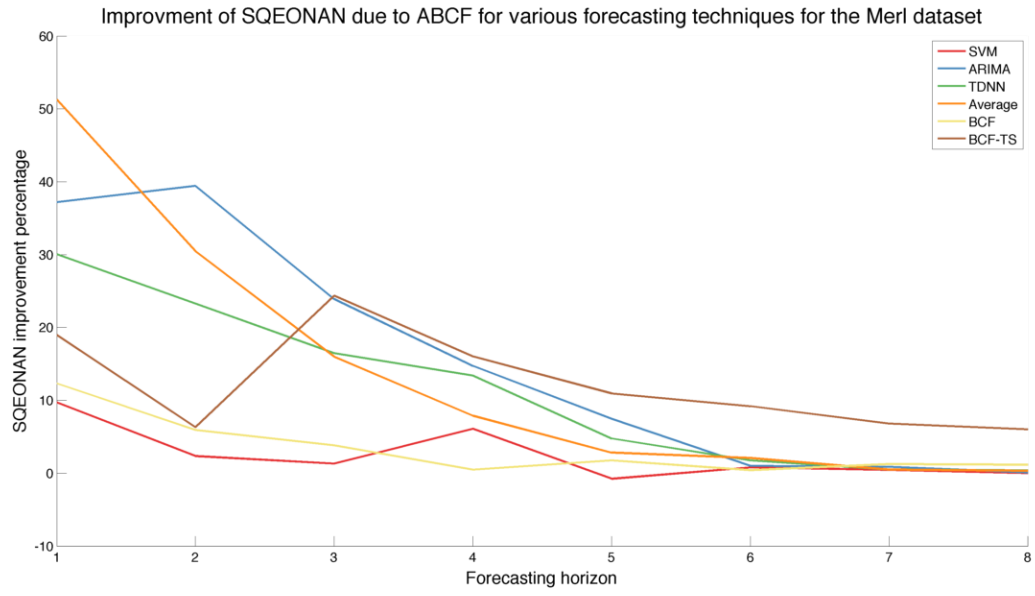


Figure 5.16

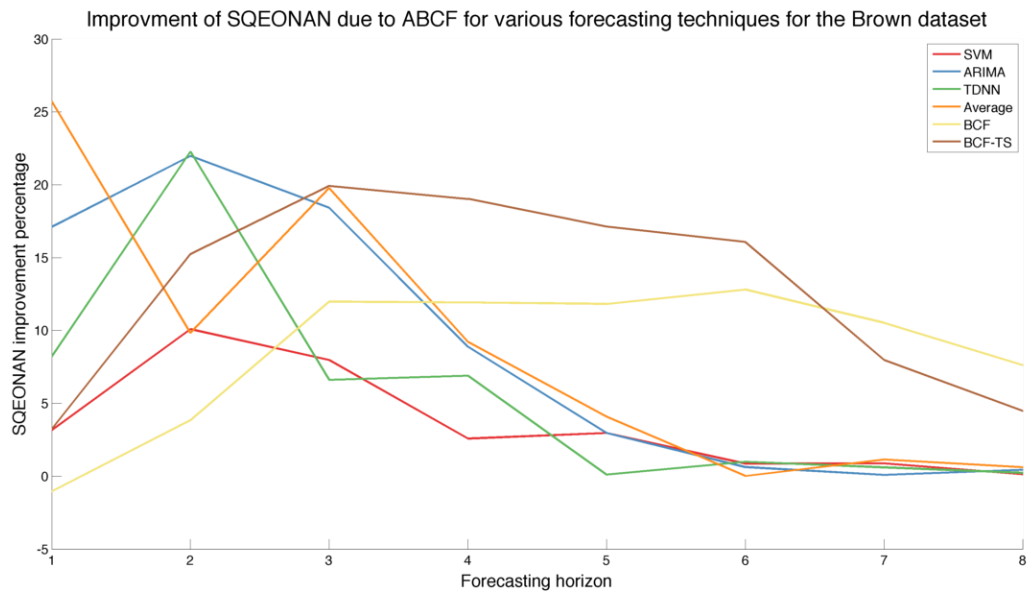


Figure 5.17

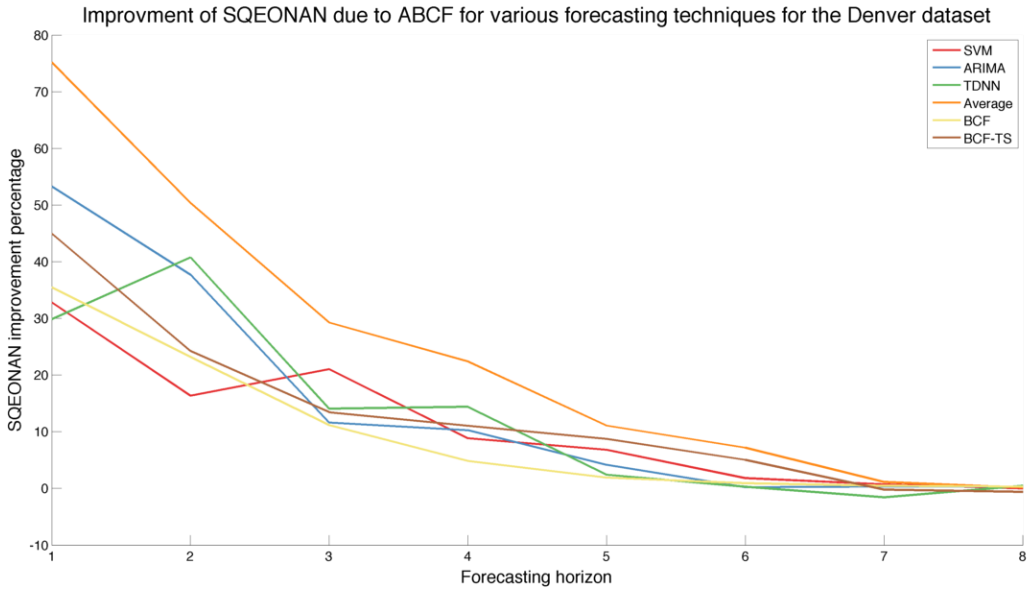


Figure 5.18

Figure 5.16, Figure 5.17, and Figure 5.18 shows the percentage improvement to the SQEONAN metric using ABCF to supplement a given base forecaster. The improvement due to ABCF varies greatly based on both forecaster and dataset. As expected, ABCF

improves a historic average forecaster by the largest margin. The improvement is over 70% on the Denver dataset and still over 25% on the Brown hall set.

At the one and two time step forecasting horizon, both BCF and our IBCF algorithm introduced in the last chapter are amongst the forecasters that benefit the least from applying ABCF. We believe the smaller improvement margins are due the properties of ensemble forecasters. Such forecasters will change forecasting models based on the structure of the dataset at any given time. These changes may not lead to clusters which are clean and thus as easily able to be modeled. I

As ABCF is an ensemble forecaster, it can make no guarantees on its improvement. For forecasts one time step ahead using the SQEONAN metric on the Brown Hall dataset, the BCF forecaster out performed a hybrid BCF + ABCF forecast. From our results, we found that such negative effects on forecasting performance were rare, occurring in only a few of our examples.

Figure 5.19, Figure 5.20, and Figure 5.21 show the improvement of applying ABCF to each model using RMSE. Figure 5.22, Figure 5.23, and Figure 5.24 show the improvements of ABCF using MASE. These images demonstrate that our forecasting improvements when applying ABCF are typically quite small (<5%) when considering the entire time series. These improvements are tied to the density of anomalies within the dataset. A dataset with more anomalies relative to the forecaster will experience a larger overall improvement in forecasting error.

Overall these results demonstrate that ABCF has the power to improve anomaly forecasting while having a little to no detrimental effect on the overall forecasting performance of our forecaster. Our worse case scenario occurred when applying ABCF to an SVM forecaster on the MERL dataset and for a forecasting horizon of 2 and 3 we experienced 3 - 4% error in forecasting performance. However, in almost all other scenarios ABCF improved the overall forecasting accuracy and greatly improved the accuracy of anomaly forecasting as measured by our SQEONAN metric.

