A METHOD FOR IMPROVED ENSEMBLE

FORECASTING FOR HUMAN 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

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1
for candidate data extraction46

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 unto 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 doesn't 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? Ideally the building control system should identify a lower than average number of occupants and modify its heating or cooling schedule to

1

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 account for 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 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 eluded 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. 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. 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 incredibly beloved. 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 affect 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. What is perhaps surprising is 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 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)
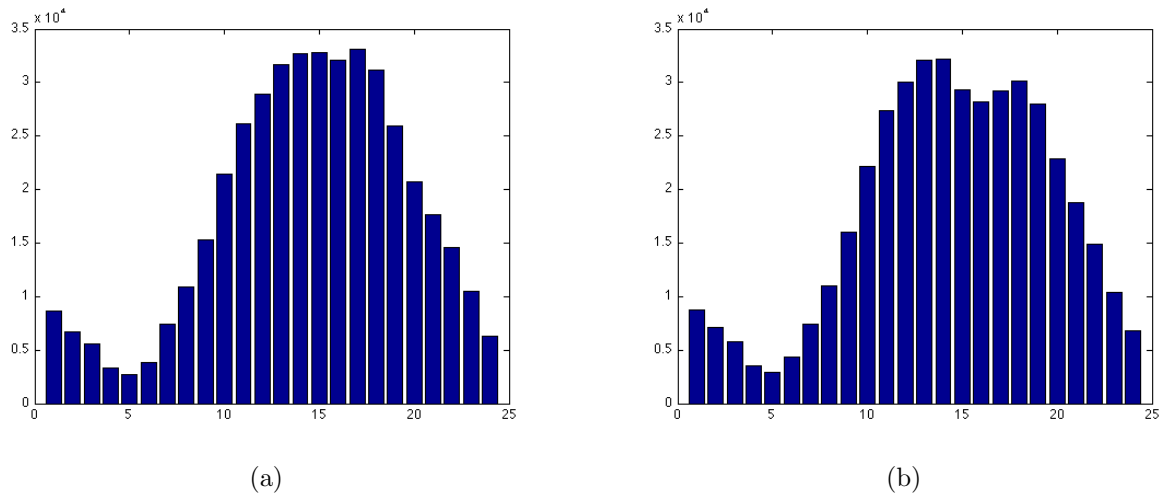
Figure 1.1: Total number of cars passing major highway sensors in Denver on Sundays in September and October 2010

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

4

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 off of the Bayesian combined forecaster (BCF) created by Petridis [40]. Chapter 4 discusses modifications we've 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've 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 cause problems with 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.
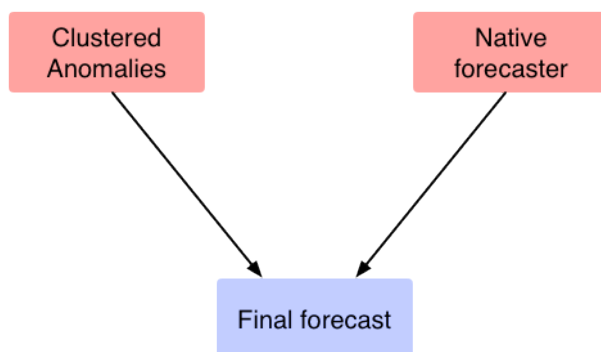
Figure 1.2: High level overview of our approach

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 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.

Finally, in satisfying our third objective, we keep everything unsupervised. For the case of infrequent activities such as sporting events, there is nothing inhibiting supervising modeling of the activity and incorporating that 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:

- Use of activity models for improved forecasting accuracy of multiple time steps into the future.

- 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: Section two defines typical forecasting models which are to be used for comparison and as base native forecasters for our final ensemble forecaster. This section also introduces some notation used through this document. Section three discusses the datasets used therein along with specifics details on any necessary changes to make the datasets compatible with our forecasters. Section four 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 section five 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

DATASETS

TODO FOR THIS CHAPTER:

1. Why we use a constrained time window instead of the whole day

2. show two sensor location for a dataset (MERL) and discuss the different shapes of the data. Talk about why it is not necessary to generate results from each location as our datasets already have different shapes and scales of counts

3. put a marker on the CSMBB sensor we use

4. put a marker on Denver sensor we use

5. solve the problem of references subfigures - this CSM latex package is different than other packages I've used before for subfigure reference

6. discuss the noise in the final datasets

7. figure out proper name of Denver traffic sensors

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 high volume with values taken at 30 minute increments. 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. All data is scaled to values between 0 and 1.

2.1 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 [9, 10]. 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, for our work, 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 [11] 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 [12] 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.

Finally, Meyn, et al [13] 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. Also, we were unable to acquire the datasets for testing and due either to the low volume of occupancy or short duration it is likely that the datasets produced by these researchers would not be sufficient

9

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 [14]. The second is a classroom and office building from the Colorado School of Mines (CSMBB) [15, 16]. Both datasets use passive infrared sensors (Figure 2.1) to estimate motion in an area.

Figure 2.1: Passive infrared motion detector

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. 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.

2.1.1 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 [14, 17–22]. However, none of this research to our knowledge focused on building occupancy forecasting. The closest related work would be in tracking individuals within the building.

(a)　　　　　　　　　　　　　　　(b)
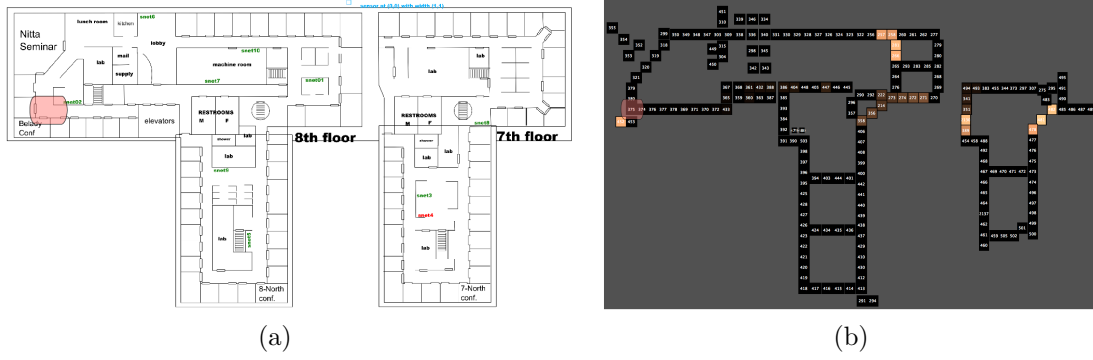
Figure 2.2: Floor plan and sensor locations for the MERL office building dataset. The red rectangle is the location of our sensor for this work.

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 Figure 2.2.

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 and holidays are observed as normal.

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 ??.

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.
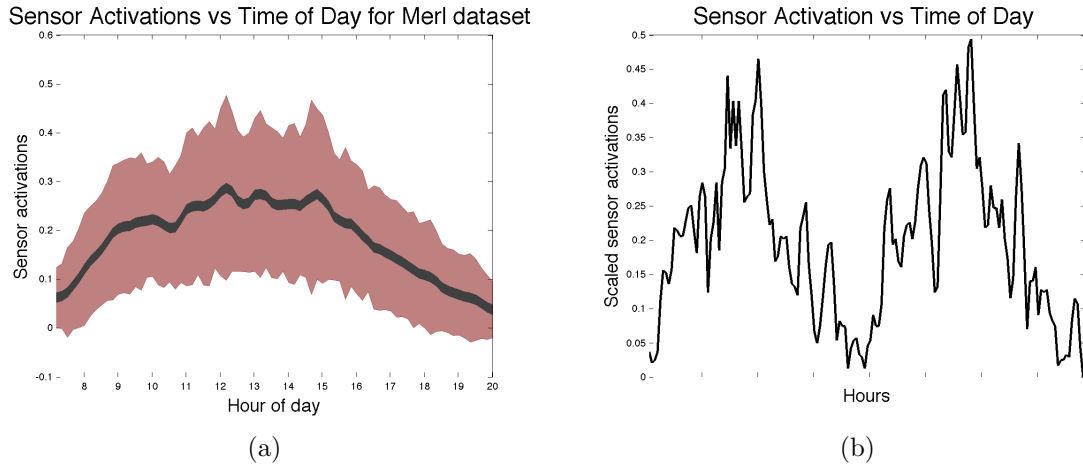
Figure 2.3: Scaled average readings for a given day and scaled readings for two days from the MERL dataset.
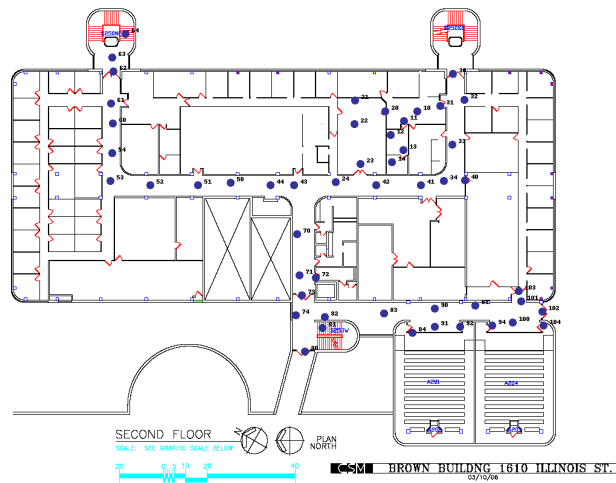
Figure 2.4: Sensor locations for the Colorado School of Mines Brown Building

As a visual for the typical data values throughout a couple of days is given in Figure ?? ?? . Despite the mean appearing smooth, we see how volatile sensor activation readings are though out the day.

2.1.2 Colorado School of Mines Dataset

The Colorado School of Mines dataset 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 the lower right of Figure Figure 2.4 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.

(a)
(b)
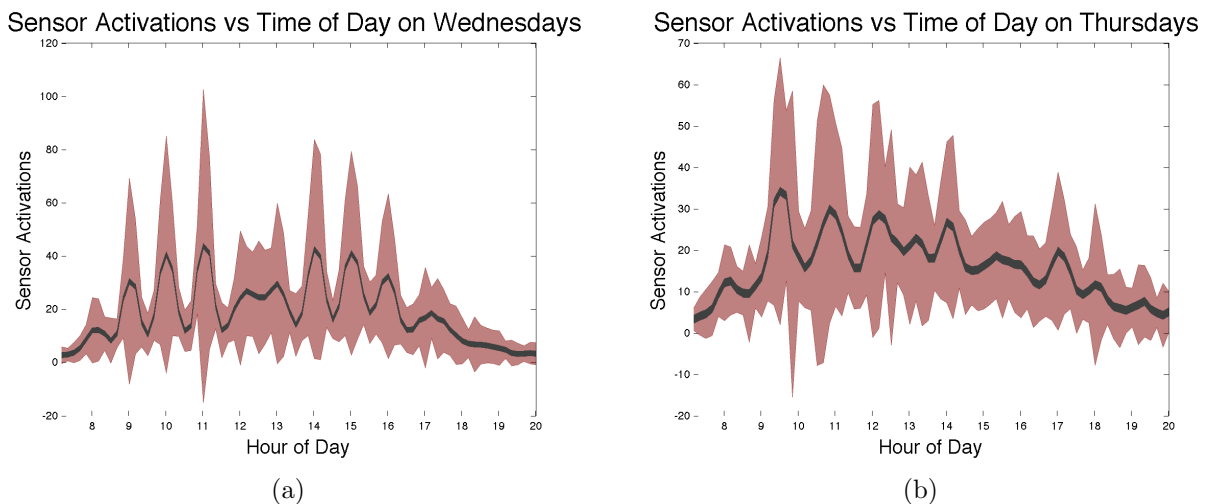
Figure 2.5: Average readings from Wednesdays and Thursdays for a given sensor from the MERL dataset.

Figure 2.5

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.

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 ??. 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. 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)

(b)
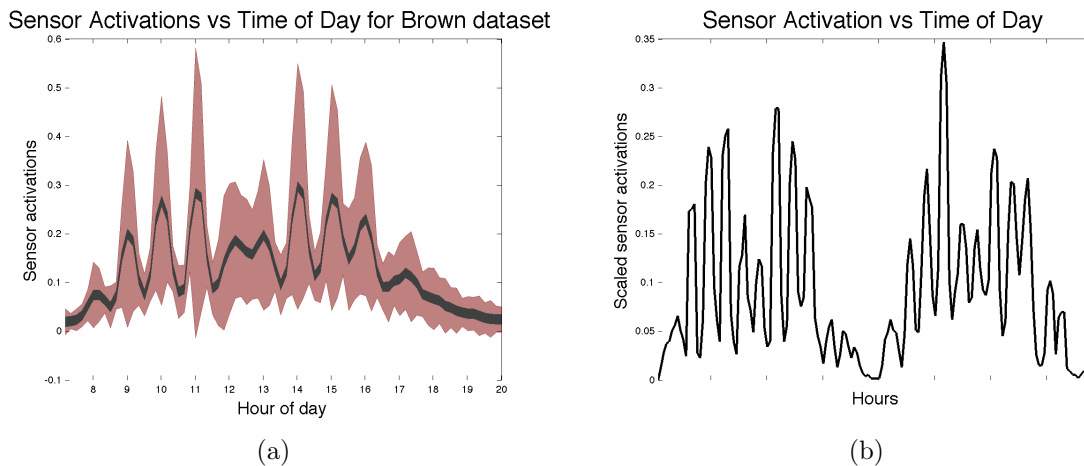
Figure 2.6: Scaled average readings for a given day and scaled readings for two days from the CSMBB dataset.

For this work, we focus on the Monday, Wednesday and Friday class schedules. Figure **??** shows the average of our CSMBB dataset. Figure **??** shows a example of raw sensor activations for two days within our CSMBB dataset.

2.2 Traffic datasets

Numerous traffic datasets have been used in the time series forecasting literature [23–26]. Vehicular traffic datasets are an excellent test for forecasting algorithms as they tend to be highly repetitive with a significant seasonal component.

