COMBINED SEASONAL ARIMA AND ACTIVITY RECOGNITION ALGORITHMS FOR USE IN TRAFFIC FORECASTING

by

James Howard

TABLE OF CONTENTS

LIST OF FIGURES

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, any possible areas of improvement in this area are heavily sought after. One idea for saving energy is to only regulate the temperature in rooms that are actually in use. While the problem of determining what rooms are in use can be solved easily by a motion sensor, this problem becomes more difficult when the lead time to heat or cool a room is considered. If accurate forecast models could be made for the occupancy of any section of the building, then a control scheme may be created that could save on total energy cost.

As another example where the forecasting of occupancy may be used to produce significant improvements, 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 [2]. As of 2005 the total estimated fuel savings would amount to approximately 17 billions gallons of motor fuels annually. If accurate estimates of future traffic patterns at each traffic light were available then dynamically changing the light timings to account for such traffic would improve overall traffic flow.

In both of the above scenarios motion through the environment can be captured through a network of many sensors. For vehicular traffic systems, networks already exist using inductive loops and radar based sensors to count the number of cars in a given unit of time. In the case of buildings, such networks are not as common. To acquire such counts one could install a network using many infrared motion sensors and cameras to count human motion through the building.

1.1 Objective and Approach

The objective of this work is to forecast the number of moving agents in a region of space δ seconds in the future. This could be represented by a hypothesis function $h(x, \delta)$. We will use mean absolute scale error (MASE) [3] and mean absolute percentage error (MAPE) as cost functions to compare with other previously implemented techniques. Due to the level of noise present in traffic scenarios, the forecasted value of a sensor reading is aggregated for a time appropriate for the setting. For vehicular traffic, most work deals with reading every 15 minutes to one hour. For building traffic, this aggregation is 3 to 5 minutes.

To assist in constructing models we make the assumption that data is generated from activities produced by human controlled entities moving through the environment. Also we assume that the activities are repetitive and on some schedule. The result of these assumptions is that from such scheduled movement we get sensors which have a spatial correlation and that for example, from week to week on the same day display similar trends. Much of the research on traffic forecasting makes a similar assumption.

We also assume that sufficient deviations from our forecasting function are often not the result of noise, but are due to an activity that does not commonly occur on that day. 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 possible combinations of activities. In an environment with many activities that could occur at multiple different times such combinations may be prevalent. Past work doesn't address the problem of overlapping activities.

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 a seasonal autoregressive integrated moving average (ARIMA) model. 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 taking outputs from all trained activity models and the background seasonal ARIMA model.

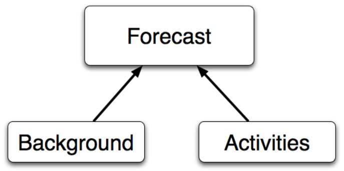
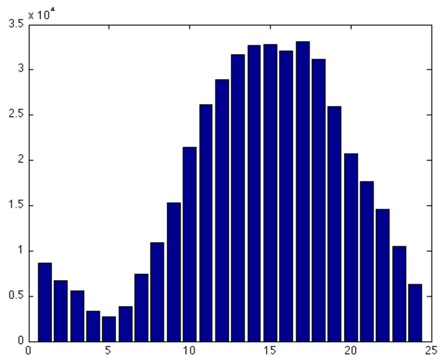
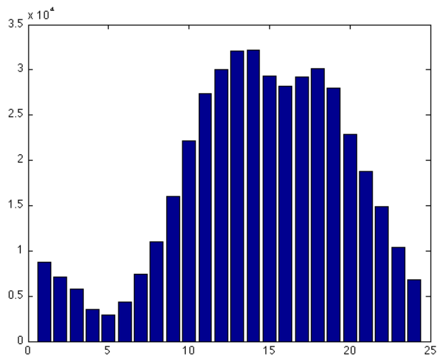


Figure 1.1: Forecasting is based on activities and a background model

1.2 Example Activity



(a) Away game Sundays



(b) Broncos game Sundays

Figure 1.2: Total number of cars passing major highway sensors on Sundays in September and October 2010

To illustrate an example of the need for our approach we provide the following example. Figure 1.2 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 1.2a with figure 1.2b 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.

Traditional parametric 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. To compound the problem further there may be multiple activities occurring at the same time such as a Rockies game and a Broncos game. It is probable that the number of occurrences of such overlapping activities are few and that no training instances exist for traditional models to handle.

Our approach will model each discrete activity separately and independent of the background model. In this case a model for Broncos games and a model for Rockies games would be trained. Once trained, accurate prediction should be possible despite the time of the games or the presence of other activities.

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 seasonal ARIMA models.
- A new representation of activities using a mixture of time series multivariate Gaussians.

- A new measure for determining activity model forecasting accuracy.

1.4 Structure of the Proposal

The remainder of this proposal is outlined as follows. Section two reviews current work related to traffic prediction and activity modeling. Section three gives a summary of each dataset used in this work. Section four details specific pieces of the overall approach. All of the approach is not solved and where possible this section details potential ways to proceed with each unsolved part of the approach. Finally section five is a time line of when the remaining work is expected to be completed.