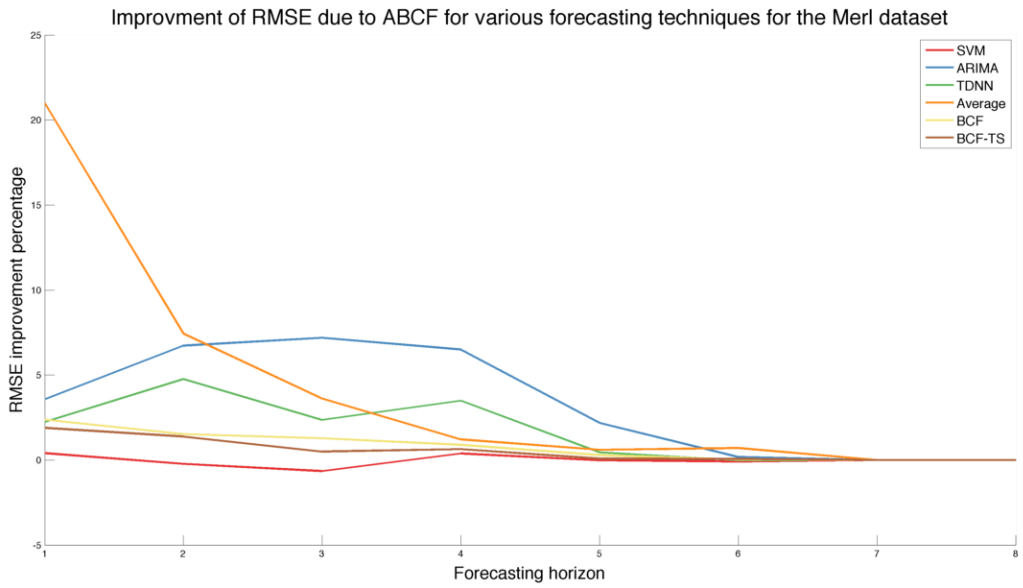


Figure 5.19

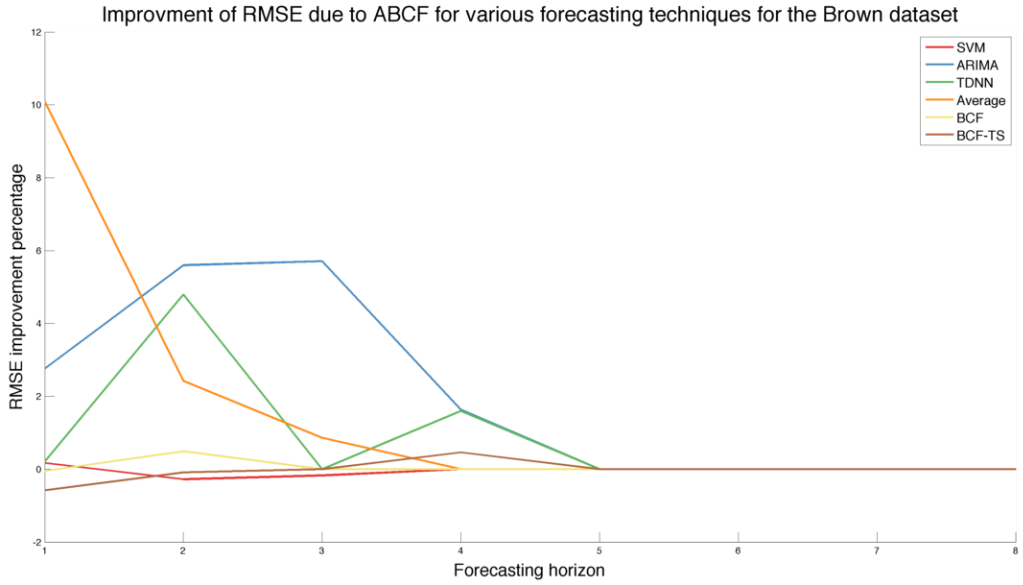


Figure 5.20

5.11 Future unexplored work on ABCF

While we are happy with the current status of the ABCF algorithm, we would recommend the following potential avenues for improvement.

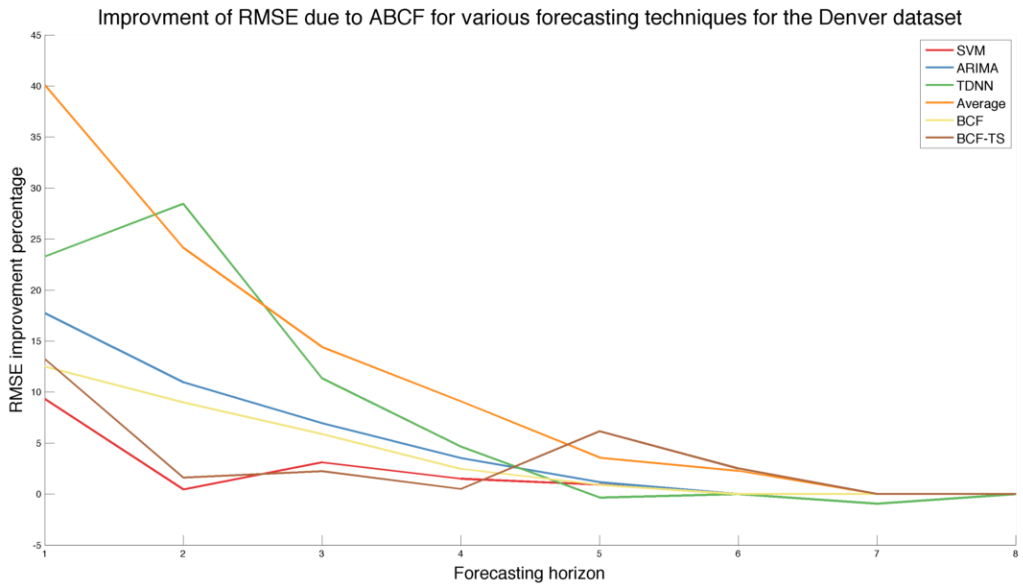


Figure 5.21

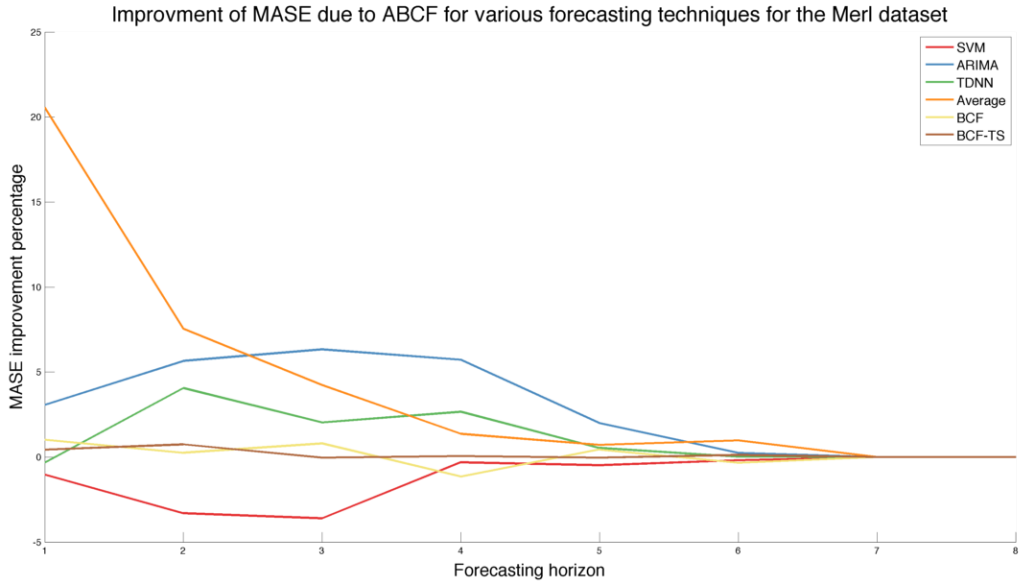


Figure 5.22

1. Explore different data extraction and clustering techniques. More research into techniques which do not rely on fixed length extracted residual windows. If a reliance on a sliding window extraction method is to continue, then trimming or growing the sliding