2.2.1 Denver traffic dataset

The Denver traffic dataset is collected from many in road (TODO FIGURE OUT SENSOR TYPE) sensors which count the number of vehicles passing the sensor throughout the day. Data is available from 2008 to 2011 on most days of the week. The data is nearly 1

14
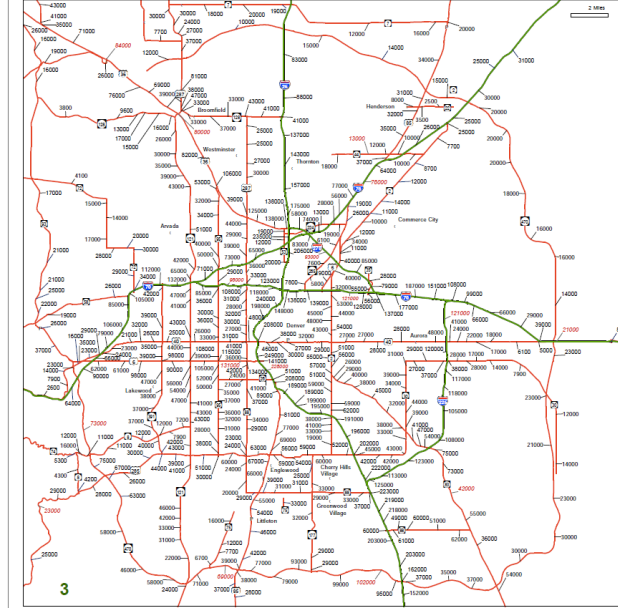
Figure 2.7: City of Denver traffic sensors.

million readings for the sensors we extracted and approximately 33,000 readings for any one sensor. 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 Figure 2.7 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 Figure 2.8. 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.
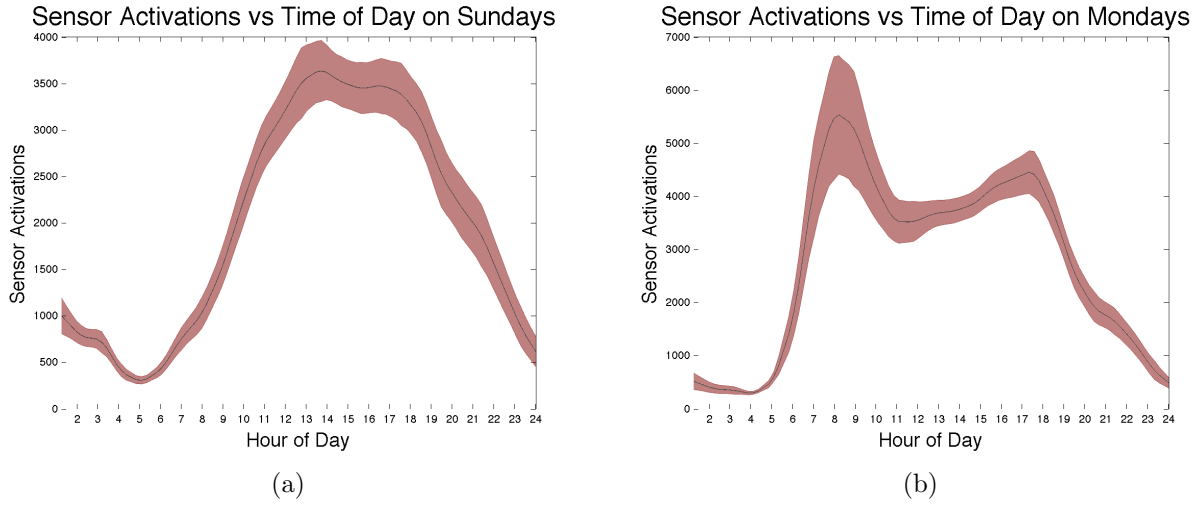
Figure 2.8: Average readings from Sunday and Monday for a given sensor from the Denver traffic dataset.

The scaled dataset that we use for this work is shown in figure **??**. 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 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, do not want to remove more reading from the dataset. Finally figure **??** shows an example of two days of scaled data. Due to the one hour readings instead of 10 minute readings, the data appears much smoother.

2.3   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.
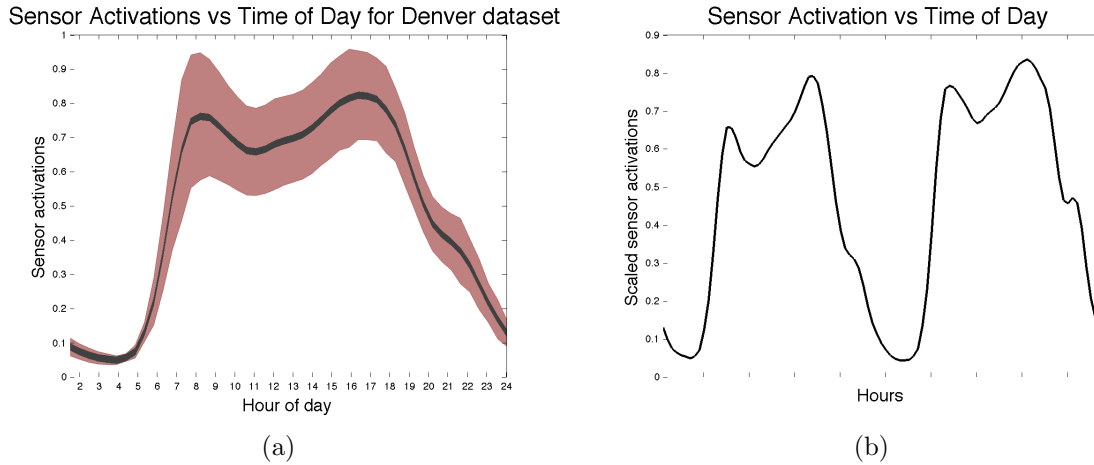
Figure 2.9: Scaled average readings for a given day and scaled readings for two days from the Denver traffic dataset.

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 approach to forecasting out perform a univariate approach this was show to not be the case by Kamarianakis and Prastacos **??**. 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 simple fact time aggregation and building scale. 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, long that 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.

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 replace 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 3

MODELS

With this chapter we explore some of the most common forecasting models on our datasets. These models are meant to be used as a baseline from which to compare our results. Also in this chapter, we introduce the metrics which will be used for our analysis. We feel this analysis useful to empirically demonstrate the various forecasting capabilities of these established models.

TODO IN THIS CHAPTER Citations Image for Historic Average Image for TDNN Image for ARIMA Image for SVM Analysis and Table of results

3.1 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

$$y_{t+1}^{k,m} = f(x_t, ..., x_1; \theta_k). \tag{3.1}$$

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

$$y_{t+2} = f(y_{t+1}, x_t, ..., x_1; \theta_k). \tag{3.2}$$

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

$$r_{t,\delta} = x_{t+\delta} - y_{t+\delta}. \tag{3.3}$$

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

3.2 Forecast measurements

We use mean absolute scale error (MASE) [27, 28] and root mean squared error (RMSE) as cost functions to compare with other previously implemented techniques. These forecasting cost functions are commonly used in forecasting literature (TODO: CITE) and provide a baseline by which to compare our novel forecasting approach to classic forecasting models.

Also, because some 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 (RMSEONAN). This metric provides a convenient way to measure the improvement our approach yields which is not easily captured by RMSE and MASE.

RMSE

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. In this case, the input series x and the forecast series y.

$$RMSE_\delta = \sqrt{\frac{\sum_{t=1}^{N-\delta} (x_{t+\delta} - y_{t+\delta})^2}{N - \delta}} \tag{3.4}$$

MASE

Mean absolute scaled error was developed to be a generally applicable measurement of forecast accuracy. The metric is especially useful for datasets with intermittent demand unlike the common closely related measurement of mean absolute percentage error (MAPE). The

reason it works well with intermittent time is that it will not return undefined or infinite values unless all dataset is equal [29].

$$MASE_\delta = \frac{\sum_{t=1}^{N-\delta} r_{t,\delta}}{\frac{N}{N-1} \sum_{i=2}^{N} |y_i - y_{i-1}|} \tag{3.5}$$

RMSEONAN

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 but he historic average (naive) model. Some later results will include larger boundaries such as two and three standard deviations. RMSEONAN is defined as

$$RMSEONAN_{\delta,\sigma} = \frac{\sqrt{\sum_{t=1}^{N-\delta} A(r_{t,\delta}; \sigma)^2}}{N - \delta}. \tag{3.6}$$

Where $A(r_\delta; \sigma)$ is

$$A(r_{t,\delta}; \sigma_t) = \begin{cases} r_{t,\delta} & \text{If } r_{t,\delta} \geq \sigma_t \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

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 an average of all values of $A(r_\delta)$. This is because RMSEONAN 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.

21

An example demonstrating the regions summed by RMSEONAN is given by Figure 3.1. In this image, only the area of the green spaces are squared and summed towards RMSEO-NAN. 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.
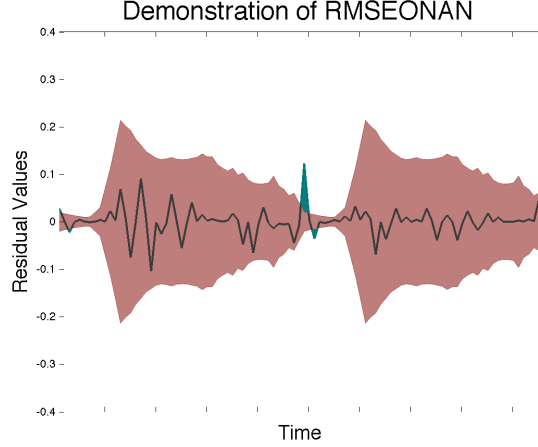
Figure 3.1: Demonstration of the area measured by RMSEONAN

3.3   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 [25, 30, 31]. Its forecasting accuracy is quite strong and it can serve as a strong baseline of comparison for a forecasting problem.

Due to that fact that our building data has periodic trends and a non stationary mean, a seasonal ARIMA model is best suited to fit our data from the class of ARMA models. The seasonal ARIMA model is defined as:

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla^D_s T_t = \theta_q(B)\Theta_Q(B^s)e_t \tag{3.8}$$

Table 3.1: 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

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 ϕ, Φ, θ, Θ are trainable parameters.

Seasonal ARIMA models are notated as

$$ARIMA(p, d, q)(P, D, Q)_s \tag{3.9}$$

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 [32].

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 similar to Williams [25]. As a verification of our model, we applied the LJung-Box test [33] 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 $p > 0.05$. Both residual sets passed: $p = 0.9964$ for MERL and $p = 0.1072$ for CSMBB. Our final model parameters can be seen in Table Table 3.1. Notice that the season is different for each model due to a difference in window of time for each day that we extracted data.

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:

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla_s^D T_{t+1} = \theta_{q-1}(B)\Theta_{Q-1}(B^s)e_t \qquad (3.10)$$

3.3.1 Fitting a Seasonal ARIMA

The underlying math of ARMA models stems from a linear filter operating on input from a stationary stochastic process. ARIMA models were created to handle non-stationary data by differencing the data to induce stationarity. Thus, a necessary step in fitting a ARIMA model to the data is first to determine the steps necessary to make the time series weakly stationary. 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.

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 **??** shows the raw counts and autocorrelation values at hourly lags of vehicle counts for one sensor over a two week period. 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 Figure 3.2(b) shows such stationarity. Applying the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) [34] 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.

 Figure 3.2(a) Figure 3.2

Most of the input parameter values for seasonal ARIMA models tend to be 0, 1, 2, or 3 [32]. Due to this small range of input values the total input space is relatively small (Six

(a) Seasonal difference counts   (b) Autocorrelation of seasonal difference counts
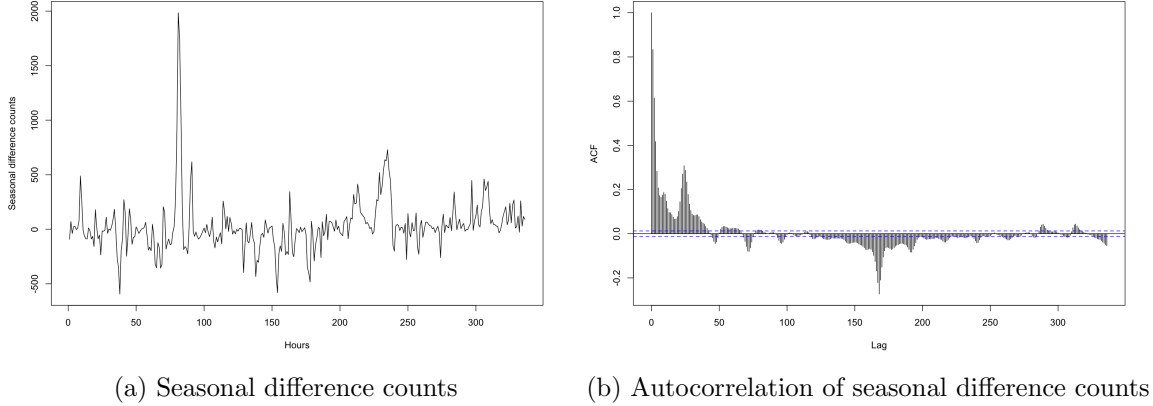
Figure 3.2: One week seasonal difference counts and autocorrelation over a two week period.

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) [35]. Our optimal model is a seasonal ARIMA $(1, 0, 1)(0, 1, 1)_{168}$.

Figure Figure 3.3 shows an example of one-step ahead prediction performed on a sample week of test data. 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. This pattern of mis-forecasting is inline with the pattern demonstrated in Figure **??**. The mean absolute percentage error (MAPE) for this week was approximately 8.2%. This MAPE is close to the results from other authors on other vehicle traffic datasets [25, 36].

3.4 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 [31], 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: INSERT IMAGE OF HISTORIC AVERAGE
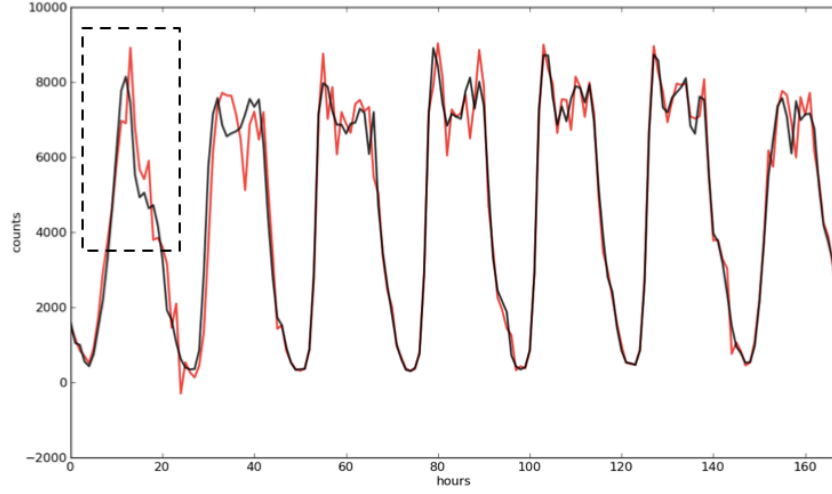
Figure 3.3: One-step ahead prediction for a sample week. Black line is original data. Red line is forecasted data. Dotted box shows an example of mis-forecasting due to a broncos game.