‘

Chapter 2

LITERATURE REVIEW

This literature review is split into two sections: one on traffic forecasting and the other on activity modeling. This split was chosen due to its logical similarity to our approach. It is first necessary to understand other forecasting techniques and then activity models which we will use to improve on the best forecasting technique. The publications were chosen based on similarity to the work laid out in this paper, novelty of technique, and extent with which the methods described may be utilized in this work. Taken together, they show a lack of research specific to using group activity modeling to improve forecasting performance in roadway or building environments.

2.1 Traffic Prediction

2.1.1 Auto Regressive Moving Average Models

Due to the importance of auto regressive moving average (ARMA) based models, specifically the seasonal auto regressive integrated moving average (ARIMA) model in this work, we briefly introduce such models in this section. For a more detailed description of ARMA and seasonal ARIMA models refer to the textbooks by Box and Jenkins [4], Franses [5], or Cryer and Chan [6]. All notation for the models used below was taken from Box and Jenkins.

An ARMA model of with a number of ϕ and θ parameters, represented by p and q respectively is defined as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - ... - \theta_q e_{t-q} \tag{2.1}$$

where $\{Y_t\}$ denotes the observed time series and $\{e_t\}$ represents an unobserved white noise series. We abbreviate the model name to ARMA(p, q). This approach involves training a set of parameters to fit a regression model based on a fixed number of historic readings. ARMA based approaches are useful for their capability to predict further than one measured time instance in the future.

For traffic data, many ARMA variations have been implemented such as ARIMA [7], VARMA [8], STARMA [9], fractional ARIMA [10], and seasonal ARIMA [11]. Numerous papers [12, 13, 14, 15, 16] have been published comparing the results of ARMA based models vs other approaches. While not always the best, ARMA models consistently forecast results on par with other state of the art approaches. Another study [9] compared univariate ARIMA models versus the multivariate STARIMA and VARIMA models. On two datasets generated from the same road network over different times the univariate ARIMA was on par or slightly out performed multivariate models.

Seasonal ARIMA

Due to our traffic data having periodic trends and a non stationary mean, it is advantageous to use a seasonal ARIMA model. An ARIMA model can handle non stationary means while periodic trends are handled by the seasonal component of the model. The seasonal ARIMA model is defined as:

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla_s^D z_t = \theta_q(B)\Theta_Q(B^s)a_t \tag{2.2}$$

where B is the backshift operator, $a_t \sim N(0, \sigma^2)$ is a white noise process, ∇_j^i is the seasonal difference operator, and $\phi,\ \Phi,\ \theta,\ \Theta$ are trainable parameters. We represent seasonal ARIMA models as $ARIMA(p, d, q)(P, D, Q)_s$ 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.

Additional motivation for using seasonal ARIMA models

A source of error in seasonal ARIMA models is the presence of large infrequent transitory shocks [17]. In the context of seasonal ARIMA models a large shock value implies a large deviation from previous readings for the same time frame. In the traffic domain this means that large shock values indicate the presence of large deviations in typical daily motion. Recall that one of our assumptions is that there exists a set of discrete activities which account for much of the motion within the environment. If these large deviations are one of these activities, then we would expect to see the seasonal ARIMA model forecast incorrectly in a consistent way.

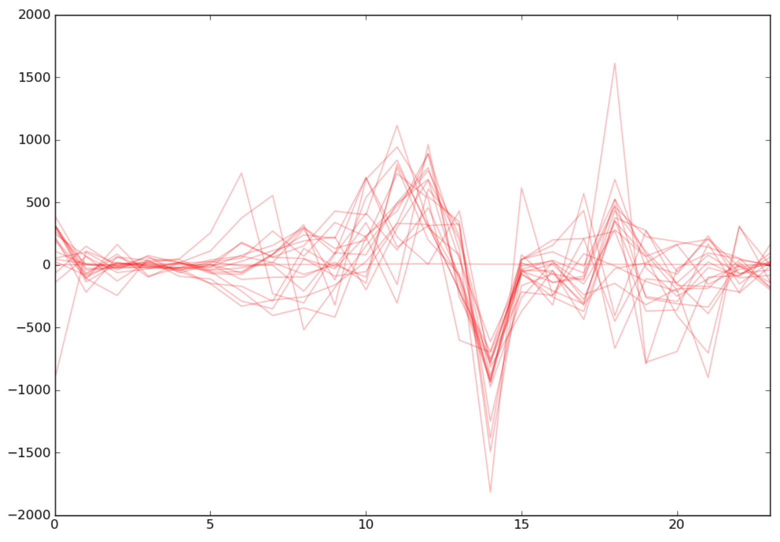


Figure 2.1: Residual counts of all Broncos afternoon home games from 2008 - 2010

Empirically, we find that activities do produce semi-consistent errors in the forecasting of the seasonal ARIMA model. Referencing the Broncos example again, figure 2.1 shows the residual (actual observations - one step ahead seasonal ARIMA forecasts) counts of all Sunday afternoon Broncos from 2008-2010. While there is some variance in the image, it is clear that the seasonal ARIMA model consistently incorrectly forecasts during the presence of a Broncos game. This work looks to exploit such consistent incorrect forecasts by using residual data to train a set of activity models.

Despite the strong performance of ARMA based approaches to handle traffic forecasting, numerous other approaches are discussed below for a complete review of the literature.

2.1.2 Other Approaches

Neural networks offer another avenue for solving traffic prediction problems. Such networks can be trained directly using a time series neural network, which is simply a network with additional input nodes to account for the past m readings. Research [15, 16] has shown performance to be generally comparable to Seasonal ARIMA models. In one study neural networks performed better and in another seasonal ARIMA approaches performed better. The actual performance is likely data dependent.