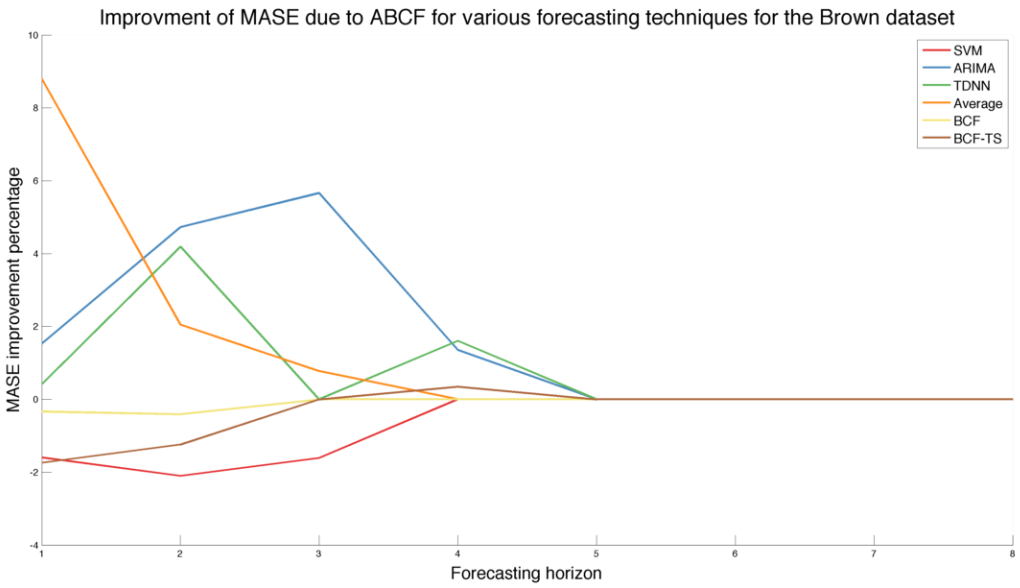


Figure 5.23

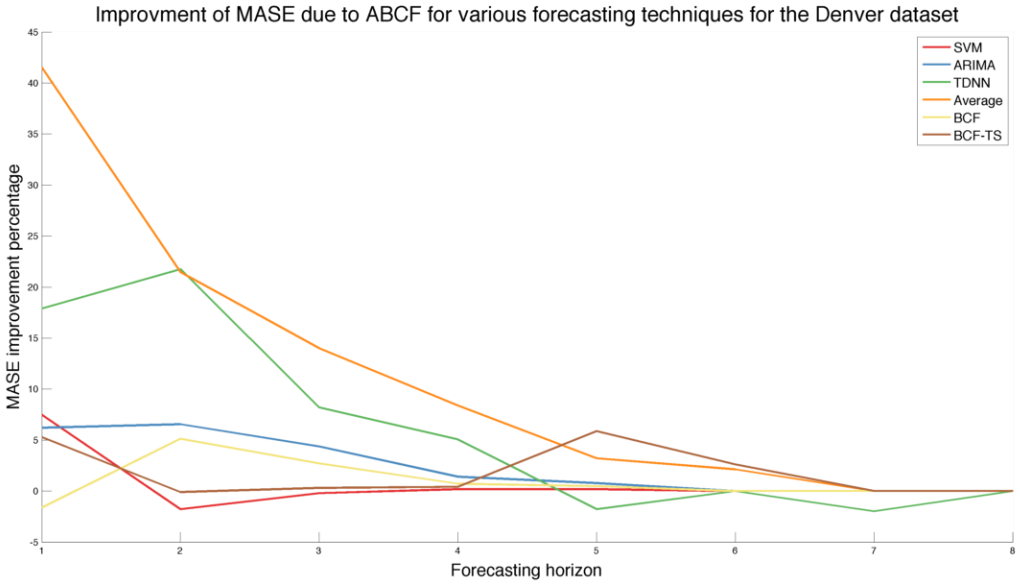


Figure 5.24

window based on the shape of any potential residual. Another possibility would be to use a sparse dictionary encoding technique to determine the a best dictionary of anomalies for each residual time series.

2. Acquire and test with other datasets. The datasets presented here give a baseline to test from, but we realize that it does not represent all types of human traffic data. We would like to test from a traffic dataset with shorter time scales (recall that the Denver dataset is every 30 minutes). Also we would like to have other types of building data. Do our approaches work with apartments or government buildings? Rich building datasets would allow us to answer this question and may lead to improvements in our approach based on the results.

3. Test with additional forecasters. The four base forecasters we've selected represent a good base, but other forecasters, such as wavelet based or Markov based, may produce some unintuitive results. We are especially interested in a forecaster or cluster which has a non-Gaussian noise distribution. Our algorithm allows for any noise distribution, but all of our tests have only been with Gaussian noise.

CHAPTER 6

CONCLUSION

Discuss our conclusions here

REFERENCES CITED

[1] US DOE. Building Energy Databook, 2010.

[2] R Balan, S Stan, and C Lapusan. A model based predictive control algorithm for building temperature control. In *3rd IEEE International Conference on Digital Ecosystems and Technologies*, pages 540 – 545, 2009.

[3] M Gwerder, B Lehmann, J Tödtli, V Dorer, and F Renggli. Control of thermally-activated building systems (TABS). *Applied energy*, 85(7):565 – 581, 2008.

[4] In-Ho Yang and Kwang-Woo Kim. Prediction of the time of room air temperature descending for heating systems in buildings. *Building and Environment*, 39(1):19–29, January 2004. ISSN 0360-1323.

[5] Yudong Ma, Francesco Borrelli, and Brandon Hencey. Model predictive control for the operation of building cooling systems. In *America Control Conference*, 2010.

[6] US DOT. National Traffic Signal Report Card, 2007.

[7] Peter Koonce, Wayne Kittelson, Lee Rodegerdts, Kevin Lee, and Caroline Swartz. The Signal Timing Manual. Technical report, Federal Highway Administration, 2008.

[8] ESPN. NFL Attendance, 2013.

[9] V Petridis, A Kehagias, L Petrou, A Bakirtzis, S Kiartzis, H Panagiotou, and N Maslaris. A Bayesian multiple models combination method for time series prediction. *Journal of intelligent and robotic systems*, 31(1):69–89, 2001.

[10] J Scott Armstrong. Evaluating forecasting methods. *Principles of forecasting*, pages 443–472, 2001.

[11] JT Yokuma and JS Armstrong. Beyond accuracy: Comparison of criteria used to select forecasting methods. *International Journal of Forecasting*, 11(4):591–597, 1995.

[12] R.J. Hyndman and A.B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[13] R.J. Hyndman. Another look at measures of forecast accuracy. *FORESIGHT*, page 46, June 2006.

[14] Rob Hyndman and Et al. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer-Verlag, Berlin, 2008.

[15] Billy M. Williams and Lester a. Hoel. Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, 129(6):664, 2003. ISSN 0733947X.

[16] Wei-Chiang Hong, Yucheng Dong, Feifeng Zheng, and Shih Yung Wei. Hybrid evolutionary algorithms in a SVR traffic flow forecasting model. *Applied Mathematics and Computation*, 217(15):6733–6747, April 2011. ISSN 00963003.

[17] G.R. Newsham and BJ Birt. Building-level occupancy data to improve ARIMA-based electricity use forecasts. In *2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys 2010)*, 2010.

[18] James Howard and William Hoff. Forecasting building occupancy using sensor network data. In *BigMine '13 Proceeding of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 87–94, Chicago, IL, 2013.

[19] Ivan Fernandez, Cruz E. Borges, and Yoseba K. Penya. Efficient building load forecasting. In *16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. Ieee, September 2011. ISBN 978-1-4577-0017-0.