3.5 Time delayed neural networks

Time delayed neural networks are a special subset of regression neural networks where the input data is a local history of data from the time series. 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:

$$T_{t+1} = \phi\{\sum_{j=1}^{J} w_j \psi_j \left[\sum_{l=0}^{m} w_{ji} T_{t-l\delta} + w_{j0} \right] + w_0\} \tag{3.11}$$

where $\phi()$ is a linear activation function on the output layer and $\psi()$ is the standard sigmoid function. A visual representation of the node architecture of a time delayed neural network is displayed in Figure Figure 3.4.

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 inputs into the next forecast.

The number of input nodes and hidden nodes for each dataset is given in Table Table 3.2.

26
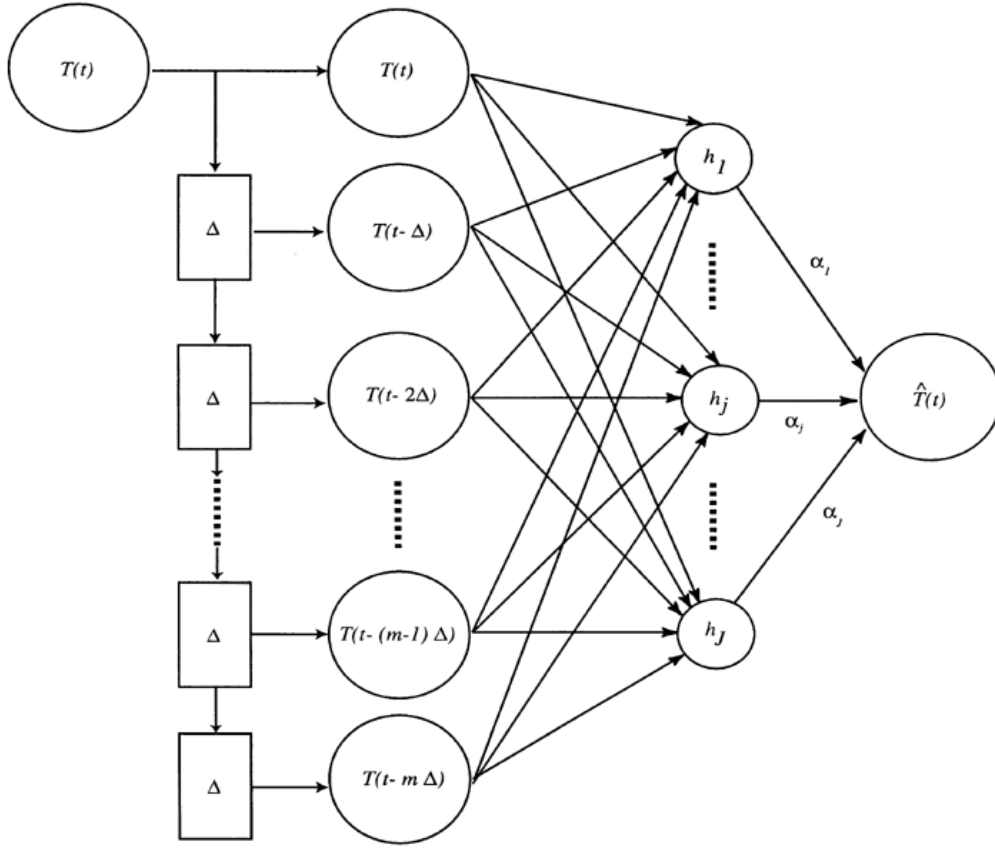
Figure 3.4: Architecture of a time delayed neural network with $m + 1$ inputs and J outputs [37].

TODO: INSERT IMAGE OF TDNN

3.6 Support Vector Regression

Support Vector Machine Regression (SVM) offers a powerful way to forecast time series. It has been used in the past successfully to forecast travel times for vehicle traffic [38].

As training SVM's 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 window. For a fixed window of size w, training input data is of the form $\{T_t, T_{t-1}, ..., T_{t-w+1}\}$. Target data is of the form T_{t+1}. Thus the training examples which we provided to our SVM was $\{T_{t+1}, [T_t, T_{t-1}, ..., T_{t-w+1}]\}$.

To perform SVM training we used the popular *libsvm* package and as parameter selection is a notoriously difficult problem for SVM. We followed the guidelines as outlined by

Table 3.2: 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

Table 3.3: RMSE forecast values per model for a horizon equal to one.

Dataset	ARIMA	TDNN	AVG	SVM	BCF	BCF-TS
MERL	5.5	2.67	4.35	2.29	2.28	**2.26**
CSMBB	10.94	14.13	27.14	11.01	8.72	**8.72**

Hsu, Chih-Chang and Lin, creators of the libsvm package [39]. 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 a factor to determine performance of those parameters.

For both the MERL and CSMBB datasets we used a window of 5. This happened to be the same window length used by [38].

Table 3.3

TODO INSERT IMAGES OF SVR

CHAPTER 4

ENSEMBLE APPROACH

write about approach here

4.1 Bayesian Combined Forecasting

The BCF approach [40] 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.

This section derives the BCF approach for readers unfamiliar with it and then describes some modifications of the approach which improves its performance for our application.

4.2 Notation

We define the time series dataset used in these models as $\{T_t^{(m)}\}$. In our application the data used for these models comes from a set of M binary infrared sensors. Each T_t^m is a 10 minute aggregate of the readings from sensor m reading at time block t.

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

$$\bar{T}_{t+1}^{k,m} = f(T_t, ..., T_1; \theta_k). \tag{4.1}$$

Thus the forecast of T_{t+1} is a function of all past data and some trained parameterization θ_k for that model.

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. Also for this work we forecast
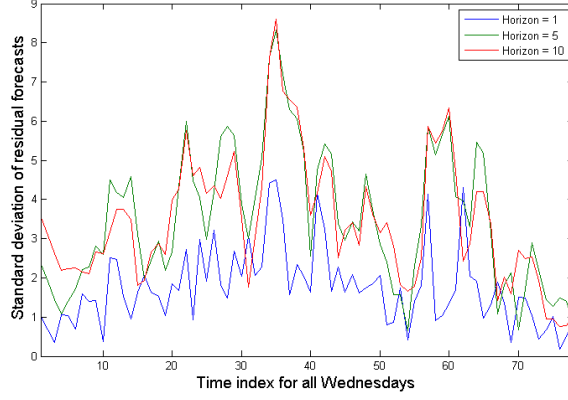
Figure 4.1: Standard deviation of support vector machine residuals for all Wednesdays in MERL dataset. Time index represents 10 minute intervals from 6:00am to 7:00pm.

a model for each individual sensor and for convenience drop the m from our forecasting notation. An example of a forecast two time steps ahead of current time t is given by

$$\bar{T}_{t+2}^k = f(\bar{T}_{t+1}, T_t, ..., T_1; \theta_k). \tag{4.2}$$

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

4.3 Bayesian Combined Forecasting Derivation

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 [40] and define p_t^k as follows

$$p_t^k = p(z = k | T_t, ..., T_1). \tag{4.3}$$

From here we apply Bayes rule and get

$$p_t^k = \frac{p(T_t | z = k, T_{t-1}, ..., T_1) \cdot p(z = k | T_{t-1}, ..., T_1)}{p(T_t, ..., T_1)}. \tag{4.4}$$

Notice that $p(z = k | T_{t-1}, ..., T_1) = p_{t-1}^k$. Thus we can create a recursive estimation based on prior p_t^k.

30

With recursive values for p_t^k and replacing $p(T_t, ..., T_1)$ with a conditional probability on z we get

$$p_t^k = \frac{p(T_t|z = k, T_{t-1}, ..., T_1) \cdot p_{t-1}^k}{\sum_{j=1}^K p(T_t|z = j, T_{t-1}, ..., T_1) \cdot p_{t-1}^j}. \tag{4.5}$$

We use the empirically observed forecasting error for each model to estimate $p(T_t|z = k, T_{t-1}, ..., T_1)$. The forecasting error for a given model at time t is

$$e_t^k = \bar{T}_t^k - T_t. \tag{4.6}$$

We can use these forecasting errors to estimate a probability distribution for each model on the random variable e_t^k. This is typically modeled as a white noise zero mean Gaussian process. For our work, we represent this as a distribution of error terms with some parameterization ω_k. Thus for each model the probability error distribution function on the model error random variable is given by $q(e_t^k; \omega_k)$.

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

$$p_t^k = p(z = k|T_t, ..., T_1) = \frac{p_{t-1}^K \cdot q(T_t - \bar{T}_t^k; \omega_k)}{\sum_{j=1}^K p_{t-1}^j \cdot q(T_t^j - \bar{T}_t^j; \omega_j)}. \tag{4.7}$$

An example of these changing normalized posterior probabilities for a small section of the MERL dataset is shown in Figure Figure 4.2.

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

$$T_{t+\delta}^{ALL} = \sum_{k=1}^K p_t^k \cdot \bar{T}_{t+\delta}^k. \tag{4.8}$$

4.4 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
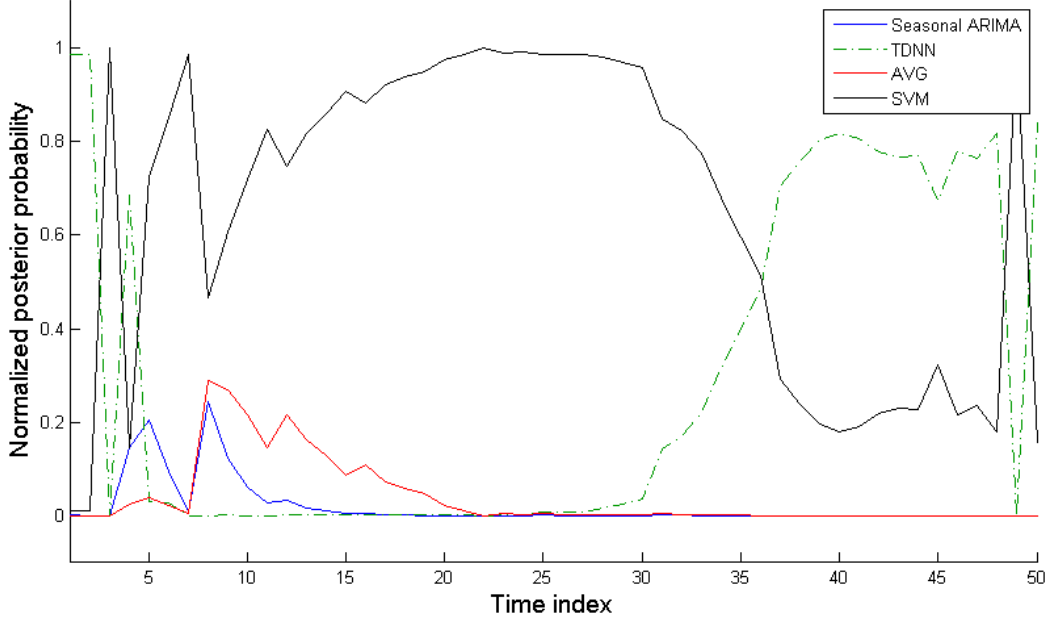
Figure 4.2: Normalized posterior probabilities of component models on a section of MERL dataset.

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.

4.4.1 Forecast δ time steps into the future

Traditional implementations of BCF in other domains [40, 41] 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. Instead of generating a model's error distribution from

$$e_k^t = \bar{T}_t^k - T_t = f(\bar{T}_t, T_{t-1}..., T_1; \theta_k) - T_t. \tag{4.9}$$

The error distribution is instead generated from

$$e_k^t = f(\bar{T}_t, ..., \bar{T}_{t-\delta+1}, T_{t-\delta}, ..., T_1; \theta_k) - T_t. \tag{4.10}$$

The reason for this change is due to the assumption that our error distribution is an accurate representation forecasting accuracy. The forecasting error distribution for models at 1 time step into the future is not necessarily the same as models at δ time steps. Thus we compute

a different error distribution for each forecast time step.

4.4.2   Improving model error distributions

Despite other implementations of BCF using fixed error distributions, our data has clear daily trends. For some of our models, the forecasted residuals follow these same trends. See Figure Figure 4.1 for an example of how the forecasting error distribution for a trained support vector regression model on the MERL dataset depends on the time and on the forecasting horizon.

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 10 minute slice of a given day. 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. For example taking the data for each time slice for each Wednesday results in 78 Gaussian error distributions for each forecasting horizon. These Gaussians are computed from a validation set representing 20% of our data. It is from this set of models 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 [32] 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.

4.4.3   Model selection thresholding

Diebold [42] 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 when such cases where no model is able to provide an accurate forecast, then it is often the case that forecasts come from the worst model.

To combat this case, 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

$$p(2\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-2}. \tag{4.11}$$

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.

4.5 Results of Ensemble

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. All results shown below are on the test set only. Figure Figure 4.3 shows an 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.

It is common for one model's normalized posterior probability to be near one when that model is currently accurate. Figure 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 on
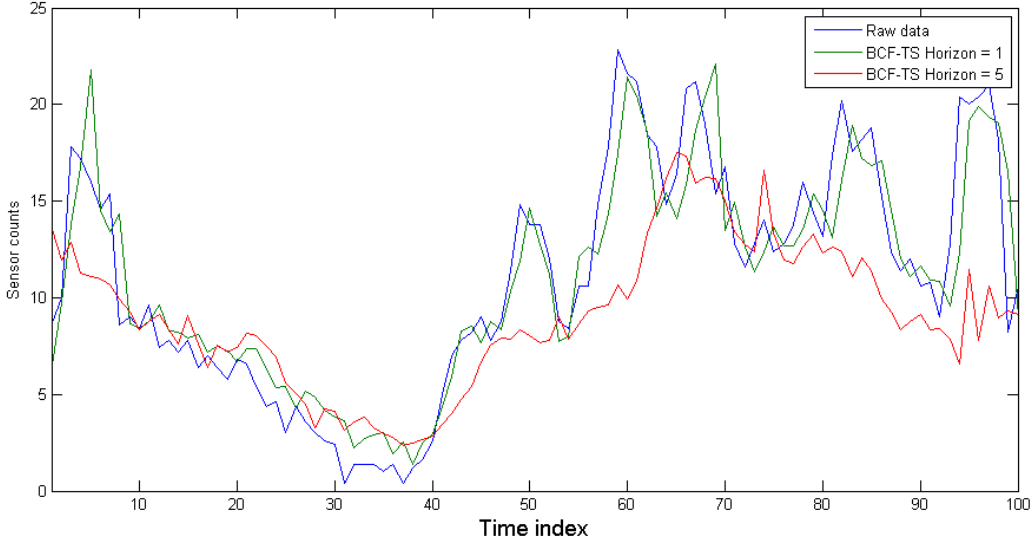
Figure 4.3: A comparison of forecasts at various horizons against real data for an sample time segment using BCF-TS.

the model's posterior probability is lower and as a result BCF-TS uses other model for its combined forecast.