Due to their ease of implementation because of the abundance of available software packages and relatively low computation cost, neural networks seem to perform well in conjunction with other models. [18] showed the feasibility of using multiple neural network models with final forecasts computed by a Bayesian combination of all other neural network models. In another study [19] implemented a neural network based on the output of a seasonal ARIMA model. In both studies, significant improvement to the MAPE for both training and test data were made with a combined approach as opposed to any singular model. While not perfectly analogous, these approaches which utilize multiple models are similar to our activity based approach.

There has also been work into mapping traffic onto a graph structure which is a representation of the spatial layout of the environment[20]. Such work uses a statistical flow model to forecast the future density on any edge within the graph. A graph approach has the benefit of improved forecasting on short time scales because it naturally limits the potential density of traffic of any edge based on the current state of the graph and how likely it is that one edge may affect another edge within a given amount of time. To perform such forecasts, vehicles are assumed to drive along a shortest path to a set of possible destinations weighted by probability of ending at that destination. When the forecast of future traffic density goes beyond the scale of a few time steps the number of possible destinations exponentially increases and as expected the future forecast error increases significantly. This approach seems most promising for relatively short forecast time scales. Also, this style of forecasting is not feasible if the data collection rate is relatively slow as at a certain collection rate there is no longer a strong correlation between the sensors within the network.

In an attempt to resolve the deficiencies in parametric models, such as ARMA, work has been done on non-parametric models. In [21, 14] a nearest neighbor approach was used where the state space was the last n lag values, and forecasting was performed by aggregating the k nearest neighbors values weighted by how recently the neighbor data was collected. The idea is that when the data becomes too "chaotic" parametric models are unable to forecast. Thus models which are not trained on pre-set parameters will be better able to account for such "chaotic" data. In practice however, ARMA based models tend to out perform nearest neighbor approaches.

2.2 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 [22, 23]. 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 [23], support vector machines [24, 23, 25], naive Bayes [23, 25], nearest neighbor [23, 25], and hidden Markov Models (HMM) [22, 26]. Comparisons amongst models have shown that performance is data dependent and that no one model appears to be best for all types of activities [23, 25]

Huynh [27] 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 [28]. 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 [29] 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 [30]. The performance level of suffix trees seems to be highly noise dependent. [31] 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.

2.3 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 [32, 33]. 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 [34, 33] 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 [35].

In a method similar to the HMM tree, work has been done to create a hierarchy of fixed filters based on possible sensor topologies [36]. 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 [37] 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.

Chapter 3

APPROACH

As briefly discussed earlier, our approach is to train a seasonal ARIMA model for a base level of forecasting. The residual forecasting errors of this approach are then run through an activity recognition algorithm to produce a set of models corresponding to instances where the ARIMA model mis-forecasts. Final forecasting is then determined by combining the forecasts of all activity models and the background seasonal ARIMA model using a Bayesian combined predictor. The entire approach is unsupervised.

While research has shown that univariate ARIMA models outperform multivariate ARIMA models for forecasting, we believe it possible that activity models will perform better when the data of spatially local sensors (neighbors) are included. For each sensor, knowledge of its neighbors allows for reduced computation time when computing what sensors are optimal for each activity. Thus, for our approach to be unsupervised, it is necessary to calculate sensor neighbors.

This section outlines all aspects of our approach. First we describe a method to calculate the temporal and spatial correlations of the data observed by the sensors giving sensor neighbors. Following this is a discussion on fitting a seasonal ARIMA model. Then we describe a couple of potential approaches to model activities. Finally we discuss prediction and performance calculations.

3.1 Spatial and Temporal Correlation of Sensors

Determining how sensors are correlated allows for unsupervised approaches to searching for the optimal set of sensors which make up an activity. This correlation is determined by calculating Pearson's correlation for each pair of sensors at a given time offset. For reference Pearson's correlation is:

$$C_{pearson}(a,b) = \frac{E(ab) - E(a)E(b)}{\sqrt{E(a^2) - E^2(a)}\sqrt{E(b^2) - E^2(b)}} \tag{3.1}$$

where $E(a)$ is the expected value of the random variable a.

Converting this formula to binary data and applying a time offset δ gives

$$C_{binary}(a,b;\delta) = \frac{E(x_a x_b) - E(x_a)E(x_b)}{\sqrt{E(x_a) - E^2(x_a)}\sqrt{E(x_b) - E^2(x_b)}} \tag{3.2}$$

where the values for x_b are always at a time offset of δ in the future of x_a. Thus if given $x_a^{(i)}$ then calculations are performed with $x_b^{(i+\delta)}$.

Next the maximum value of C_{binary} for all δ is found

$$\arg\max_{\delta} \; C_{binary}(a,b;\delta) \quad \forall\delta \in \{1,2,...,M\}). \tag{3.3}$$

In practice, as a way to reduce computation time, the calculation can be performed with a much smaller maximum δ than M. The returned maximum value of δ for each pair of sensors is a good indicator of travel time between the two sensors. This information is useful for determining what sensors to use when calculating activities.