[20] George Box, Gwilym Jenkings, and Gregory Reinsel. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Inc., 4th edition, 2008. ISBN 978 0 470 27284 8.

[21] Robert Nau. Seasonal Arima Models, 2014.

[22] Denis Kwiatkowski, P.C.B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[23] G. M. Ljung and G. E. P. Box. On a Measure of a Lack of Fit in Time Series Models. *Biometrika*, 65(2):297–303, 1978.

[24] Hirotsugu Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[25] Brian L. Smith and Michael J. Demetsky. Traffic Flow Forecasting: Comparison of Modeling Approaches. *Journal of Transportation Engineering*, 123(4)(August):261–266, 1997.

[26] J.V. Hansen and R.D. Nelson. Forecasting and recombining time-series compents by using neural networks. *Joural of the Operational Research Society*, 54:307–317, 2003.

[27] B Abdulhai, H Porwal, and W Recker. Short Term Freeway Traffic Flow Prediction Using Genetically-Optimized Time-Delay-Based Neural Networks. *California Partners for Advanced Transit and Highways (PATH)*, 1999.

[28] Sherif Ishak, Prashanth Kotha, and Ciprian Alecsandru. Optimization of dynamic neural network performance for short-term traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, 1836(1):45–56, 2003.

[29] C.-H. Wu, J.-M. Ho, and D.T. Lee. Travel-Time Prediction With Support Vector Regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281, December 2004. ISSN 1524-9050.

[30] Dali Wei and Hongchao Liu. An Adaptive-Margin Support Vector Regression for Short-Term Traffic Flow Forecast. *Journal of Intelligent Transportation Systems*, (806), 2013.

[31] Dali Wei and Hongchao Liu. An adaptive support vector regression for short-term traffic flow forecasting. In *91st Annual Meeting of the Transportation Research Board*, Washington D.C., 2012.

[32] Vladimir Vapnik. *Statistical Learning Theory*. Wiley Interscience, 1 edition, 1998.

[33] A. Smola and B. Scholkopf. A tutorial on support vector regression. In *NeuroColt2-TR*, 1998.

[34] CW Hsu, CC Chang, and CJ Lin. A practical guide to support vector classification. (1):1–16, 2003.

[35] J. Page, D. Robinson, N. Morel, and J.-L. Scartezzini. A generalised stochastich model for the simulation of occupant presence. *Energy and Buildings*, 40:83–98, 2008.

[36] Rhys Goldstein, Alex Tessier, and Azam Khan. Schedule-calibrated occupant behavior simulation. In *Proceedings of the 2010 Spring Simulation Multiconference on - SpringSim '10*, page 1, New York, New York, USA, 2010. ACM Press. ISBN 9781450300698.

[37] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building - BuildSys '10*, page 1, New York, New York, USA, 2010. ACM Press. ISBN 9781450304580.

[38] Sunil Mamidi, YH Chang, and R Maheswaran. Improving building energy efficiency with a network of sensing, learning and prediction agents. In *International Conference on Autonomous Agents and Multi Agent Systems. AAMAS 2012*, 2012.

[39] Sean Meyn, Amit Surana, Yiqing Lin, and SM Oggianu. A sensor-utility-network method for estimation of occupancy distribution in buildings. *Proceedings of IEEE*, pages 1494–1500, 2009.

[40] CR Wren, YA Ivanov, Darren Leigh, and J Westbues. The MERL motion detector dataset: 2007 workshop on massive datasets. In *ICMI Workshop on Massive Datasets*, 2007.

[41] William A Hoff and James W Howard. Activity recognition in a dense sensor network. In *1st International Conference on Sensor Networks and Applications (SNA2009)*, page 6, 2009.

[42] Christopher R. Wren and Srinivasa G Rao. Self-configuring , Lightweight Sensor Networks for Ubiquitous Computing. In *International Conference Ubiquitous Computing*, 2003.

[43] Christopher R. Wren and E. Tapia. Toward scalable activity recognition for sensor networks. *Location-and Context-Awareness*, pages 168–185, 2006.

[44] Christopher R. Wren, Yuri Ivanov, Ishwinder Kaur, Darren Leigh, and Jonathan Westhues. SocialMotion : Measuring the Hidden Social Life of a Building. In *International Symposium on Location and Context Awareness (LoCA)*, pages 85–102, 2007.

[45] Wen Dong, Bruno Lepri, and Alex (Sandy) Pentland. Modeling the co-evolution of behaviors and social relationships using mobile phone data. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia - MUM '11*, pages 134–143, New York, New York, USA, 2011. ACM Press. ISBN 9781450310963.

[46] D.C. Minnen and Christopher R. Wren. Finding temporal patterns by data decomposition. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 608–613. Ieee, 2004. ISBN 0-7695-2122-3.

[47] Christopher R. Wren, D. C. Minnen, and S. Rao. Similarity-based analysis for large networks of ultra-low resolution sensors. *Pattern Recognition*, 39(10):1918–1931, October 2006. ISSN 00313203.

[48] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network: comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 1857:74–84, 2003.

[49] Brian L. Smith, Billy M. Williams, and R. Keith Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, 10(4):303–321, August 2002. ISSN 0968090X.

[50] Yang Zhang and Yuncai Liu. Comparison of Parametric and Nonparametric Techniques for Non-peak Traffic Forecasting. *World Academy of Science, Engineering and Technology*, 51:8–14, 2009.

[51] M.S. Tracton and E Kalnay. Operational ensemble prediction at the National Meteorological Center: practical aspects. *Weather and Forecasting*, 8(3):379–398, 1993.

[52] Hailing Zhang and Zhaoxia Pu. Beating the Uncertainties: Ensemble Forecasting and Ensemble-Based Data Assimilation in Modern Numerical Weather Prediction. *Advance in Meteorology*, page 10, 2010.

[53] Weizhong Zheng, Der-Horng Lee, and Qixin Shi. Short-Term Freeway Traffic Flow Prediction: Bayesian Combined Neural Network Approach. *Journal of Transportation Engineering*, 132(2):114, 2006. ISSN 0733947X.

[54] F Diebold. A Note on Bayesian Forecast Combination Procedures. In A. H. Westlund and P. Hackl, editors, *Economic Structural Change: Analysis and Forecasting*, pages 225–232. Springer-Verlag, 1991.

[55] G Marsaglia, WW Tsang, and J Wang. Evaluating Kolmogorov's Distribution. *Journal of Statistical Software*, 8(18):1–4, 2003.

[56] R.H.C Lopes, I Reid, and P.R Hobson. The two-dimensional Kolmogorov-Smirnov test. In *International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, Amsterdam, the Netherlands, 2007.

[57] A Kontonikas and A Kostakis. On monetary policy and stock market anomalies. *Journal of Business Finance & ...*, 2013.

[58] S.C Thushara and Prabath Perera. Day of the Week Effect of Stock Returns: Empirical Evidence from Colombo Stock Exchange. *Kelaniya Journal of Management*, 2014.

[59] AG Tartakovsky, A.S Polunchenko, and G Sokolov. Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):4–11, 2013.

[60] P. Gogoi, D. K. Bhattacharyya, B. Borah, and J. K. Kalita. A Survey of Outlier Detection Methods in Network Anomaly Identification. *The Computer Journal*, 54(4): 570–588, March 2011. ISSN 0010-4620.

[61] Eamonn Keogh, Stefano Lonardi, and B. Chui. Finding surprising patterns in a time series database in linear time and space. In *The 8th ACM SIGKDD Internationl Conference on Knowledge Discovery and Data Mining*, pages 550–556, Edmonton, Alberta, Canada, 2002.

[62] Florian Stoffel, Fabian Fischer, and Daniel a. Keim. Finding anomalies in time-series using visual correlation for interactive root cause analysis. *Proceedings of the Tenth Workshop on Visualization for Cyber Security - VizSec '13*, pages 65–72, 2013.

[63] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 34(4):219, October 2004. ISSN 01464833.