Figure Figure 4.5 shows the results of the root mean squared error (RMSE) of forecasts across a forecast horizon up to 10 time steps (100 minutes) into the future for each model. These plots show that BCF-TS has the lowest error. However, the average model shows itself to be a strong indicator of future activity for forecasts beyond 60 minutes into the future. Forecasts were performed for significantly longer horizons, but the results were uninteresting as the total RMSE of models converged to roughly the values at a forecasting horizon of 10 time steps.

Table 4.1: Run times (in seconds) for each forecasting horizon.

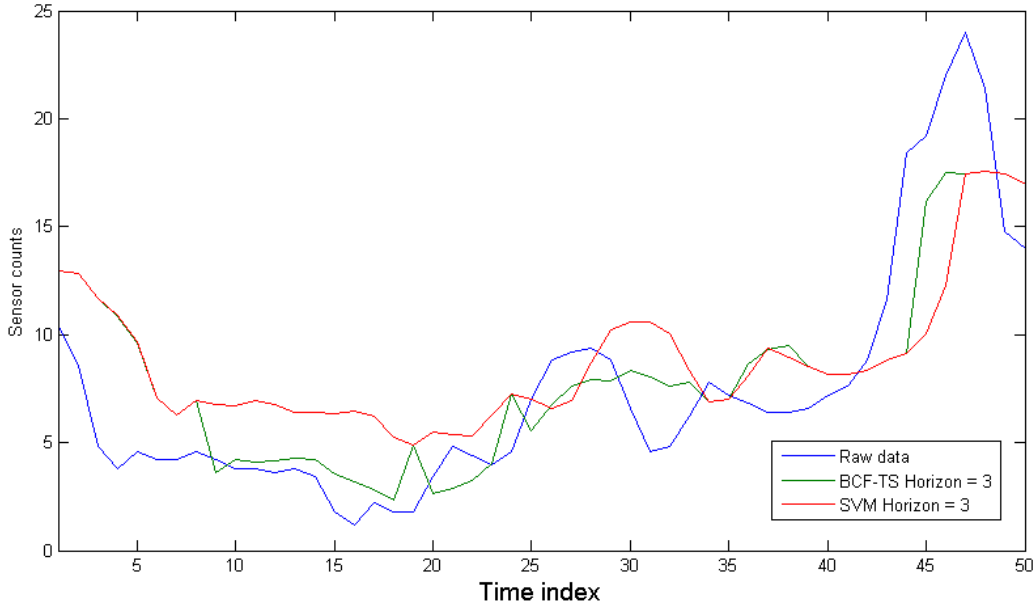| Algorithm	1	2	3	5	8	10
Average	0.001	0.001	0.001	0.001	0.001	0.001
ARIMA	0.043	0.045	0.046	0.053	0.058	0.063
SVM	0.048	0.910	0.137	0.227	0.357	0.444
TDNN	20.87	21.60	21.82	21.28	22.73	22.50
BCF-TS	20.97	21.26	21.27	21.34	21.57	21.63

35

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

In the CSMBB 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 forecasting horizons. This is likely due to a stronger seasonal component to the CSMBB 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 [31]; where in the case of strong seasonal data, results are better for short horizon forecasts, but longer forecasts favor historic averages.

BCF and BCF-TS were both better at a horizon of one time step for all component models (see Table Table 3.3). In the MERL dataset standard BCF was outperformed by SVM and later the average model for all forecast beyond one horizon. However the BCF-TS model showed significant improvement in RMSE scores for all forecasting horizons unto 60 minutes. For horizons of 10 time steps and greater, the average model is about as good as the BCF-TS approach.

Table Table 4.1 shows the run time in seconds of each forecasting algorithm at a given forecasting horizon. The times are for forecasting the entire test set on the MERL dataset for a single sensor, approximately twenty weeks worth of data. In general BCF-TS was slower than any component model, but the times are still such that real-time forecasting is possible.
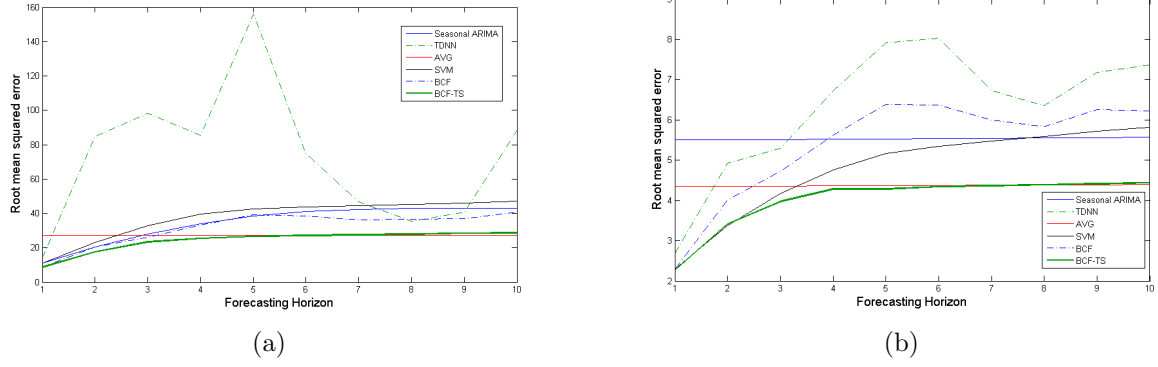

Figure 4.5: Root mean square error of forecasting for each model vs forecasting horizon.

IMPROVED ENSEMBLE

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 describes more in depth some of the problems with BCF. 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 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 misforecasts may come from large human controlled scheduled events (such as sporting events,
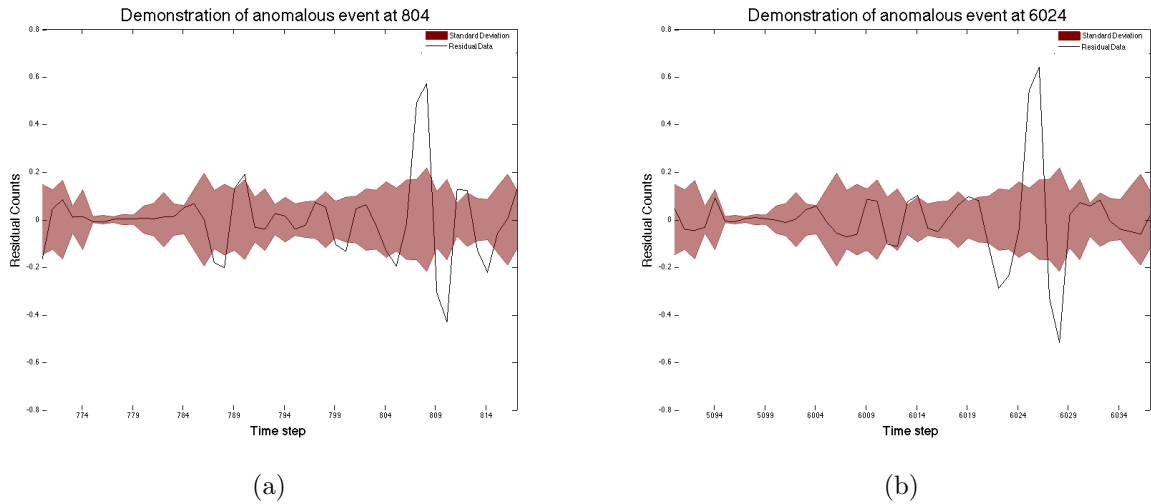

(a)   (b)

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

(a)   (b)

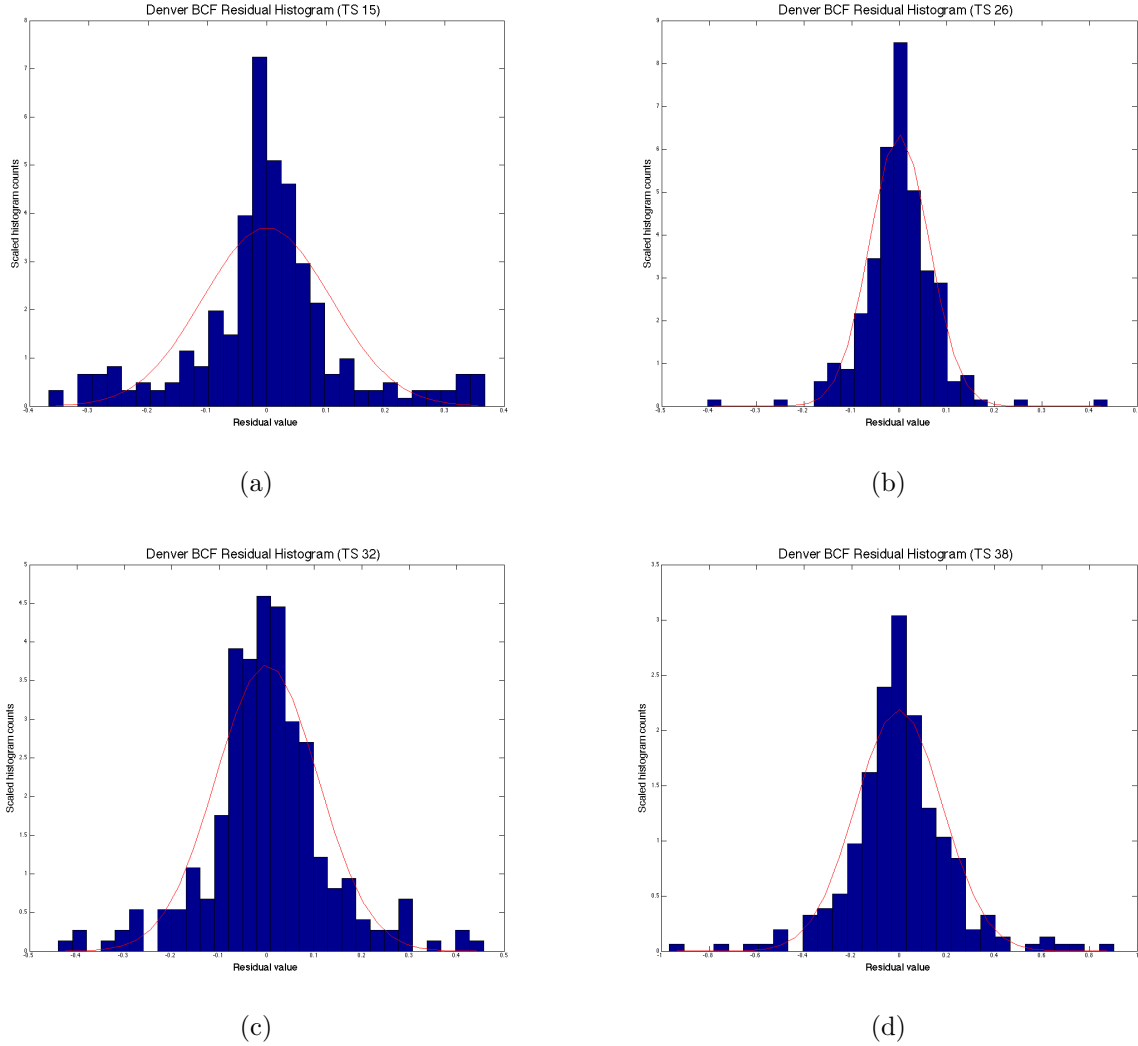(c)   (d)

Figure 5.2: Scaled histogram of IBCF residual values for the Denver traffic dataset at various 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.

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 [43, 44] 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 shows these results graphically. In this figure, a histogram of the data from four different 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 events along evidence that the remaining 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)
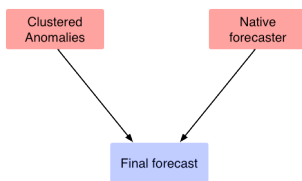

Figure 5.3: Extremly high level overview of our approach

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 forecaster 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 human controlled traffic systems - that 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 5.7 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. Anomaly modeling as used for improved forecasting seems to have limited results in the forecasting literature. We believe that this limited amount of anomaly modeling is often due to the nature of anomalies, either they are random and seemingly often times difficult to predict as is the case with stock market anomalies [45, 46] 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 [47, 48] 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 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."

A common method of anomaly detection is through the use of tools to assist in visual identification [49–51]. 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) [52] 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 is value is declared. A problem with this approach is that the thresholds need to be known a priori.

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 [53]. These algorithms can operate offline, online [47], top down, bottom up and globally. An excellent summary of current state of the art in change point

detection spanning all major formulations (Bayesian, minimax, and generalized Bayesian) is found in [54]

TODO CLEAN UP CHANGE POINT TECHNIQUES

Spectral Methods: Widely used in other fields to distinguish hidden patterns and trends from a noisy background. [55]

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 accelerometers 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 [56, 57]. 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 [57], support vector machines [57–59], naive Bayes [57, 59], nearest neighbor [57, 59], and hidden Markov Models (HMM) [56, 60]. Comparisons amongst models have shown that performance is data dependent and that no one model appears to be best for all types of activities [57, 59]

Huynh [61] 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 [62]. 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 [63] 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 [64]. The performance level of suffix trees seems to be highly noise dependent. [65] 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 data from the sensors is normally binary which leads to ambiguity determining true counts and direction of motion for cars or people. Despite this ambiguity it is relatively easy to automatically construct the topology of the sensors [17, 22]. Empirically it seems as though this topology roughly correlates to the spatial distribution of the sensors. We will use this spatial information later to better model events which occur over multiple sensors.

HMMs have been used as a model for learned activities. These models are used to build a tree [21, 22] 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 [66].

In a method similar to the HMM tree, work has been done to create a hierarchy of fixed filters based on possible sensor topologies [18]. 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 [67] 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 and concludes that his finding demonstrates a fundamental flaw present in many of the activity recognition techniques to be described as the typical approach is to search for an optimal set of activity models for a fixed feature space and window of time.

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.