Figure 3.1 shows an example of this approach. This figure displays the maximum pairwise sensor correlations across a δ of up to 10 seconds. The values were then

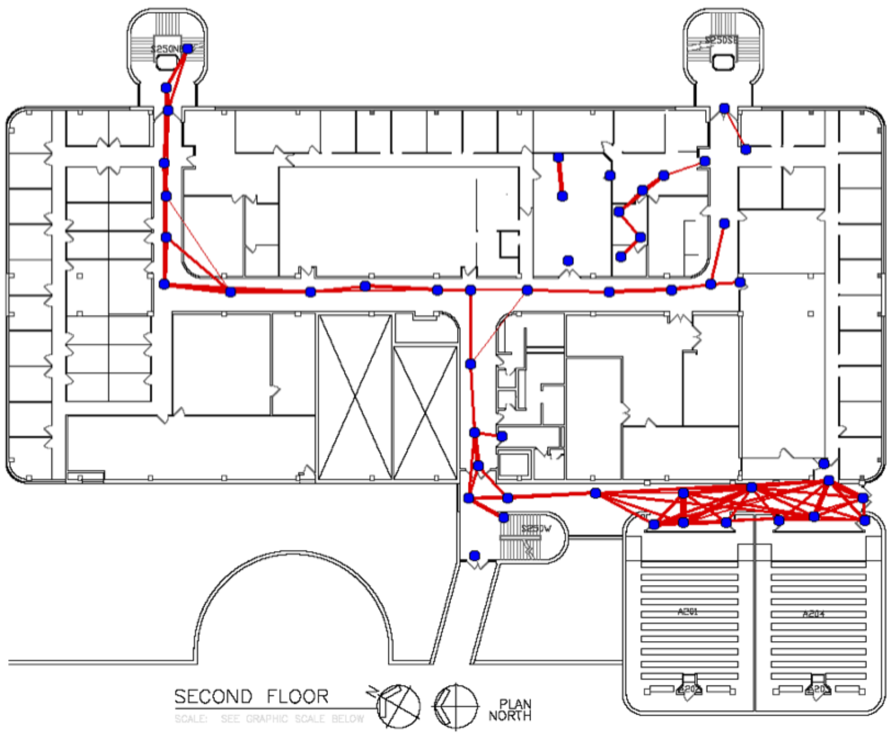
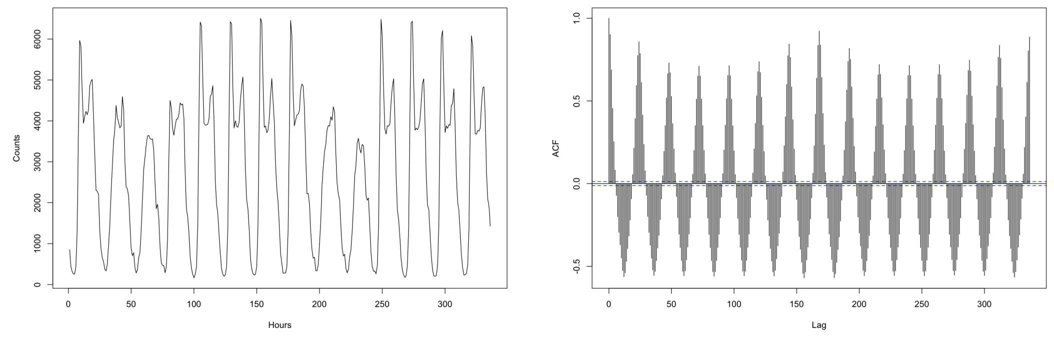


Figure 3.1: Pairwise sensor correlation

thresholded to prevent displaying every line between two sensors. The line thickness represents the correlation value. While the pairings are not perfect in all places, this figure does show the potential of this approach.

3.2 Seasonal ARIMA Model

The seasonal ARIMA model has two important roles in this work. One role is its direct predictive capabilities. The other role is for the determination of activities. The residual data created from subtracting the current observations from seasonal ARIMA forecasts is used as the basis for activities. This section briefly outlines the steps necessary to fit a seasonal ARIMA model for our problem.



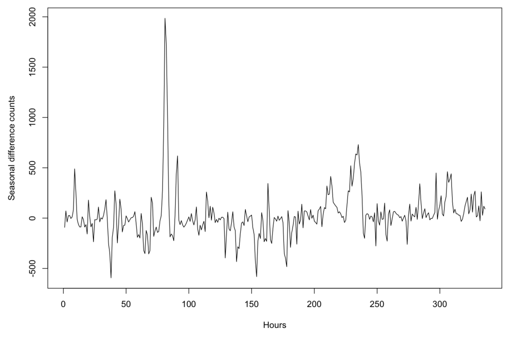
(a) Vehicle counts (b) Autocorrelation of vehicle counts

Figure 3.2: Counts and autocorrelation over a two week period.

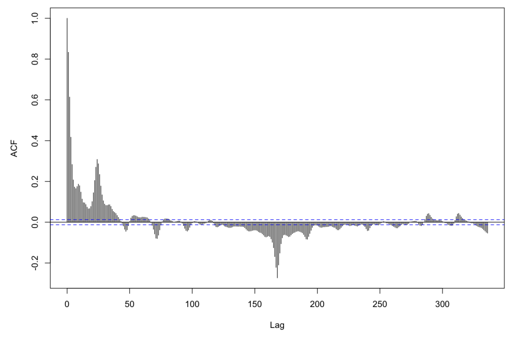
3.2.1 Fitting a Seasonal ARIMA

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 3.2 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



(a) Seasonal difference counts



(b) Autocorrelation of seasonal difference counts

Figure 3.3: One week seasonal difference counts and autocorrelation over a two week period.