[64] Lei Shi, Qi Liao, and Chunxin Yang. Investigating Network Traffic Through Compressed Graph Visualization. In *IEEE Conference of Visual Analytics Science and Technology*, pages 179–180, 2012. ISBN 9781467347532.

[65] E.S Page. A test for a change in a parameter occuring at an unknown point. *Biometrika*, 42:523–527, 1955.

[66] Arnaud Dessein and Arshia Cont. *Online change detection in exponential families with unknown parameters*. Springer Berlin Heidelber, 2013.

[67] AS Polunchenko and AG Tartakovsky. State-of-the-art in sequential change-point detection. *Methodology and Computing in ...*, pages 1–34, 2012.

[68] J. Reeves, J. Chen, X. L. Wang, R. Lund, and Q. Q. Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meterology and Climatology*, 46(6):900 – 915, 2007.

[69] R.P. Adams and D.J MacKay. Bayesian Online Changepoint Detection. Technical report, University of Cambridge, 2007.

[70] S Liu, M Yamada, N Collier, and M Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:73–83, 2013.

[71] Liang Wang, Tao Gu, Xianping Tao, and Jian Lu. Sensor-Based Human Activity Recognition in a Multi-User Scenario. *Ambient Intelligence*, pages 78–87, 2009.

[72] Ling Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[73] N.C. Krishnan and S. Panchanathan. Analysis of Low Resolution Accelerometer Data for Continuous Human Activity Recognition. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3337–3340. IEEE, 2008.

[74] Mitja Luštrek and Boštjan Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33(2):205–212, 2009.

[75] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Fourth IEEE International Conference on Multimodal Interfaces*, pages 3–8. IEEE Comput. Soc, 2002. ISBN 0-7695-1834-6.

[76] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*, page 10, 2008.

[77] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003. ISSN 1532-4435.

[78] Raffay Hamid, Siddhartha Maddi, Aaron Bobick, and Irfan Essa. Structure from Statistics - Unsupervised Activity Analysis Using Suffix Trees. In *Internation Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007. IEEE.

[79] Yoav Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156. MORGAN KAUFMANN PUBLISHERS, INC., 1996.

[80] Raffay Hamid, Siddhartha Maddi, Aaron Bobick, and Irfan Essa. Unsupervised analysis of activity sequences using event-motifs. In *The 4th ACM international workshop on Video surveillance and sensor networks - VSSN '06*, page 71, New York, New York, USA, 2006. ACM Press. ISBN 1595934960.

[81] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing.*, volume 6, pages 3037–3040. Ieee, 1999. ISBN 0-7803-5041-3.

[82] Tâm Huynh and Bernt Schiele. Analyzing Features for Activity Recognition. In *The 2005 joint conference on Smart objects and ambient intelligence innovative context-aware services: usages and technologies*, number october, pages 159–163, New York, New York, USA, 2005. ACM Press. ISBN 1595933042.

[83] T. Kim, G. Shakhnarovich, and R. Urtasun. Sparse coding for learning interpretable spatio-temporal primitives. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2010.

[84] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689 – 696, 2009.

[85] Andrew Y. Ng. Machine Learning Lecture Notes, 2014.

[86] Emil Eirola and Amaury Lendasse. Gaussian Mixture Models for Time Series Modelling, Forecasting, and Interpolation. In *Advances in Intelligent Data Analysis XII*, pages 162 – 173. Springer Berlin Heidelberg, 2013.

[87] Leena Kalliovirta, Mika Meitz, and Pentti Saikkonen. A Gaussian mixture autoregressive model for univariate time series. Technical report, Helsinki Center of Economic Research, 2012.

[88] Robert L. Thorndike. Who Belong in the Family? *Psychometrika*, 18(4):267–276, 1953.

[89] Gideon E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2): 461–464, 1978.

[90] Peter J. Rousseuw. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*, 20:53 – 65, 1987.

[91] Mingjin Yan. *Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion*. PhD thesis, 2005.

APPENDIX A - OTHER RESULTS

All remaining results are in this section.

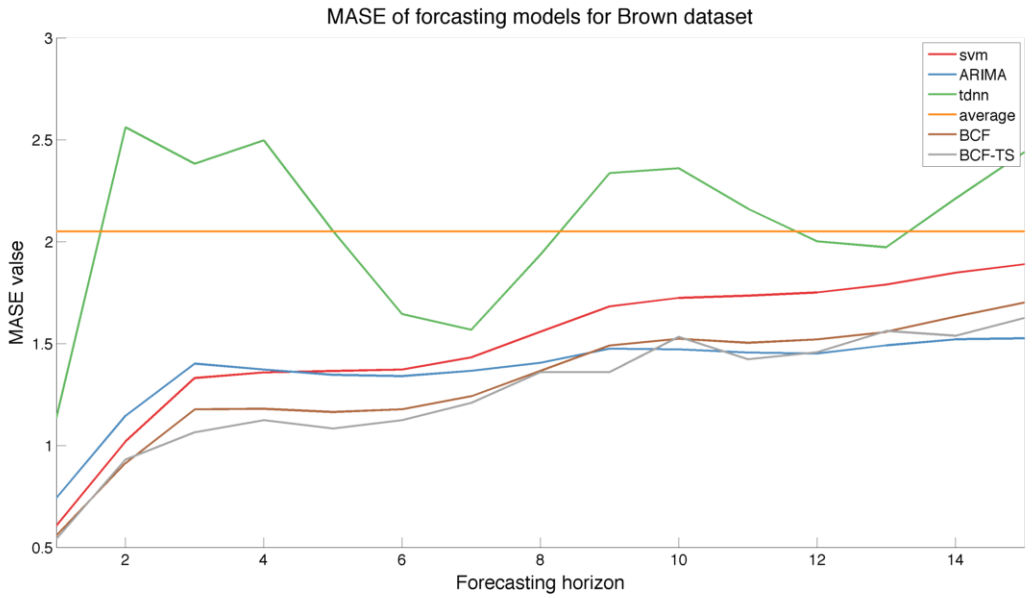


Figure A.1: Brown dataset: Root mean square error of forecasting for each model vs forecasting horizon.

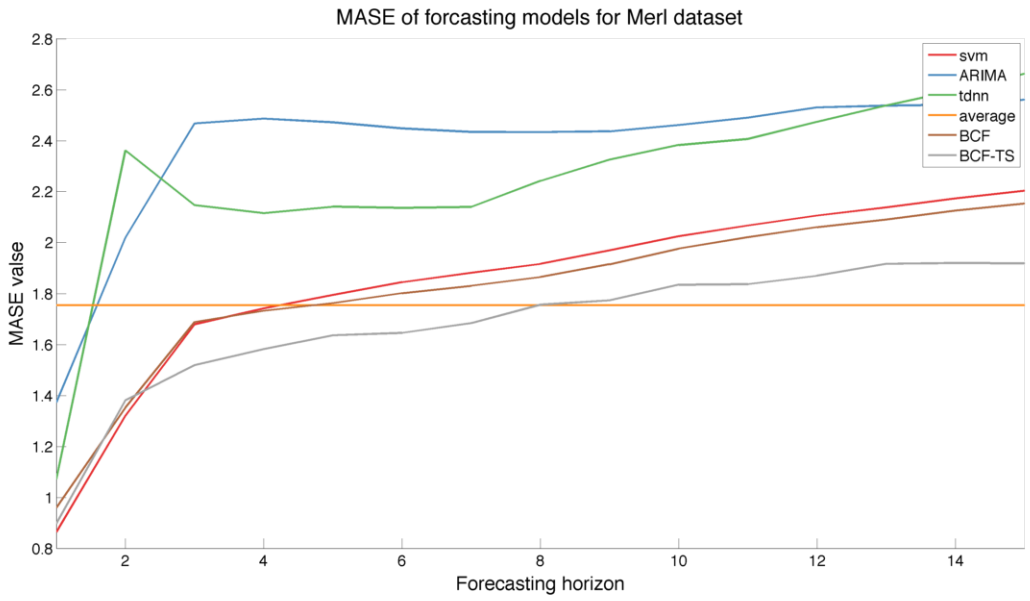


Figure A.2: Merl dataset: root mean square error of forecasting for each model vs forecasting horizon.