Sliding window data extraction

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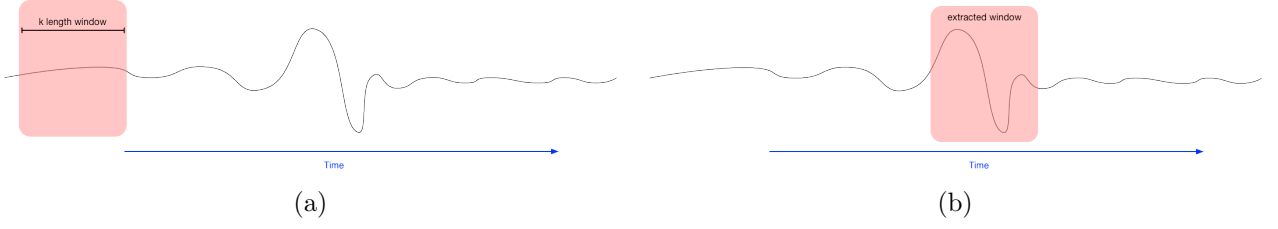
Figure 5.4: Demonstration of a fixed width sliding window looking for locally maximal deviations from background behavior.

of potential candidate 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.

TODO FINISH THIS ALGORITHM

Algorithm 1 Algorithm for candidate data extraction

      $\triangleright$ %comment: servers[] contains the index of servers whose data rate are sorted in descending order%
servers[]= index(of all servers)
serverIndex[]=servers[0..K]
linearlyIndependentServerIndex[]=0
$[Z] \leftarrow 0$
for $i = 0$ to $serverIndex.length$ **do**
$\triangleright$ %comment: find the equation corresponding the serverIndex from the mapping at the File Server%
    $eqn = equation(serverIndex[i])$
                                   $\triangleright$ %comment: try insert equation into Z using OFG%
end forend for
while (linearlyIndependentServerIndex.length!=K) **do**
        $\triangleright$ %comment: remove all the server index which were not inserted in Z%
    temp[]=serverIndex[]-linearlyIndependentServerIndex
    if (linearlyIndependentServerIndex.length=K) **then**
        break
    end if
end while

Algorithm 1 describes precisely 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

Future work should allow for anomaly models which are non-constant in length. An interesting possibility for the extraction and molding sparse dictionary learning. For this work however, we believe that constant length anomaly extraction is sufficient to show the strength of our combined forecasting approach.
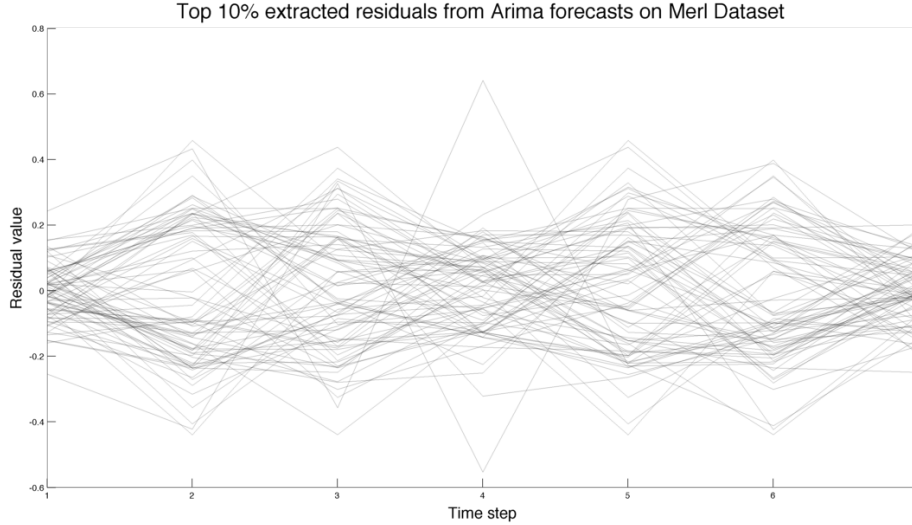

Figure 5.5: Extracted residuals from the Merl dataset using the sliding window extraction method with window length of 7

5.5 TSMG derivation

TODO Move this derivation to the APPENDIX AND JUST DISCUSS OUR MODEL APPROACH HERE.

Discuss the silhouette score for best fit clusters.

A mixture of Gaussians is a strongly supported stochastic data clustering technique used in activity recognition. Traditionally, a mixture of Gaussians is implemented for either a one dimensional time series (CITE PAPER TO SUPPORT THIS) or for data vectors with no time element. Here we combine the two approaches, creating a mixture of Gaussians for multi-dimensional time series data. While this approach has not yet been implemented, based on the success of mixture of Gaussians in other domains, we expect good results.

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 $\{x^{(i)}\}$ we maximize

$$\ell(\theta) = \sum_{i=1}^{\mathbf{M}} log\{p(x^{(i)}|\theta)\} \tag{5.1}$$

where $\mathbf{M}$ is the total number of time series instances.

The expectation maximization (EM) algorithm is commonly used to maximize dataset likelihood. To use this algorithm we need to define a set of variables

$$w_k^{(i)} = p(z = k|x^{(i)}) \tag{5.2}$$

where $\mathbf{K}$ is the total number of Gaussians to train and k is an index of $\mathbf{K}$.

The general equation for the likelihood of the models is:

$$\ell(\theta|x) = \sum_{i=1}^{\mathbf{M}}\sum_{k=1}^{\mathbf{K}} w_k^{(i)} \log\{\frac{p(x^{(i)}|z = k)p(z = k)}{w_k^{(i)}}\} \tag{5.3}$$

In the traditional mixture of Gaussians algorithm each model is ostensibly a Gaussian. To make this algorithm work with multi-dimensional time series, we define the models instead by

$$p(x^{(i)}|z = k) = \prod_{n=1}^{\mathbf{N}} \mathcal{N}_n(x^{(i)}) \tag{5.4}$$

where $\mathbf{N}$ is the length of each time series instance. Thus our model for each time series is $\mathbf{N}$ independent multivariate Gaussians.

Combining equations 5.3 and 5.4 gives the following log likelihood

$$\ell(\theta|x) = \sum_{i=1}^{\mathbf{M}}\sum_{k=1}^{\mathbf{K}} w_k^{(i)}\{\log \frac{p(z = k)}{w_k} + \sum_{n=1}^{\mathbf{N}} \log \mathcal{N}_n(x^{(i)})\} \tag{5.5}$$

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

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

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 θ. The results of these maximizations are given below:

$$\theta_k = \frac{1}{\mathbf{M}} \sum_{i=1}^{\mathbf{M}} w_k^{(i)} \tag{5.7}$$

$$\mu_{k,n} = \frac{\sum_{i=1}^{\mathbf{M}} w_k^{(i)} x_n^{(i)}}{\sum_{i=1}^{\mathbf{M}} w_k^{(i)}} \tag{5.8}$$

$$\Sigma_{k,n} = \frac{\sum_{i=1}^{\mathbf{M}} w_k^{(i)} (x^{(i)} - \mu_{k,n})(x^{(i)} - \mu_{k,n})^{\mathrm{T}}}{\sum_{i=1}^{\mathbf{M}} w_k^{(i)}} \tag{5.9}$$

5.6   Sample representative clusters and a discussion about them

This section visually demonstrates a few of the clusters extracted from our datasets. These cluster sets were chosen from each dataset and from three of our base forecasting approaches. The structure from these clusters again leads further credibility to the residual data containing some structure.

Figure 5.6 is a set of clusters from the Merl dataset extracted from the SVM forecaster at a horizon of 20 minutes. The extracted window length is 7 time steps (past 70 minutes). These clusters were taken from the top 10% of the MERL data residuals.

Figure 5.7 is the set of clusters from the Brown dataset extracted from the Arima forecaster. The clusters from the dataset were taken from the top 15% of the Brown data residuals. Notice how these clusters, unlike the clusters from Figure 5.6 do not begin nor end on zero. This likely means that the true residuals are longer in length and those extracted. 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 5.8 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. Unlike in the extracted clusters of MERL and Brown, these clusters appear to have "sharp"
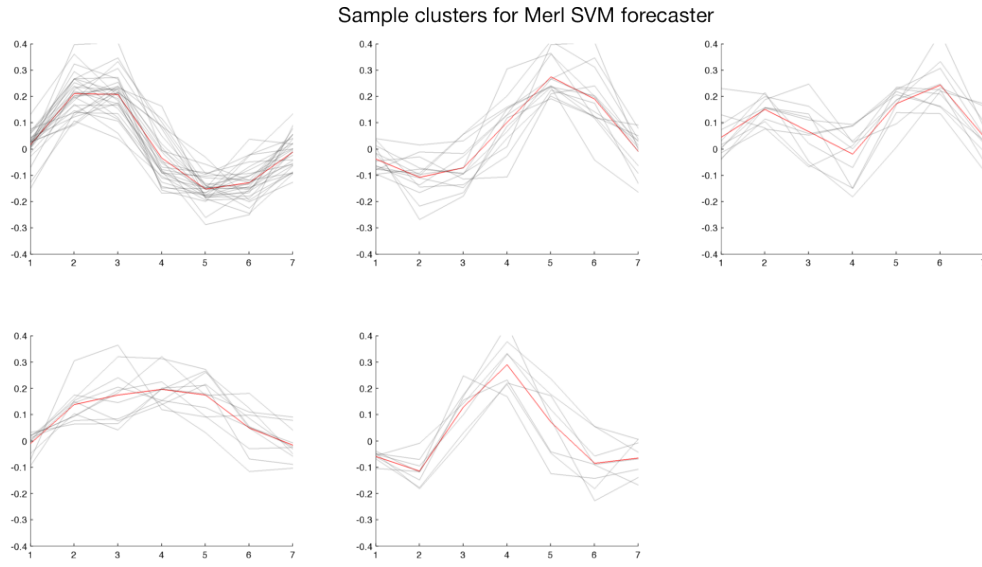
Figure 5.6: Extracted residuals from the Merl dataset. Data was taken from the top 10% of the SVM forecaster's residuals.
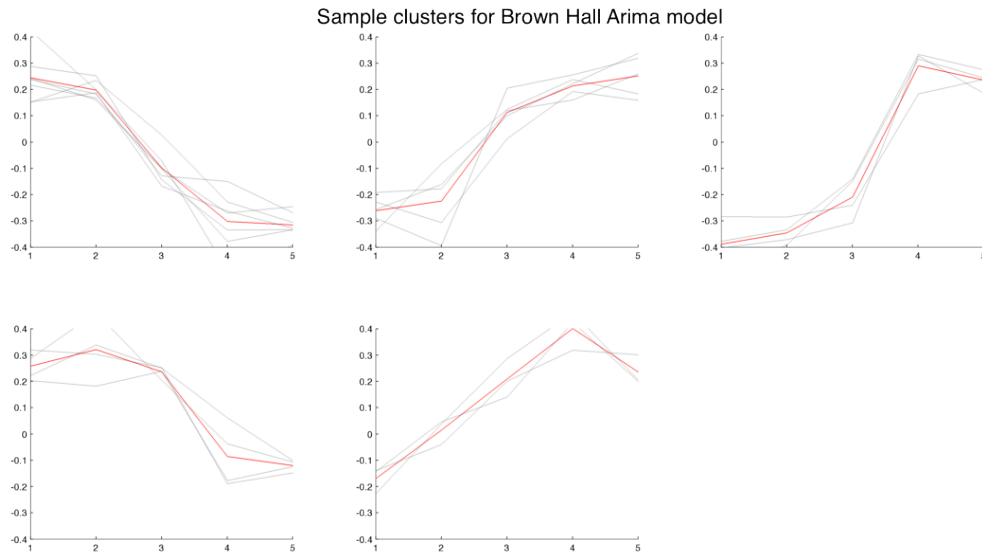

Figure 5.7: Extracted residuals from the Brown dataset. Data was taken from the top 15% of the Arima forecaster's residuals.
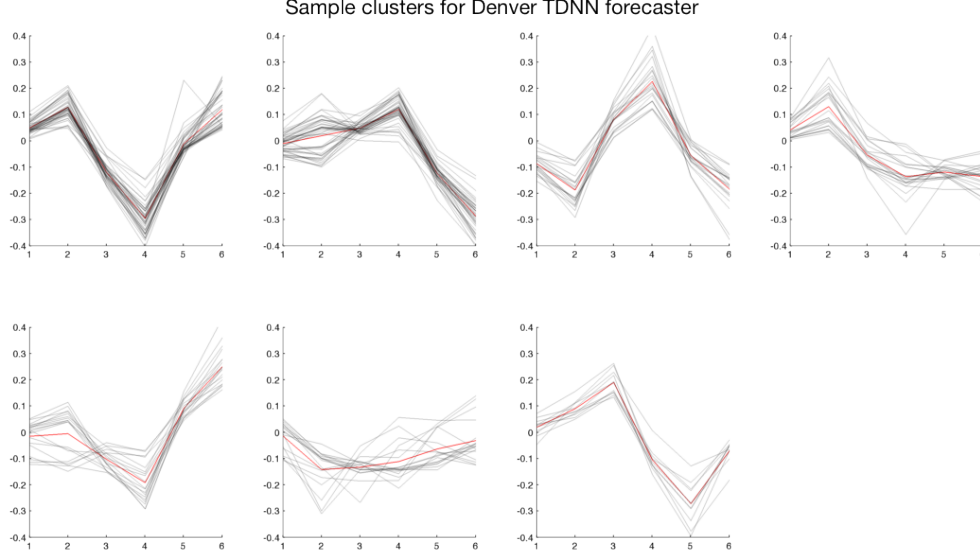
Figure 5.8: Extracted residuals from the Denver dataset. Data was taken from the top 10% of the TDNN forecaster's residuals.

transition. This is likely due to the data being aggregated every 30 minutes instead of every 10 minutes like the other dataset. Also, some of the clusters begin or end with a 0 residual, but others drift higher or lower. Such a clustering is likely a scenario where modeling these clusters through a non-fixed technique may lead to better results.

5.7 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 $p(c_t|x_t)$.