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 3.3b shows such stationarity. Applying the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) [38] 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.

Most of the input parameter values for seasonal ARIMA models tend to be 0, 1, 2, or 3 [4]. Due to this small range of input values the total input space is relatively small (Six parameters with four possible values equates to $4^6 = 4096$) allowing us to apply a brute-force search for the best model. Model performance is determined by the Akaike information criterion (AIC) [39]. Our optimal model is a seasonal ARIMA $(1, 0, 1)(0, 1, 1)_{168}$.

Figure 3.4 shows an example of one-step ahead prediction performed on a sample

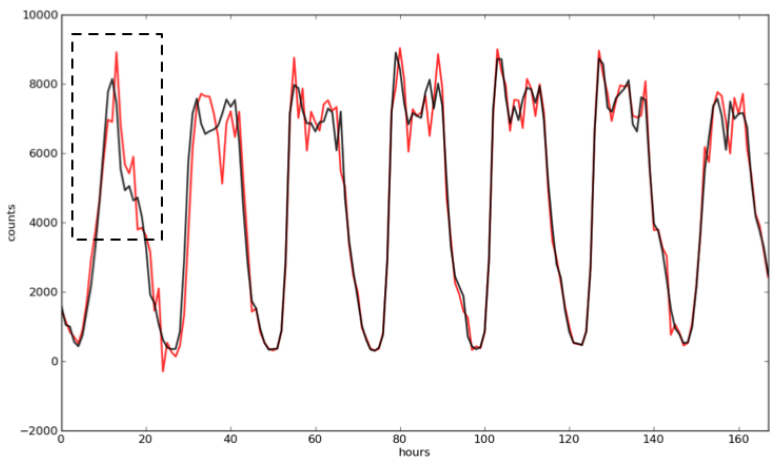


Figure 3.4: One-step ahead prediction for a sample week. Black line is original data. Red line is forecasted data. Dotted box shows an example of mis-forecasting due to a broncos game.

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 2.1. 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 [7, 40].

3.3 Activity Recognition

This section first details some of the challenges to obtaining training data for activities in an unsupervised manner. Next we describe two algorithms used to cluster activities from the training dataset. Finally we introduce a new measure of the length of time required to classify activities.

3.3.1 Training Data

Segmenting a time series of data which will be used to train activities is an important problem. A time series that is too long may encompass multiple activities, while one that is too short will not adequately describe an entire activity. Factors that need to be considered are what length of time each time series will cover, the features included in the time series and if the time series will overlap.

The problem of unsupervised time series data segmentation for activity recognition is not commonly discussed in the literature. In much of the activity recognition literature the problem is not applicable as data either splits naturally (such as RFID item usage data) or is supervised (such as human wearable accelerometers).

One approach we have implemented involves sliding a fixed window over the residual data to parse out local time series of maximum deviation. The problem with this approach is that it does not consider windows of different lengths. As an initial solution to this problem, we will repeat this process with multiple window lengths and include all time series as valid training data. This approach is not ideal and we will continue to search for a better solution to isolate residual data associated with activities.

3.3.2 Models

This section details two activity recognition and representation approaches. The hidden Markov model approach has shown promise when applied to building data. The time series mixture of Gaussians has not been implemented and we introduce it here. Other conventional approaches such parametric regression and time series neural networks may need to be implemented for comparison, but are not discussed here due to brevity.

Hidden Markov Model

A useful approach to time series modeling is the hidden Markov model (HMM) [41, 42]. This model is derived from first order Markov chains having the property that future states are dependent only on the current state. Mathematically this property can be show as:

$$P(x_t|x_{t-1}, x_{t-2}, ...) = P(x_t|x^{t-1}) \tag{3.4}$$

where x_t is the value of time series x at time t.

To cluster the time series and construct HMM activities models we implement a k-means clustering algorithm of HMMs. This work is similar to that of [33] with the notable exception that we initialize parameters according to the K-Means++ algorithm [43] instead of randomly and that we remove outliers. Convergence is based on the silhouette [44] score of the current clustering.

The algorithm we use to create HMMs is described below:

1. Assign each observation sequence to a cluster according to the K-Means++ algorithm
2. Train a HMM for each cluster
3. Compute the silhouette score
4. Repeat
   (a) For each observation sequence compute the similarity to each HMM cluster
   (b) Assign each observation sequence to the most similar cluster
   (c) For each cluster remove outliers
   (d) Train a HMM for each cluster