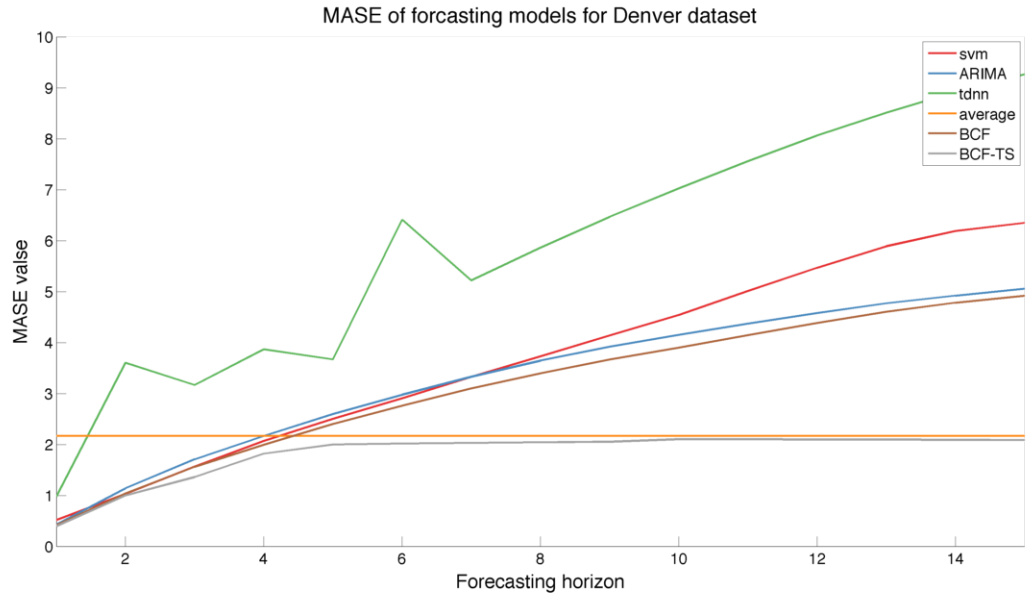


Figure A.3: Denver dataset: root mean square error of forecasting for each model vs forecasting horizon.

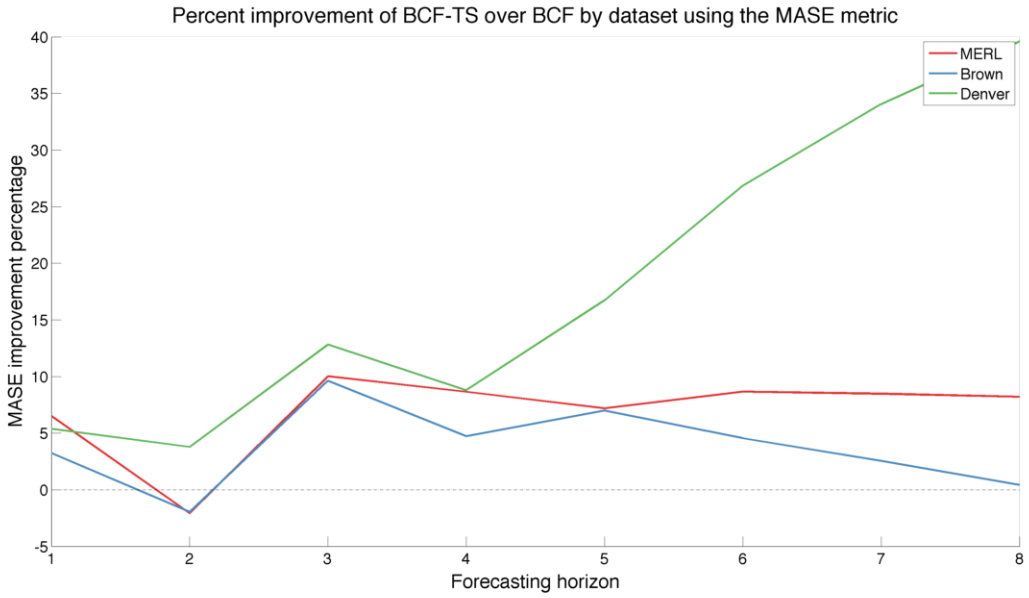
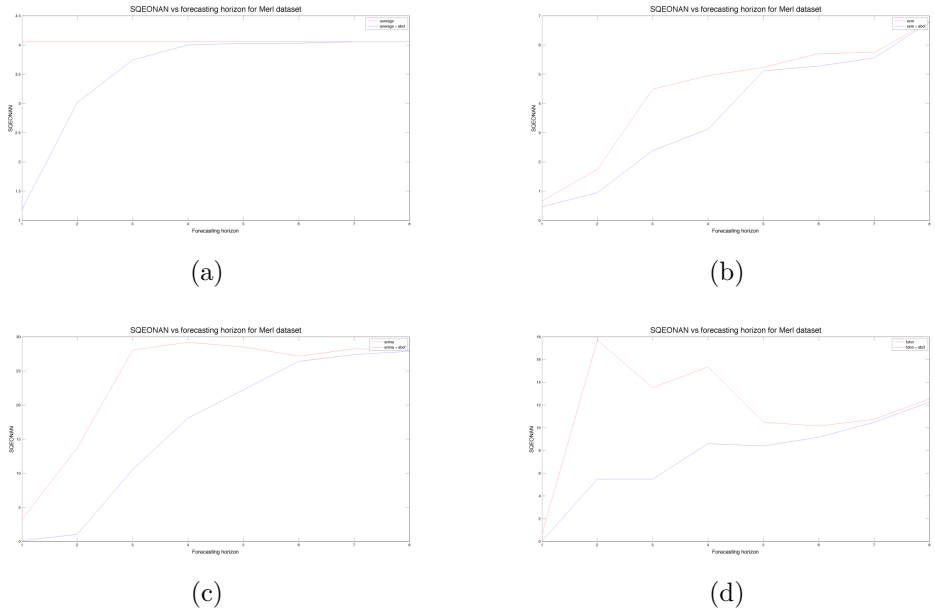


Figure A.4: Improvement percentage of MASE values of BCF-TS compared to BCF.



(a) (b)

(c) (d)