51

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

$$\sum_i^{len(a)} p(a_t^{(i)}|x_t,\ldots,x_1) + \sum_i^{len(b)} p(b_t^{(i)}|x_t,\ldots,x_1) + p(c_t|x_t,\ldots,x_1). \qquad (5.10)$$

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

$$p(a_{t+1}^{(i+1)}|x_{t+1},\ldots,x_1) = \frac{p(x_{t+1}|a_{t+1}^{(i+1)},x_t,\ldots,x_1)p(a_{t+1}^{(i+1)}|x_t,\ldots,x_1)}{p(x_{t+1})} \qquad (5.11)$$

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

$$
\begin{aligned}
p(x_{t+1}) = &\sum_i^{len(a)-1} p(x_{t+1}|a_{t+1}^{(i+1)},x_t,\ldots,x_1)p(a_{t+1}^{(i+1)}|x_t,\ldots,x_1)+ \\
&\sum_i^{len(b)-1} p(x_{t+1}|b_{t+1}^{(i+1)},x_t,\ldots,x_1)p(b_{t+1}^{(i+1)}|x_t,\ldots,x_1)+ \\
&p(x_{t+1}|c_{t+1},x_t,\ldots,x_1)p(c_{t+1}|x_t,\ldots,x_1).
\end{aligned}
\qquad (5.12)
$$

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
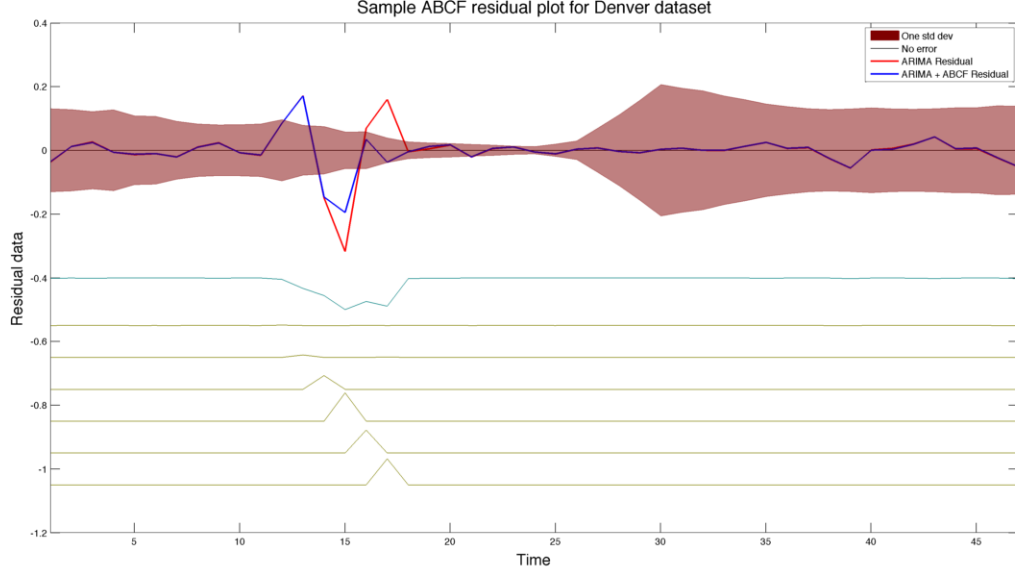
Figure 5.9: Sample residual and the respective probabilities of each time step for the base forecaster and the a single activity model.

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

$$p(a_{t+1}^{(i+1)}|x_t, \ldots, x_1) = \begin{cases} p(a_t^{(i)}|x_t, \ldots, x_1) & : i < len(a) \\ 0 & : otherwise. \end{cases} \tag{5.13}$$

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.

$$\begin{aligned} p(a_{t+1}^{(1)}|x_t, \ldots, x_1) = p(a_{t+1}^{(1)}) = \theta_a \\ p(b_{t+1}^{(1)}|x_t, \ldots, x_1) = p(b_{t+1}^{(1)}) = \theta_b \end{aligned} \tag{5.14}$$

5.8 Demonstration of ABCF

Figure 5.9 is a demonstration of ABCF. This figure displays one 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

53

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 individual likelihoods of each time step of a given clustered anomaly Figure 5.10

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 15. 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 12, there is a slight perturbation in the top 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

Figure 5.10: Extracted sample clustered anomaly.

the background model is once again the most prevalent model amongst all the extracted anomalies.

Figure 5.9 displays a significant improvement in SQEONAN. 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 5.11 shows the final result of this forecast window. This figure shows clearly how ABCF improves an ARIMA forecaster for this data segment during the later parts of an anomaly.
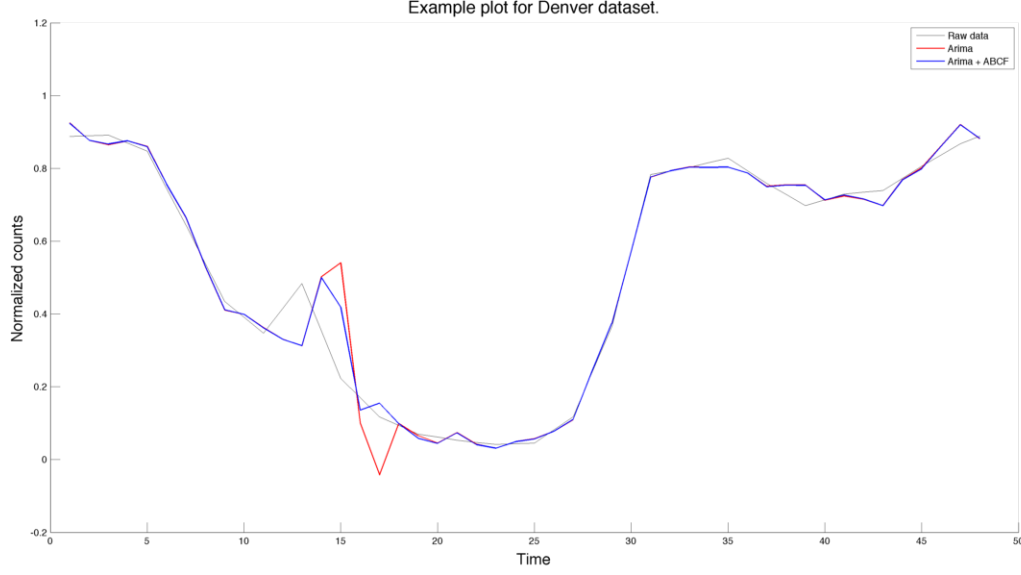
Figure 5.11: Two horizon time step results when applying ABCF to an ARIMA forecaster for a segment of the Denver dataset.

5.9 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.

Results per forecaster

Figure 5.12 demonstrate the improvements to the SQEONAN metric of applying our ABCF algorithm to each dataset for each activity model. A set of sample RMSE and MASE results are given in Figure 5.13 and Figure 5.14. In most of these figures, ABCF improves or is equal to the forecasting algorithm on its own for each forecasting horizon up to 15 time steps in the future. Maximum total improvement is typically achieved between two and four time steps into the future. Maximum percentage improvement is typically between one and two time steps into the future.

As ABCF is an ensemble forecaster, it can make no guarantees on its improvement. Figure 5.12(d) and Figure 5.14(d) demonstrate cases where ABCF performed worse than

simply the base forecaster. For forecasts one time step ahead 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 5.13 and Figure 5.14 demonstrates the results of ABCF applied to the same some of the same datasets from Figure 5.12. Notice that in most cases, there are significantly smaller improvements in both MASE and RMSE than that measured with SQEONAN. This result is expected. ABCF improves forecasting 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 5.13(b) and (d) show almost no difference.

(a)

(b)

(c)

(d)
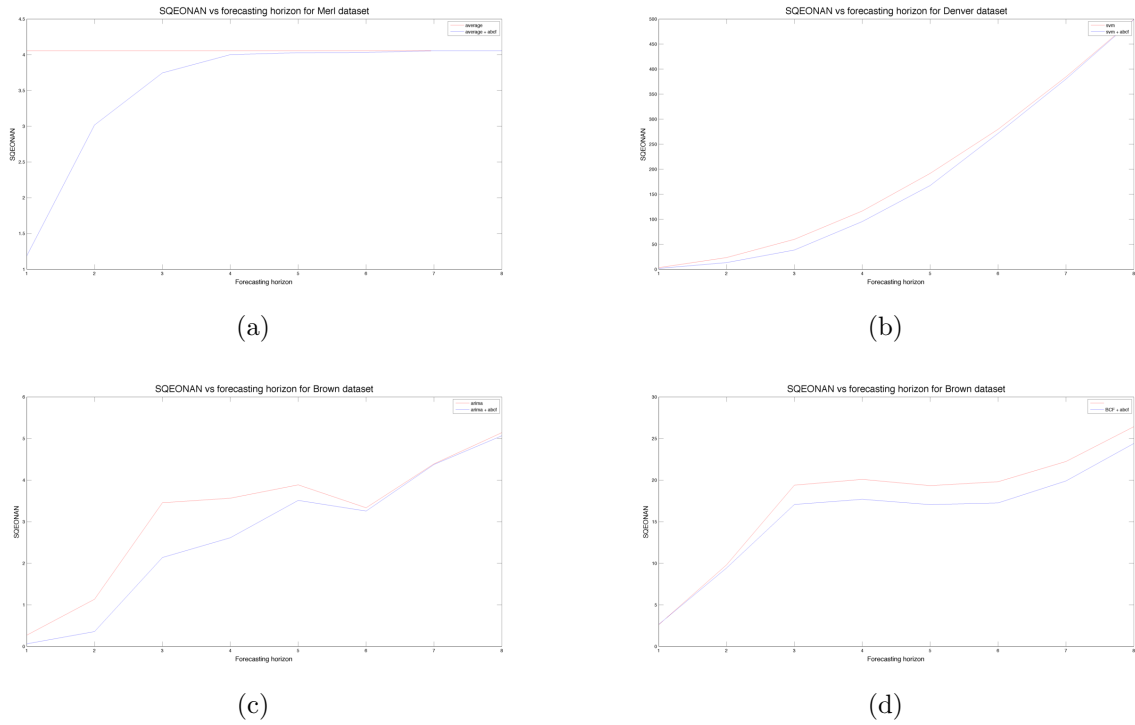
Figure 5.12: SQEONAN results of four sample forecasters with and without ABCF
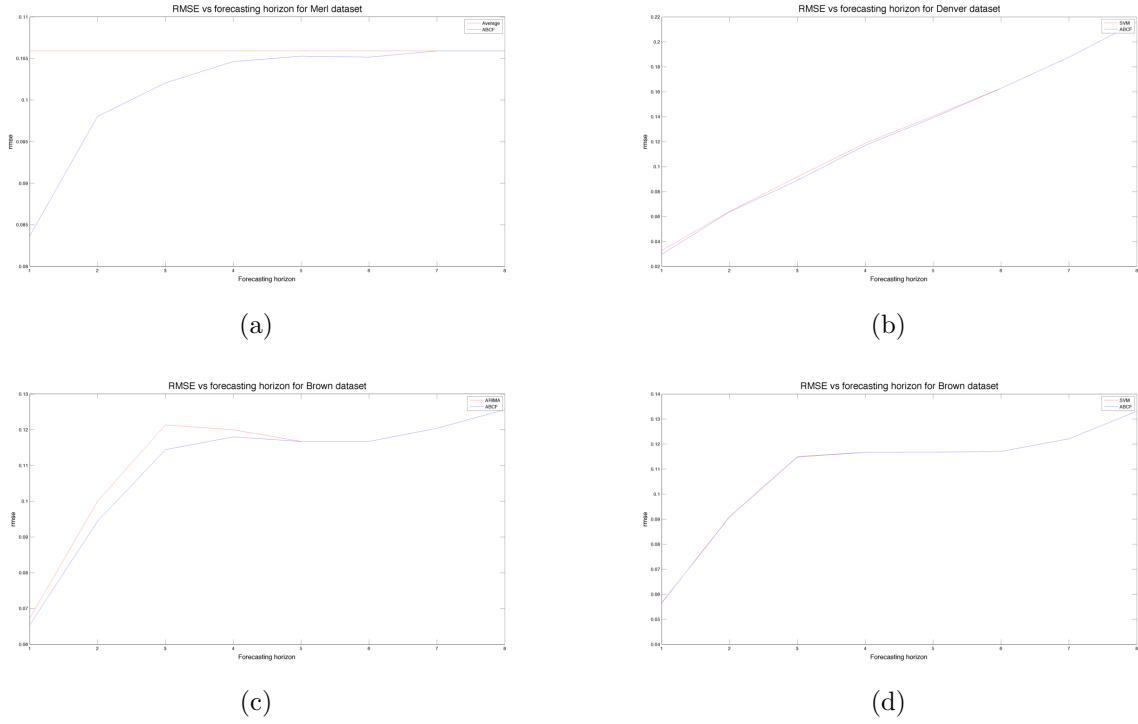
Figure 5.13: RMSE results of four sample forecasters with and without ABCF

Aggregated Results

Figure 5.15 shows the percentage improvement 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.

Both BCF and our IBCF algorithm introduced in the last chapter appear to have the smallest margin for improvement. In the case of the Denver dataset, forecasts more than 10 time steps in the future are negatively affected by 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. (TODO back this up with silhouette scores)

(a)　　　　　　　　　　　　　　　(b)

(c)　　　　　　　　　　　　　　　(d)
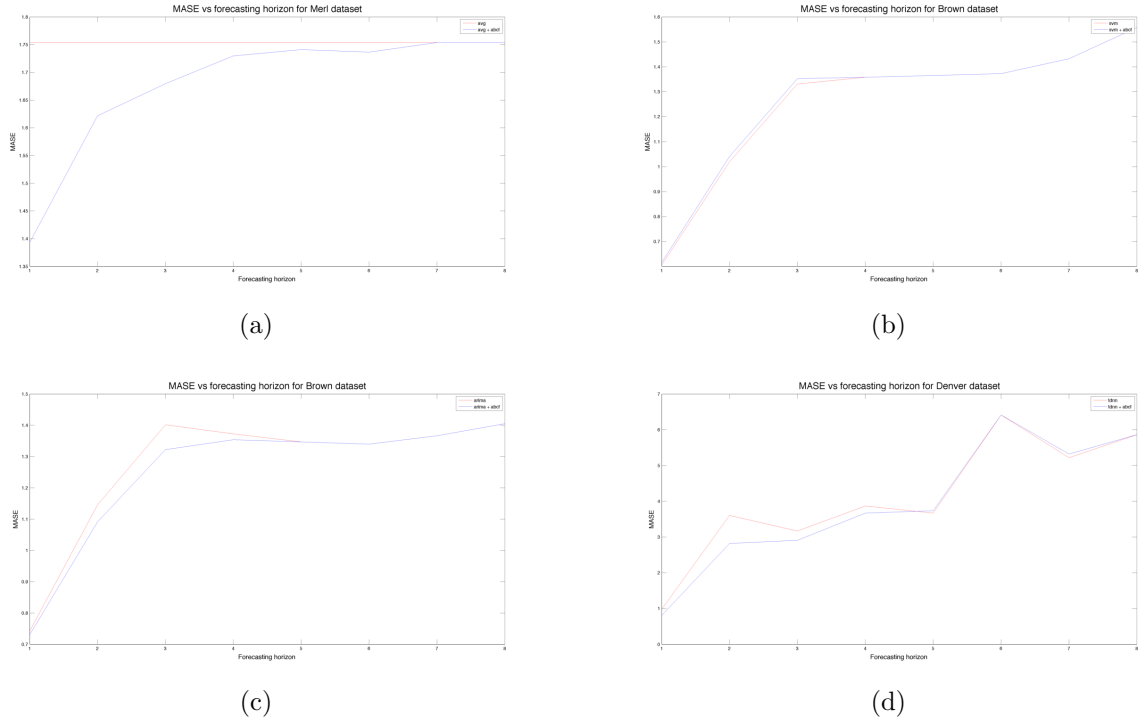
Figure 5.14: MASE results of four sample forecasters with and without ABCF

TODO include selected RMSE and MASE results

TODO include overall average improvement

TODO include results of sqeonan with ABCF improvements vs a few other forecasters

5.10 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.