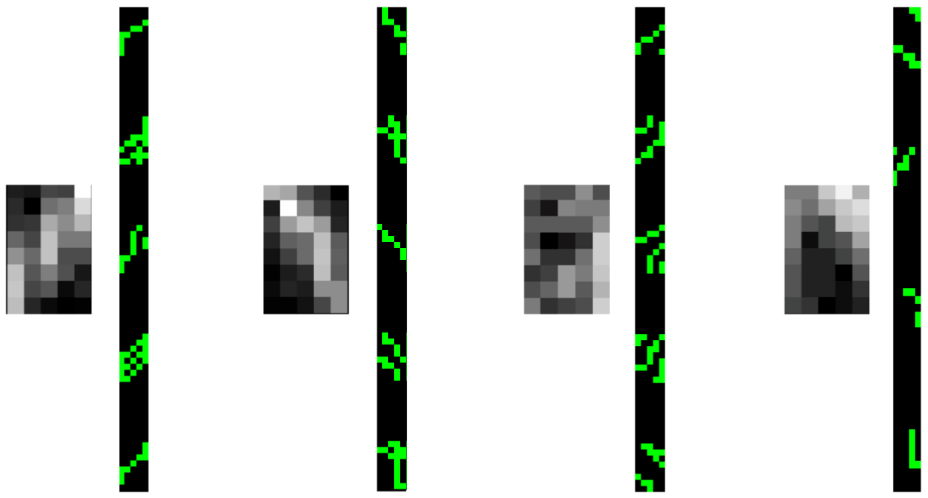


Figure 3.5: An example of training four HMMs to describe 200 observations of data.

   (e) Compute the silhouette score

5. Until the silhouette score has not yet converged
6. Attempt to reintroduce outliers
7. Train a HMM for each cluster

Similarity can be computed by measuring the log-likelihood of the observation sequence to the HMM. Fitting these log-likelihood scores for each HMM cluster to a normal distribution, outliers can be removed if they are sufficiently far from the HMM cluster. Two standard deviations seems to be a good empirical value for this threshold. Similarly, because outliers are removed before the HMM clusters have converged to their final states, once removed outliers might now fit to one of the HMM clusters. Outlier observation sequences are sampled against all HMM clusters and included if they are within the outlier threshold.

Figure 3.5 shows an example of applying K-means clustering to Hidden Markov models. The four models were trained from a five sensor neighborhood taken from the Brown Building Dataset. The black bars show example observations while the large rectangles show a representation of the HMM trained for that data. This run was performed on 200 observations and the above algorithm found these four HMMs to describe the data.

Performing this type of clustering to train HMMs has shown considerable promise. While we have not yet trained HMMs on residual data, we believe this approach will translate well and accurately capture activities due to the similarity of residual data to the raw data on which we have previously worked.

Time Series Mixture of Gaussians

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

$$\ell(\theta) = \sum_{i=1}^{\mathbf{M}} log\{p(x^{(i)}|\theta)\} \tag{3.5}$$

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

$$w_k^{(i)} = p(z = k|x^{(i)}) \tag{3.6}$$

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:

$$\ell(\theta|x) = \sum_{i=1}^{\mathbf{M}} \sum_{k=1}^{\mathbf{K}} w_k^{(i)} \log\{\frac{p(x^{(i)}|z = k)p(z = k)}{w_k^{(i)}}\} \tag{3.7}$$

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

$$p(x^{(i)}|z = k) = \prod_{n=1}^{\mathbf{N}} \mathcal{N}_n(x^{(i)}) \tag{3.8}$$

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 3.7 and 3.8 gives the following log likelihood

$$\ell(\theta|x) = \sum_{i=1}^{\mathbf{M}} \sum_{k=1}^{\mathbf{K}} w_k^{(i)}\{\log \frac{p(z = k)}{w_k} + \sum_{n=1}^{\mathbf{N}} \log \mathcal{N}_n(x^{(i)})\} \tag{3.9}$$

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

$$w_k^{(i)} = p(z = k|x^{(i)}) \tag{3.10}$$

M-Step For the maximization step, it is assumed that we know the values of $w_k^{(i)}$. Thus, we need to maximize equation 3.9 with respect to μ, Σ, and θ. The results of these maximizations are given below:

$$\theta_k = \frac{1}{\mathbf{M}} \sum_{i=1}^{\mathbf{M}} w_k^{(i)} \tag{3.11}$$

$$\mu_{k,n} = \frac{\sum_{i=1}^{\mathbf{M}} w_k^{(i)} x_n^{(i)}}{\sum_{i=1}^{\mathbf{M}} w_k^{(i)}} \tag{3.12}$$

$$\Sigma_{k,n} = \frac{\sum_{i=1}^{\mathbf{M}} w_k^{(i)} (x^{(i)} - \mu_{k,n})(x^{(i)} - \mu_{k,n})^{\mathrm{T}}}{\sum_{i=1}^{\mathbf{M}} w_k^{(i)}} \tag{3.13}$$

3.3.3 Performance Metrics

The final measure for forecasting performance is MAPE or MASE. This measure encompasses the all steps of our algorithm. It would be helpful to have a measure of performance for activities alone. Because most activity recognition work has a classified dataset, the techniques commonly used, such as precision and recall, will not apply for our problem. Ostensibly, a set of activity models which are both accurate and are identified with small amounts of data leads to greater forecasting accuracy as the earlier an activity model can be identified, the earlier its forecasts may be applied to final forecasting. Thus, as a way to compare activity recognition techniques, it is important to know both the accuracy of the activity models and the time it takes to correctly identify the correct model for a given time series.

To calculate activity model accuracy the traditional mean squared error or root mean square cost functions should work well. For calculating the time to identify the correct model we propose a measure defined here. To calculate this we first define activity identification as:

$$C(x) = \arg\max_k A_k(x) \ \forall k \tag{3.14}$$

where x is a time series of length M and $A_k(x)$ is the probability that time series x

corresponds to activity k.