Figure A.5: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm

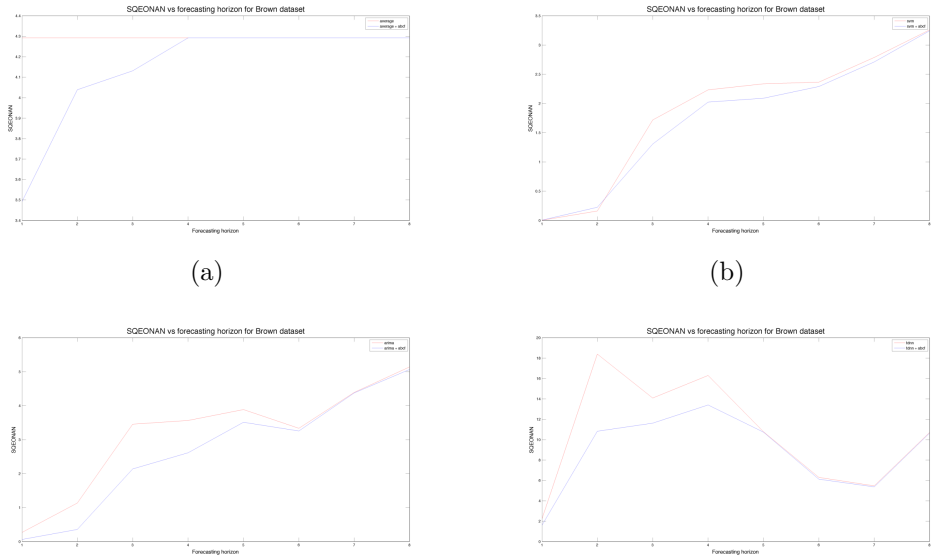


Figure A.6: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm

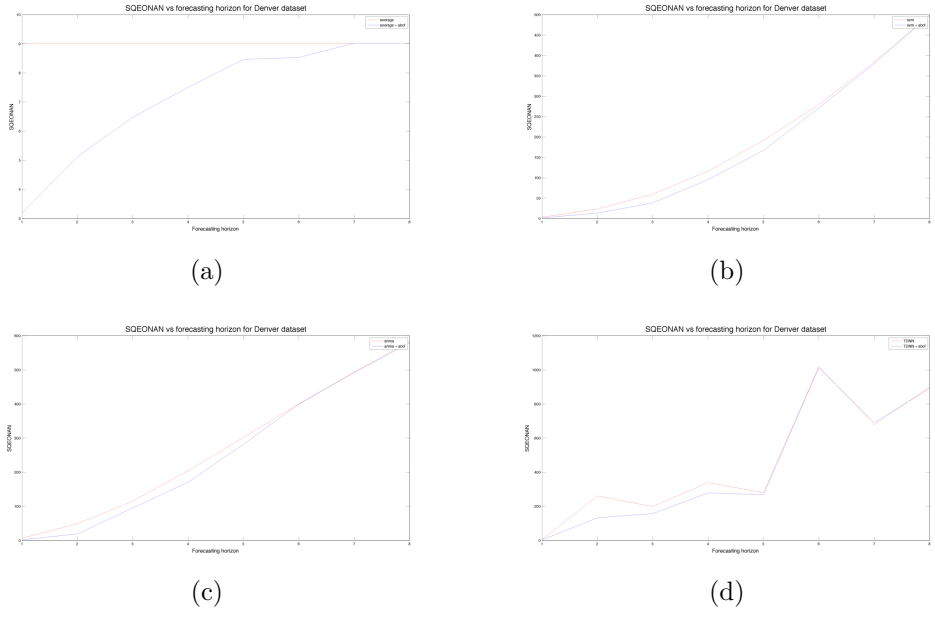
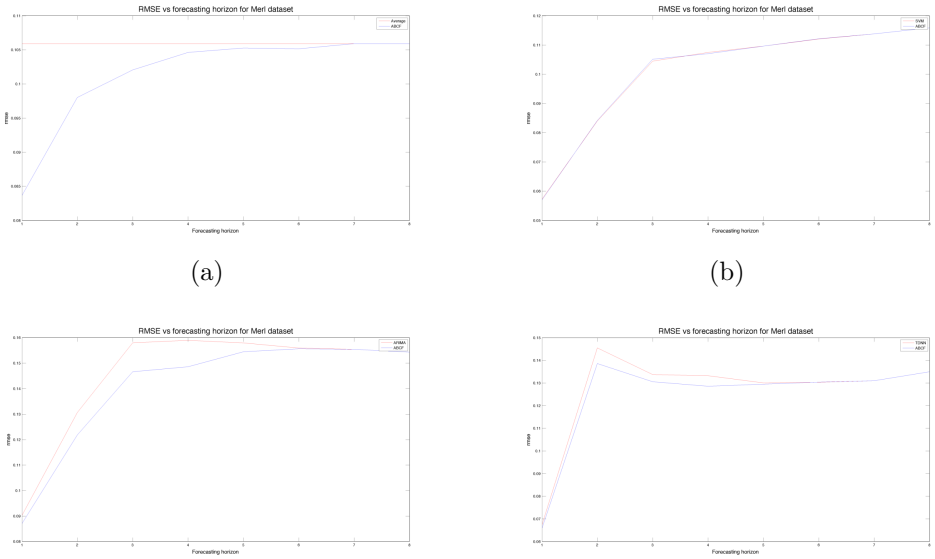


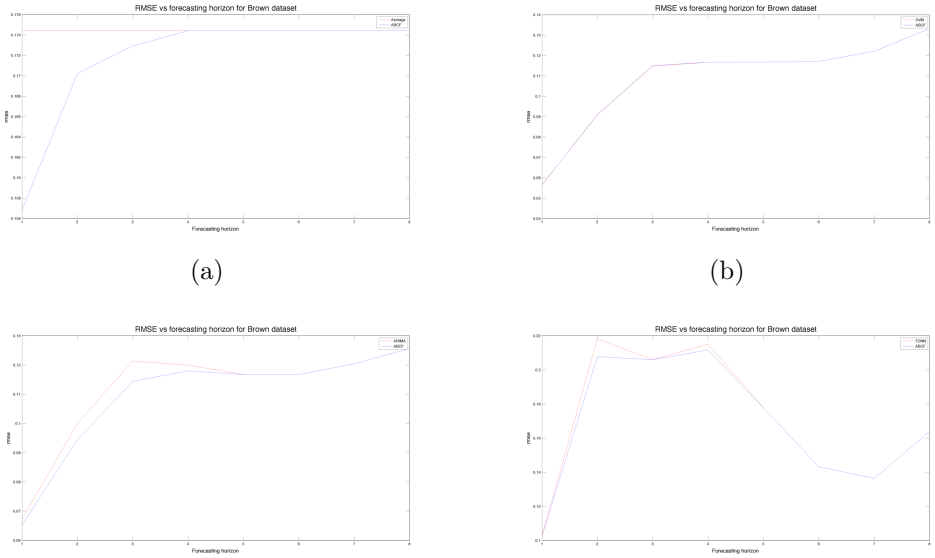
Figure A.7: Results for the four base forecasting algorithms for the Denver dataset and the improvements to SQEONAN from using our ABCF algorithm



(a) (b)

(c) (d)

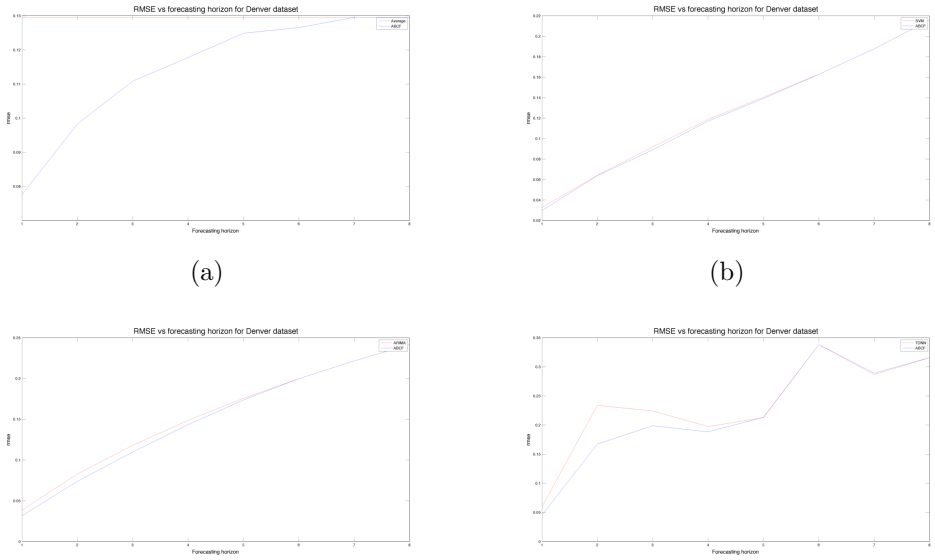
Figure A.8: Results for the four base forecasting algorithms for the MERL dataset and the improvements to RMSE from using our ABCF algorithm



(a) (b)

(c) (d)