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 o continue, then trimming or growing the sliding 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.

(a)

(b)

(c)
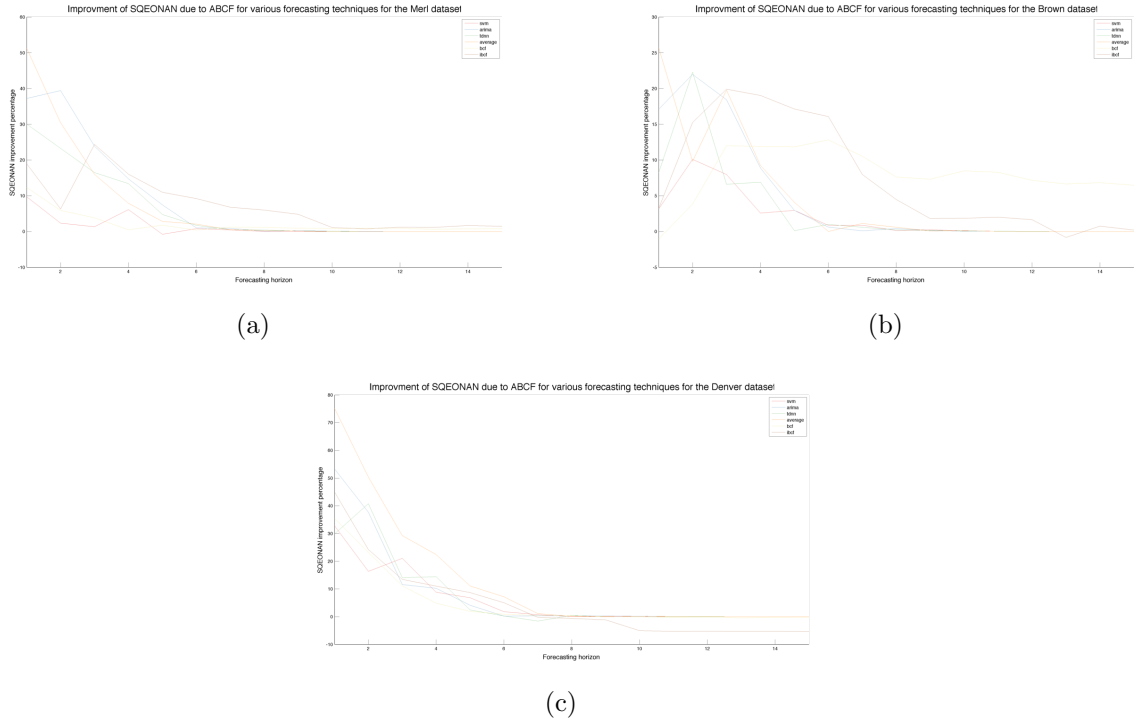
Figure 5.15: Percentage improvement to SQEONAN due to application of ABCF
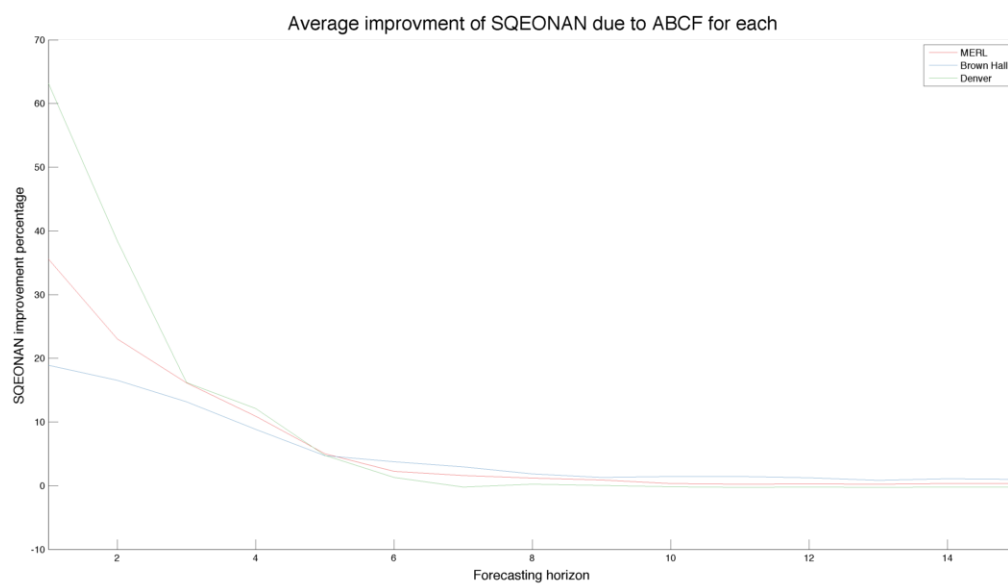
Figure 5.16: Average percent improvement of applying ABCF to each forecaster to a given dataset.

CHAPTER 6
CONCLUSION

Discuss our conclusions here

REFERENCES CITED

[1] US DOE. Building Energy Databook, 2010. URL `http://buildingsdatabook.eren.doe.gov/`.

[2] KMC Controls. Understanding Building Automation and Control Systems, 2013.

[3] KMC Controls. Zone Control with Variable Air Volume Controls (VAV), 2013.

[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. doi: http://dx.doi.org/10.1016/j.buildenv.2003.08.003.

[5] Yudong Ma, Francesco Borrelli, and Brandon Hencey. Model predictive control for the operation of building cooling systems. In *America Control Conference*, 2010. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5739562`.

[6] US DOT. National Traffic Signal Report Card, 2007. URL `http://www.ite.org/reportcard/NTSRCExecSummaryfinal.pdf`.

[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] 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.

[10] 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. doi: 10.1145/1878537.1878725. URL `http://portal.acm.org/citation.cfm?doid=1878537.1878725`.

[11] 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. doi: 10.1145/1878431.1878433. URL `http://portal.acm.org/citation.cfm?doid=1878431.1878433`.

[12] 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. URL `http://dl.acm.org/citation.cfm?id=2343582`.

[13] 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. URL `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+Sensor-Utility-Network+Method+for+Estimation+of+Occupancy+Distribution+in+Buildings#0`.

[14] 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. URL `https://merl.com/reports/docs/TR2007-069.pdf`.

[15] 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.

[16] 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. URL `http://dl.acm.org/citation.cfm?id=2501233`.

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

[18] Christopher R. Wren and E. Tapia. Toward scalable activity recognition for sensor networks. *Location-and Context-Awareness*, pages 168–185, 2006. URL `http://www.springerlink.com/index/7114556523154474.pdf`.

[19] 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.

[20] 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. doi: 10.1145/2107596.2107613. URL `http://dl.acm.org/citation.cfm?doid=2107596.2107613`.

[21] 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. doi: 10.1109/AFGR.2004.1301600. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1301600`.

[22] 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. doi: 10.1016/j.patcog.2006.04.009. URL `http://linkinghub.elsevier.com/retrieve/pii/S0031320306001610`.

[23] 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. URL `http://trb.metapress.com/index/W863M1341762TU51.pdf`.

[24] 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. doi: 10.1016/S0968-090X(02)00009-8. URL `http://linkinghub.elsevier.com/retrieve/pii/S0968090X02000098`.

[25] 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. doi: 10.1061/(ASCE)0733-947X(2003)129:6(664). URL `http://link.aip.org/link/JTPEDI/v129/i6/p664/s1&Agg=doi`.

[26] 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. URL `http://www.akademik.unsri.ac.id/download/journal/files/waset/v51-40.pdf`.

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

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

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

[30] 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. doi: 10.1016/j.amc.2011.01.073. URL `http://linkinghub.elsevier.com/retrieve/pii/S0096300311001081`.

[31] 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.

[32] 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.

[33] 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.

[34] 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. URL `http://www.sciencedirect.com/science/article/pii/030440769290104Y`.

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

[36] Brian L. Smith and Michael J. Demetsky. Traffic Flow Forecasting: Comparison of Modeling Approaches. *Journal of Transportation Engineering*, 123(4)(August):261–266, 1997. URL `http://link.aip.org/link/?JTPEDI/123/261/1`.

[37] 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.

[38] 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. doi: 10.1109/TITS.2004.837813. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1364002`.

[39] CW Hsu, CC Chang, and CJ Lin. A practical guide to support vector classification. (1):1–16, 2003. URL `https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf`.

[40] 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. URL `http://www.springerlink.com/index/X6UR20K8780000T2.pdf`.

[41] 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. doi: 10.1061/(ASCE)0733-947X(2006) 132:2(114). URL `http://link.aip.org/link/JTPEDI/v132/i2/p114/s1&Agg=doi`.

[42] 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.

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

[44] 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.

[45] A Kontonikas and A Kostakis. On monetary policy and stock market anomalies. *Journal of Business Finance & . . .*, 2013. URL `http://core.kmi.open.ac.uk/download/pdf/9995047.pdfhttp://onlinelibrary.wiley.com/doi/10.1111/jbfa.12028/full`.

[46] 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.

[47] 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. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6380529`.

[48] 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. doi: 10.1093/comjnl/bxr026. URL `http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/bxr026`.

[49] 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. doi: 10.1145/2517957.2517966. URL `http://dl.acm.org/citation.cfm?doid=2517957.2517966`.

[50] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 34(4):219, October 2004. ISSN 01464833. doi: 10.1145/1030194.1015492. URL `http://portal.acm.org/citation.cfm?doid=1030194.1015492`.

[51] 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. URL `http://users.soe.ucsc.edu/~pang/visweek/2012/vast/challenge/shi.pdf`.

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

[53] Arnaud Dessein and Arshia Cont. *Online change detection in exponential families with unknown parameters*. Springer Berlin Heidelber, 2013. URL `http://link.springer.com/chapter/10.1007/978-3-642-40020-9_70`.

[54] AS Polunchenko and AG Tartakovsky. State-of-the-art in sequential change-point detection. *Methodology and Computing in ...*, pages 1–34, 2012. URL `http://link.springer.com/article/10.1007/s11009-011-9256-5`.

[55] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment - IMW '02*, page 71, 2002. doi: 10.1145/637201.637210. URL `http://portal.acm.org/citation.cfm?doid=637201.637210`.

[56] 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. URL `http://www.springerlink.com/index/w455252278468kl2.pdf`.

[57] Ling Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004. URL `http://www.springerlink.com/index/9AQFLYK4F47KHYJD.pdf`.

[58] 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. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4518365`.

[59] Mitja Luštrek and Boštjan Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33(2):205–212, 2009. URL `http://dis.ijs.si/mitjal/documents/Fall_detection_and_activity_recognition_with_machine_learning-Informatica-09.pdf`.

[60] 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. doi: 10.1109/ICMI.2002.1166960. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1166960`.

[61] 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. doi: 10.1145/1409635.1409638. URL `http://portal.acm.org/citation.cfm?doid=1409635.1409638`.

[62] 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. doi: 10.1162/jmlr.2003.3.4-5.993. URL `http://www.crossref.org/jmlr_DOI.html`.

[63] 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. URL `http://www.computer.org/portal/web/csdl/doi/10.1109/ICCV.2007.4408894`.

[64] 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. URL `http://www.public.asu.edu/~jye02/CLASSES/Fall-2005/PAPERS/boosting-icml.pdf`.

[65] 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. doi: 10.1145/1178782.1178794. URL `http://portal.acm.org/citation.cfm?doid=1178782.1178794`.

[66] 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. doi: 10.1109/ICASSP.1999.757481. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=757481http://www.computer.org/portal/web/csdl/doi/10.1109/ICASSP.1999.757481`.

[67] 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. doi: 10.1145/1107548.1107591. URL `http://portal.acm.org/citation.cfm?doid=1107548.1107591`.

All remaining results are in this section.