Next, we introduce another term which we call an identification set. This set is defined as:

$$c_M = \{C(x_{1:2}), C(x_{1:3}), ..., C(x_{1:M})\} \tag{3.15}$$

where $x_{i:j}$ represents all data from time offset i to time offset j. Notice that the length of c will be equal to the length of the time series M.

Using the identification set, it is now possible to calculate the identification time.

$$J(x) = \max_i \{i | c_M = c_{M-1} =, ..., c_{M-i}\} \tag{3.16}$$

This measure calculates for a given time series the earliest time at which no misidentification occur. The measure we are interested in is the average identification time for a set of activity models.

Using both activity model accuracy and identification time allows for a comparison of activity recognition techniques and determination of the number of activity models. If too many activity models are trained then it is expected that training set activity accuracy will increase, but identification time will increase. If too few activity models are trained, training set activity accuracy will decrease, but identification time will decrease. Finding a balance between accuracy and classification time will be performed empirically.

3.4 Forecasting Model

To improve on the forecasts of the seasonal ARIMA model we attempt to forecast the error in the ARIMA model. To forecast this error we want to use the forecasts of all trained activity models. The approach that we propose to implement this forecasting is to use a Bayesian Combined Predictor (BCP) [45]. A BCP is a method for combining the forecasts of multiple models given the probability for each model based on current observations and the probability of each model given a short history of observations. The BCP is defined as:

$$p_k^{(t)} = p(k|x_{1:t}) = \frac{p_k^{t-1} \cdot err(x_t - h_k(x^{1:t-1})}{\sum_{j=1}^{K} p_j^{t-1} \cdot err(x^{(t)} - h_j(x_{1:t-1})} \tag{3.17}$$

where $p(Z = k|x)$ is the probability of model k given data x, K is the total number of models, h_k is the forecast from model k, and $err()$ is the probability mass function of the forecasting error for model k.

Forecasting is then performed by finding the model which is most model likely describing the current activity and taking the forecast from that model.

$$h_k = h_{k*} \quad where \quad k* = \arg \max_{1 \leq k \leq K} p_k^{(t)} \tag{3.18}$$

If the forecasting error of each model is assumed to be Gaussian, calculating the BCP is relatively easy. Also, because BCP will always select from a model to forecast, we introduce a null model $\sim N(0, \sigma^2)$ which we represent as a Gaussian with zero mean and standard deviation σ which is calculated from the seasonal ARIMA model.

BCP resolves some of the problems present with multiple model forecasting. Because research has shown [37] empirically that there exists no one model, feature space, or time scale that best describes all the possible activities, it is likely optimal to train models which incorporate different features and time scales. BCP allows for forecasting with a heterogeneous mixture of models by allowing each model its own probability mass error function and history of readings.

Chapter 4

TESTBEDS

Included here is a brief description of the data sources that will be used for this work.

4.1 University Buildings

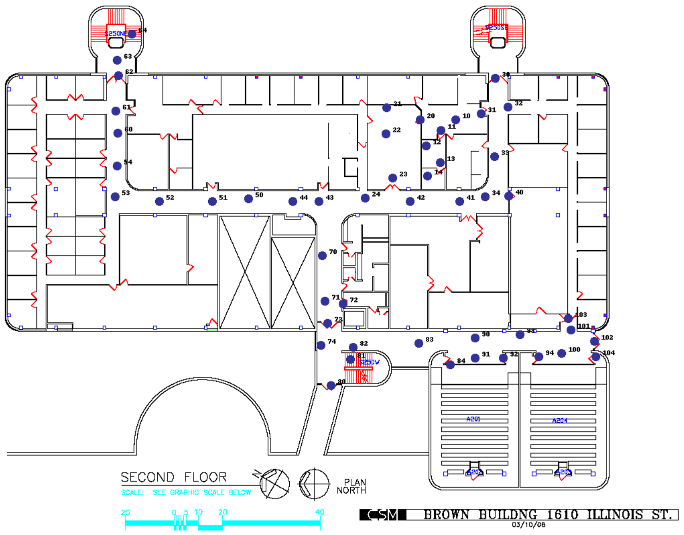


Figure 4.1: Mote locations on the second floor of the Colorado School of Mines Brown Building.

Infrared motion counts from a class room building at the Colorado School of Mines. Simple infrared motion sensors were placed throughout the floor and were polled for motion data every second. This data has a predictable time of day model based on class times with Monday, Wednesday and Friday following a different schedule than Tuesday and Thursday.

Additionally we plan to have access to the Colorado School of Mines Center for Technology and Learning Media building's camera data. Camera data will be converted into motion data describing the number of people moving in any direction at a given time. This building serves as a class room building, computing center, and student congregation area due to the presence of an Einstein's bagels.