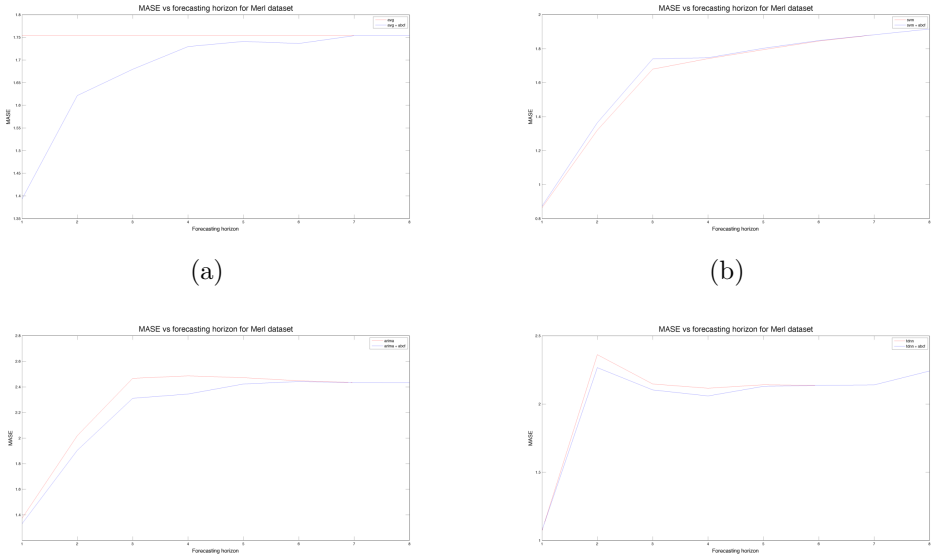
Figure A.9: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to RMSE from using our ABCF algorithm



(a) (b)

(c) (d)

Figure A.10: Results for the four base forecasting algorithms for the Denver dataset and the improvements to RMSE from using our ABCF algorithm



(a) (b)

(c) (d)

Figure A.11: Results for the four base forecasting algorithms for the MERL dataset and the improvements to MASE from using our ABCF algorithm

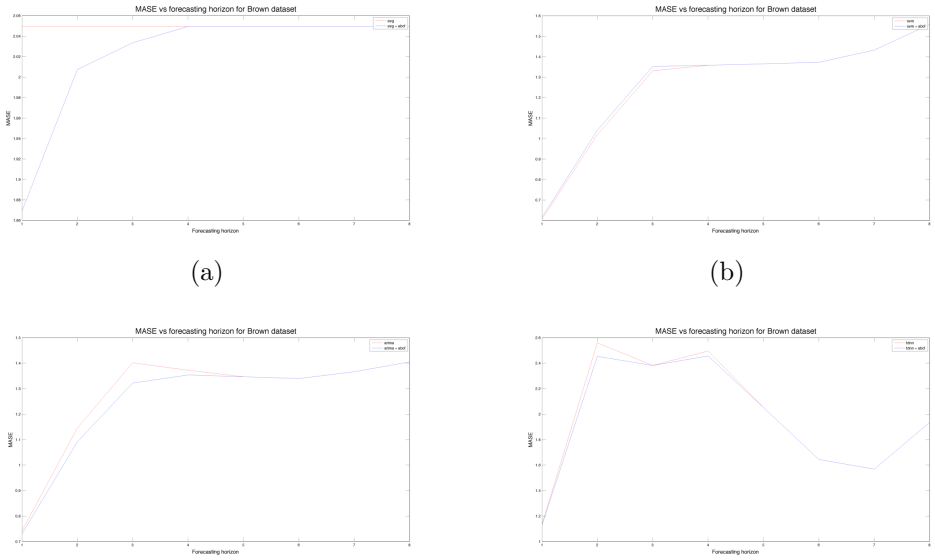


Figure A.12: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to MASE from using our ABCF algorithm

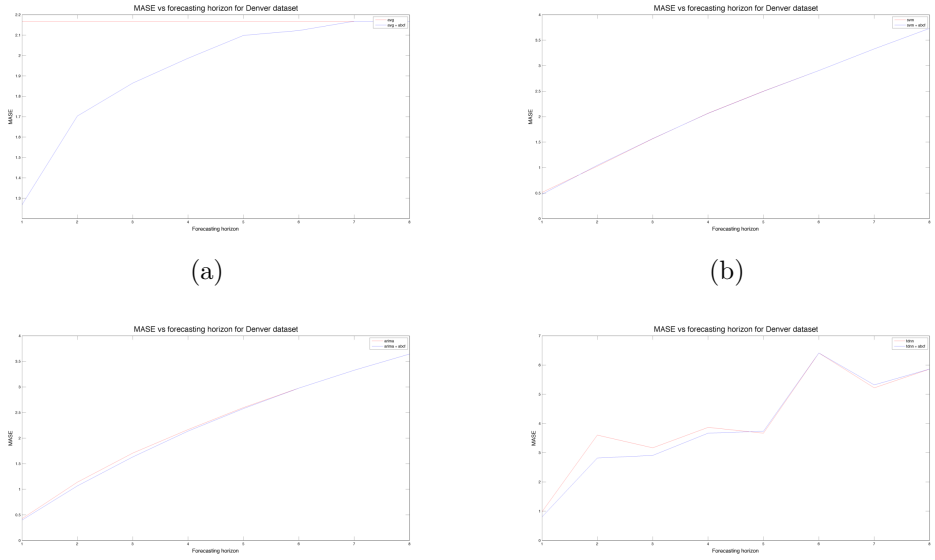


Figure A.13: Results for the four base forecasting algorithms for the Denver dataset and the improvements to MASE from using our ABCF algorithm

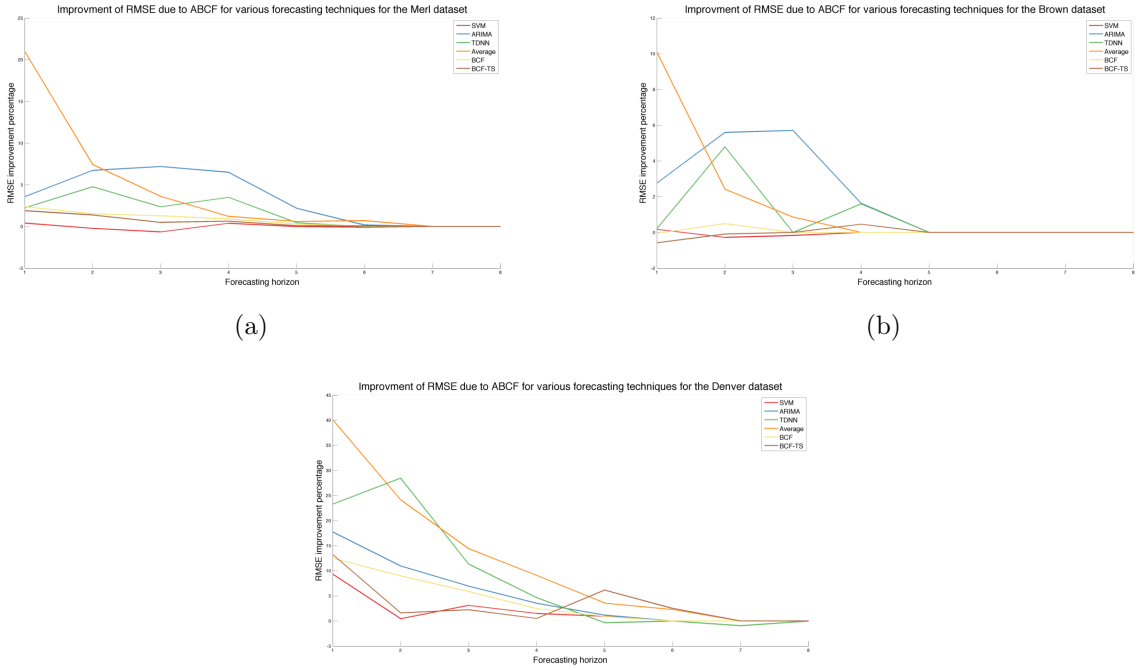


Figure A.14: Percentage improvement to RMSE due to application of ABCF

APPENDIX B - FIGURE REFERENCES

This thesis was far too long in coming and far to full of various figures that do not have nice references. [1]

[1] This footnote is simply to reference all of the unreferenced images. Enjoy. Figure A.5, Figure A.6, Figure A.7, Figure A.8, Figure A.9, Figure A.10, Figure A.11, Figure A.12, Figure A.13, Figure A.14,