(a)

(b)

(c)

(d)
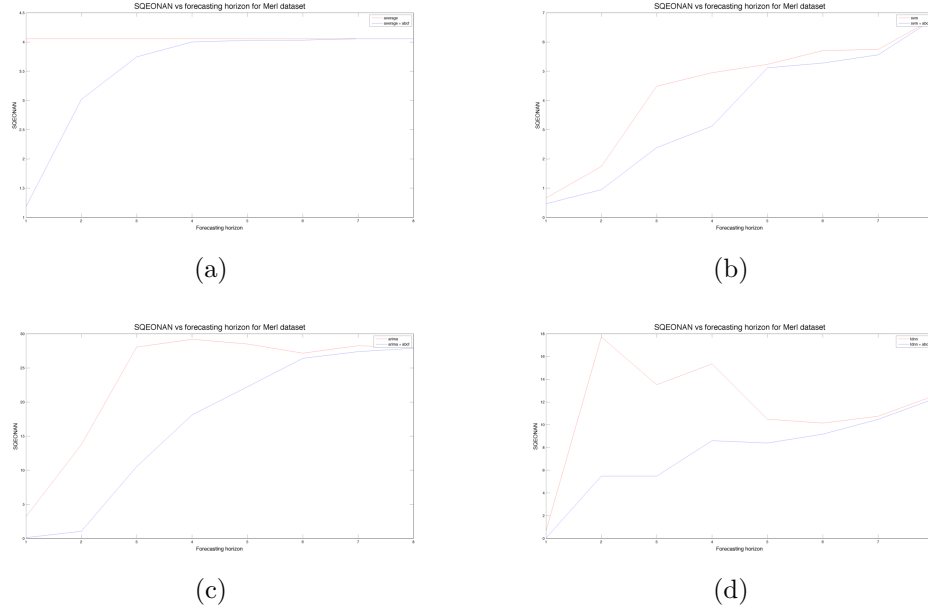
Figure A.1: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm
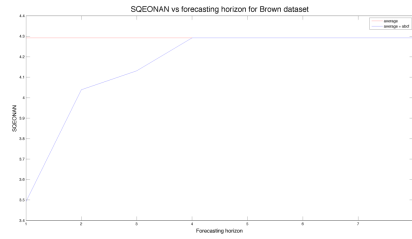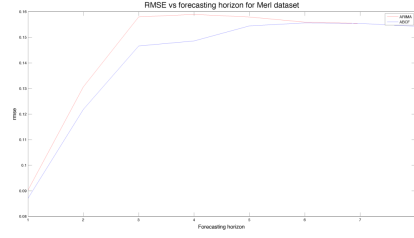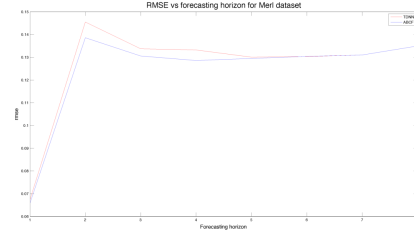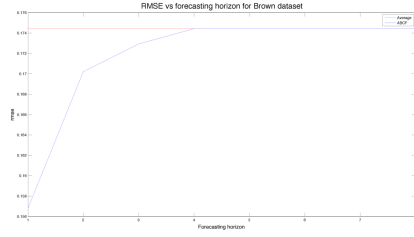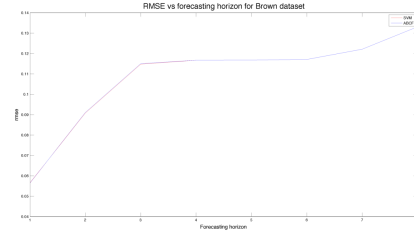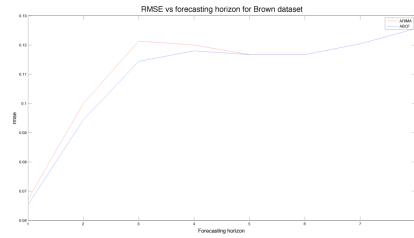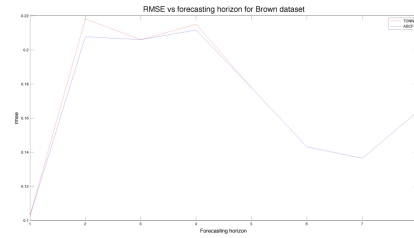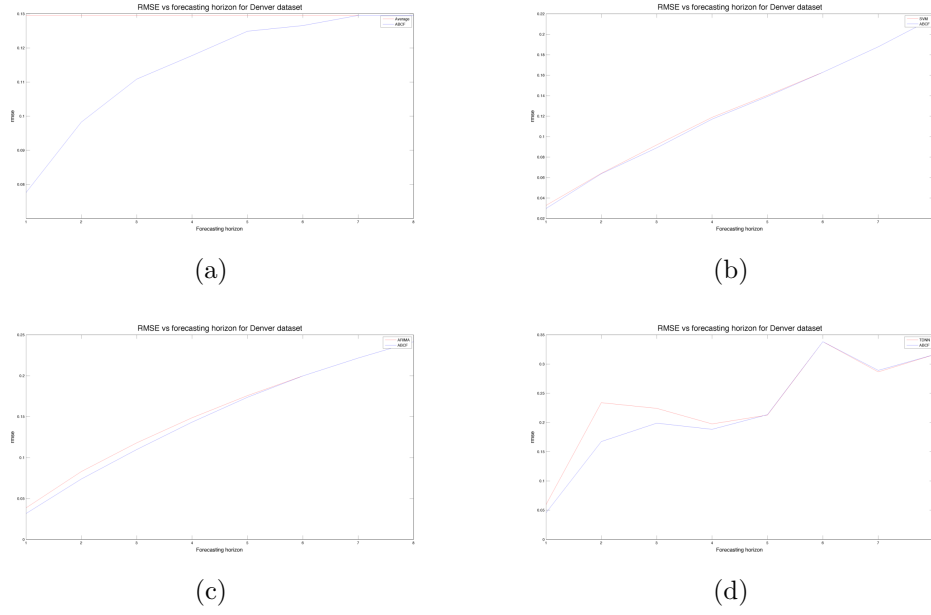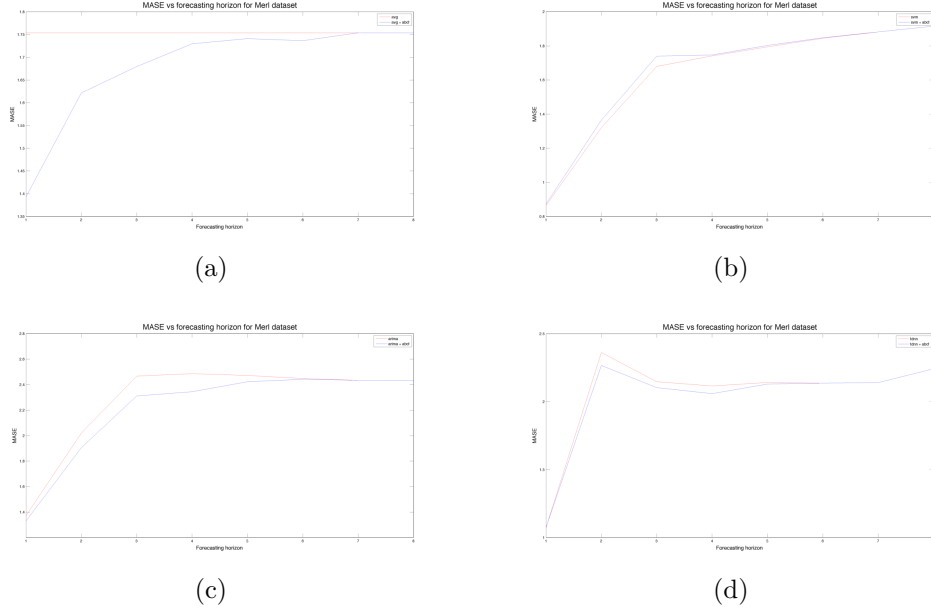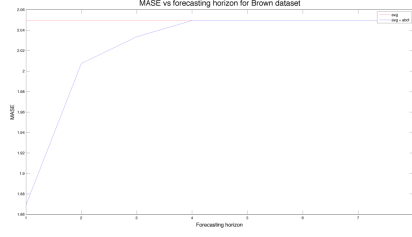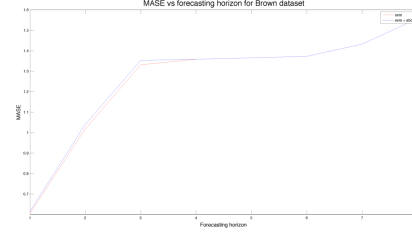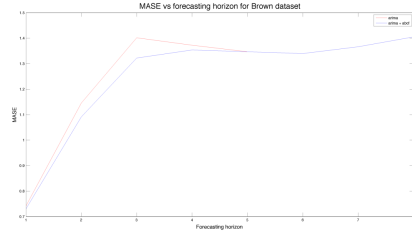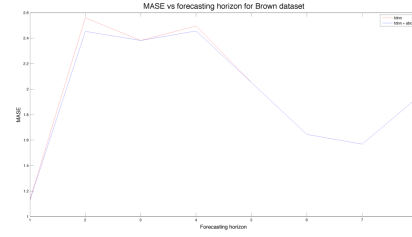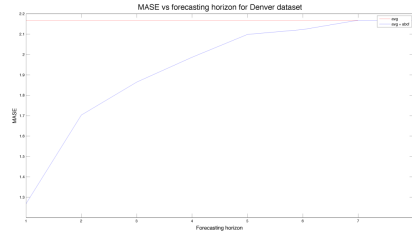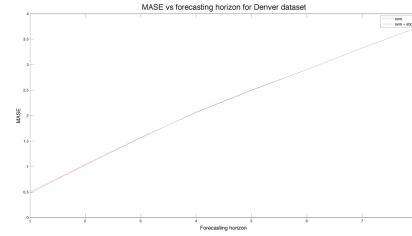
(a)
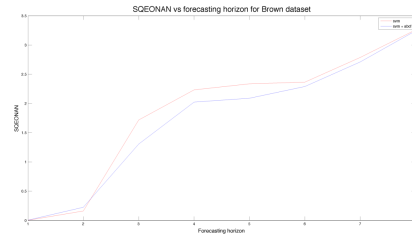
(b)
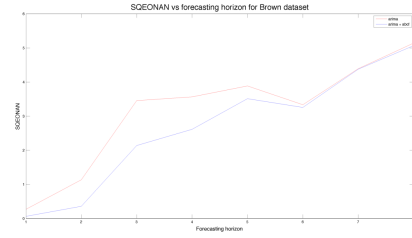
(c)
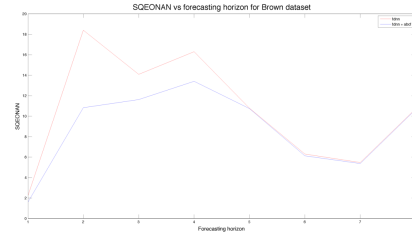
(d)

Figure A.2: Results for the four base forecasting algorithms for the Brown Hall dataset and the improvements to SQEONAN from using our ABCF algorithm
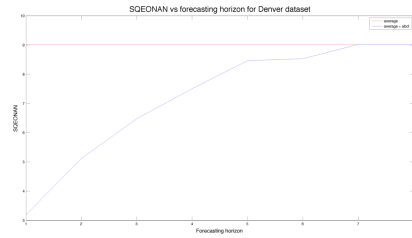
(a)
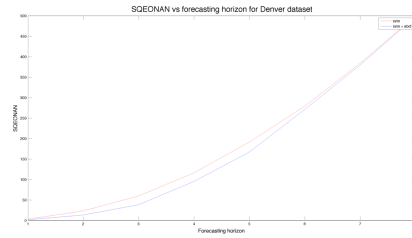
(b)
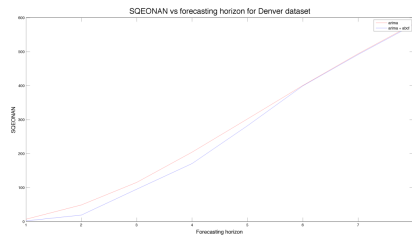
(c)
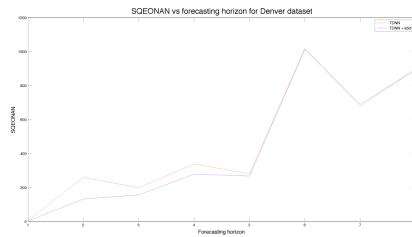
(d)

Figure A.3: Results for the four base forecasting algorithms for the Denver dataset and the improvements to SQEONAN from using our ABCF algorithm
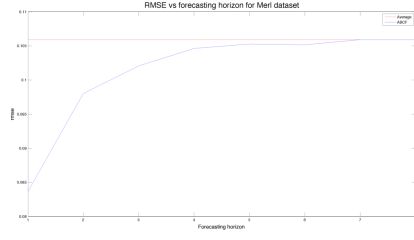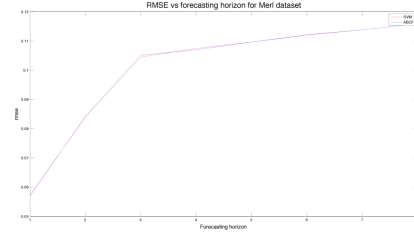
Figure A.4: Results for the four base forecasting algorithms for the MERL dataset and the improvements to RMSE from using our ABCF algorithm

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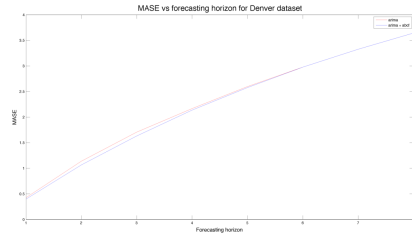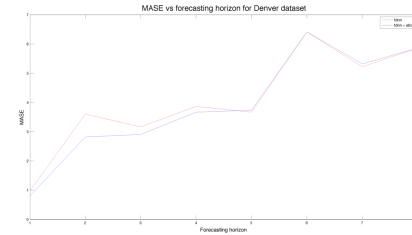
(a)

(b)

(c)

(d)

Figure A.6: 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.7: Results for the four base forecasting algorithms for the MERL dataset and the improvements to MASE from using our ABCF algorithm

(a)　　　　　　　　　　　　　　　　　　(b)

(c)　　　　　　　　　　　　　　　　　　(d)

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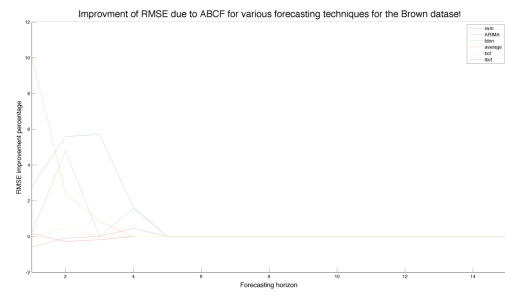

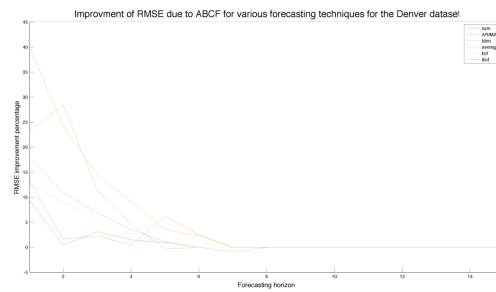(a)　　　　　　　　　　　　　　　　　　(b)

(c)　　　　　　　　　　　　　　　　　　(d)

Figure A.9: Results for the four base forecasting algorithms for the Denver dataset and the improvements to MASE from using our ABCF algorithm
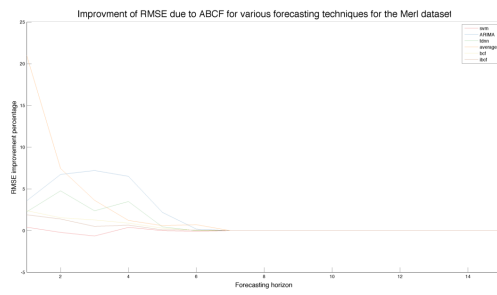
(a)

(b)

(c)

Figure A.10: 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]