4.2 Denver Traffic

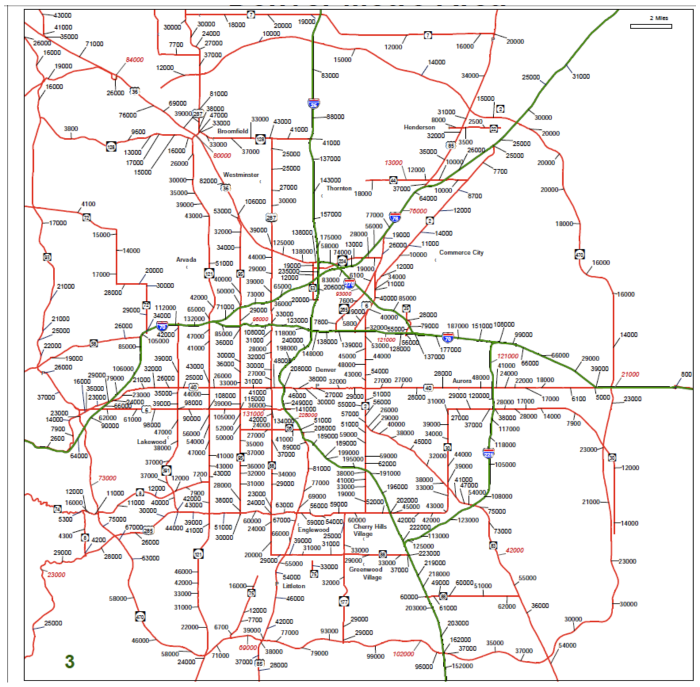


Figure 4.2: Traffic count sensor locations near the city of Denver.

Traffic count sensors from various locations near the city of Denver. 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.

4.3 Simulator

As another way to produce building motion behavior a simulator has been created. It is used primarily to create specific test runs which are used to give a better indicator of the performance of our methods.

The simulator creates a model of background information by mimicking typical single person activities such as occasional travel to the bathroom, vending machine, water cooler, etc. This background information is present throughout the simulation however it does not have to be constant. The possible locations and function that determines frequency of travel can be changed at any time.

On top of this background model, the simulator mimics various types of activities. Sample activities include meetings, lunch time exodus, or fire drills. For each of these activities, the number of people present along with functions that determine probability of arrival time (which includes the possibility of being late) and leave times are possible. These parameters allow for multiple types of meetings to be simulated.

One additional but important feature of the simulation is that it allows for sensors to have any type of sensing function. The simulator can mimic almost any type of sensor instead of simply radial infrared sensors.

Chapter 5

IMPLEMENTATION PLAN

5.1 Expected Results

Because seasonal ARIMA models appear to be on par with the current state of the art approach for performing traffic prediction, we at the least expect to outperform such models. We expect to empirically demonstrate on test data a performance improvement over seasonal ARIMA models. We also hope to demonstrate the efficacy of our activity recognition method and the accuracy of our performance metric.

5.2 Completion Timeline

Dissertation Proposal	January 2012
Finish Activity Recognition Models	January thru March 2011
Finish Prediction Model	April 2011
Dissertation Defense	August 2012

Table 5.1: Research Timeline

Thus, I plan to defend my thesis at the close of the summer semester 2012.

REFERENCES

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

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

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

[4] George Box, Gwilym Jenkings, and Gregory Reinsel. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Inc., 4th edition, 2008.

[5] Philip Hans Franses. *Time Series models for Business and Economic Forecasting*. Cambridge University Press, 1st edition, 1998.

[6] Jonathan D. Cryer and Kung-Sik Chan. *Time Series Analysis With Applications in R*. Springer, 2nd editio edition, 2008.

[7] 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.

[8] Wanli Min, Laura Wynter, and Yasuo Amemiya. Road Traffic Prediction with Spatio-Temporal Correlations. In *Sixth Triennial Symposium on Transportation Analysis*, pages 1–11, 2007.

[9] 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.

[10] O.W.W. Yang, Fei Xue, Jiakun Liu, Yantai Shu, and Lianfang Zhang. Traffic modeling based on FARIMA models. In *Engineering Solutions for the Next Millennium. 1999 IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.99TH8411)*, volume 1, pages 162–167. Ieee, 1999.

[11] Lionel Fillatre, D. Marakov, and S. Vaton. Forecasting seasonal traffic flows, 2003.

[12] M. Van Der Voort, Mark Dougherty, and Susan Watson. Combining Kohonen maps with ARIMA time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, 4(5):307–318, 1996.

[13] Vincent Cho. A comparison of three different approaches to tourist arrival forecasting. *Tourism Management*, 23(3):323–330, 2003.

[14] 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.

[15] J Taylor, L Demenezes, and P Mcsharry. A comparison of univariate methods for forecasting electricity demand up to a day ahead. *International Journal of Forecasting*, 22(1):1–16, January 2006.

[16] Sherif Ishak and Ciprian Alecsandru. Optimizing Traffic Prediction Performance of Neural Networks under Various Topological, Input, and Traffic Condition Settings. *Transportation Engineering*, 130(4):452–464, 2004.

[17] N.S. Balke and T.B. Fomby. Large shocks, small shocks, and economic fluctuations: outliers in macroeconomic time series. *Journal of Applied Econometrics*, 9(2):181–200, 1994.

[18] 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.

[19] F Tseng. Combining neural network model with seasonal time series ARIMA model. *Technological Forecasting and Social Change*, 69(1):71–87, January 2002.

[20] H.P. Kriegel, M. Renz, M. Schubert, and A Zufle. Statistical Density Prediction in Traffic Networks. In *8th SIAM Conference on Data Mining*. Citeseer, 2008.

[21] 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.

[22] 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.

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

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

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

[41] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[42] B. H. Juang and L. R. Rabiner. Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3):251, August 1991.

[43] David Arthur and Sergei Vassilvitskii. k-means ++ : The Advantages of Careful Seeding. In *The eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[44] L Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, 1990.

[45] 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.