

South Dakota State University
**Open PRAIRIE: Open Public Research Access Institutional
Repository and Information Exchange**


Theses and Dissertations

2016

Data Center Load Forecast Using Dependent Mixture Model

Md Riaz Ahmed Khan
South Dakota State University

Follow this and additional works at: <http://openprairie.sdstate.edu/etd>

 Part of the [Data Storage Systems Commons](#), and the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Khan, Md Riaz Ahmed, "Data Center Load Forecast Using Dependent Mixture Model" (2016). *Theses and Dissertations*. 1120.
<http://openprairie.sdstate.edu/etd/1120>

This Thesis - Open Access is brought to you for free and open access by Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. For more information, please contact michael.biondo@sdstate.edu.

DATA CENTER LOAD FORECAST USING DEPENDENT MIXTURE MODEL

BY

MD RIAZ AHMED KHAN

A thesis submitted in partial fulfillment of the requirements for the

Master of Science

Major in Electrical Engineering

South Dakota State University

2016

DATA CENTER LOAD FORECAST BY DEPENDENT MIXTURE MODEL

This thesis is approved as a creditable and independent investigation by a candidate for the Master of Science in Electrical Engineering degree and is acceptable for meeting the thesis requirements for this degree. Acceptance of this thesis does not imply that the conclusions reached by the candidates are necessarily the conclusions of the major department.

Reinaldo Tonkoski, Ph.D.
Thesis Advisor

Date

Semhar Michael, Ph.D.
Thesis Co-advisor

Date

Steven Hietpas, Ph.D.
Head, Electrical Engineering and Computer Science

Date

Dean, Graduate School

Date

ACKNOWLEDGEMENTS

I would like to acknowledge South Dakota Board of Regents (SDBoR) Microsoft Inc. for the financial support. I would also like to thank my advisor, co-advisor and committee members for their constant help and guidance.

CONTENTS

ABBREVIATIONS	vii
LIST OF FIGURES	ix
LIST OF TABLES	x
ABSTRACT	xi
CHAPTER 1 : INTRODUCTION	1
1.1 Background	1
1.2 Datacenter overview	3
1.3 Literature review	3
1.4 Contributions	6
1.5 Thesis outline	6
CHAPTER 2 : THEORY	7
2.1 Dependent mixture model	9
2.1.1 Markov process	10
2.1.2 Hidden Markov model	13
2.1.3 Hidden Markov model elements	14
2.1.4 Evaluating hidden Markov model parameters	17
2.2 Data fitting and n-step ahead forecast	26
2.2.1 Data fitting	26
2.2.2 n-step ahead forecast	27

2.3	BIC: criterion of number of states	27
2.4	MAPE: metric of prediction accuracy	28
CHAPTER 3 : PROCEDURES		29
3.1	Case Study I	29
3.1.1	Model Selection	30
3.2	Case Study II	33
3.2.1	Model selection	35
3.3	Case Study III	38
3.3.1	Model Selection	39
CHAPTER 4 : LOAD FORECAST AND PERFORMANCE EVALUATION . . .		42
4.1	Load Forecast	42
4.1.1	Forecast: Case I	42
4.1.2	Forecast: Case II	44
4.1.3	Forecast: Case III	45
4.2	Forecast by other methods	47
4.2.1	Simple Moving Average (SMA)	48
4.2.2	Simple Exponential Moving Average (SEMA)	49
4.2.3	Auto Regression Integrated Moving Average (ARIMA)	50
4.2.4	Auto Regression Neural Network (ARNN)	51
4.3	Results Comparison	52
4.4	Economic Impact	56

CHAPTER 5 : CONCLUSION 64

REFERENCES 66

ABBREVIATIONS

ANN	Artificial Neural Network
AR	Auto Regression
BIC	Bayesian Information Criterion
IID	Independent and Identically Distributed
ISO	Independent System Operator
MA	Moving Average
MAPE	Mean Absolute Percentage Deviation
MSE	Mean Squared Error
VPP	Virtual Power Plant

LIST OF FIGURES

Figure 2.1.	State diagram and transition matrix of simple Markov process (Sunny- Cloudy example)	12
Figure 2.2.	Transition grid for Markov process	12
Figure 2.3.	Urn-ball illustration of discrete symbol HMM	14
Figure 2.4.	Observation sequence generation using HMM	17
Figure 2.5.	Graphical illustration of sequence generation by HMM	18
Figure 3.1.	Data center load, Case I	30
Figure 3.2.	Load profile of 15 selected days, Case I	31
Figure 3.3.	Monthly average load with linear and first order Fourier fit, Case I	31
Figure 3.4.	BIC and execution time vs number of states, Case I	32
Figure 3.5.	Fitted values from two models, Case I	33
Figure 3.6.	Normalized Load vs Time, Case II	34
Figure 3.7.	Load and pure sinusoidal curve, case II	36
Figure 3.8.	BIC and execution time vs number of states, Case II	37
Figure 3.9.	Fitted values from two models, Case II	37
Figure 3.10.	Normalized Load vs Time, Case III	38
Figure 3.11.	Load and pure sinusoidal curve, Case III	39
Figure 3.12.	BIC and Execution time vs number of states of model 1, Case III	40
Figure 3.13.	Fitted values from three models, Case III	41
Figure 4.1.	Forecast MAPE values, Case I	43

Figure 4.2.	Load forecast, Case I	44
Figure 4.3.	Forecast MAPE values, Case II	45
Figure 4.4.	Load forecast, Case II	46
Figure 4.5.	Forecast MAPE values, Case III	46
Figure 4.6.	Load forecast, Case III	47
Figure 4.7.	Determining weights, SEMA model	50
Figure 4.8.	Synopsis of ARNN model [23]	52
Figure 4.9.	Models comparison	52
Figure 4.10.	Solar and Wind power profile for a typical day	57
Figure 4.11.	Locational Marginal Price (LMP) for a typical day	57
Figure 4.12.	Natural gas generator cost curve	58
Figure 4.13.	Natural gas generator efficiency curve [10]	58
Figure 4.14.	Profiles for actual load	61
Figure 4.15.	Profiles for actual and forecast load	62
Figure 4.16.	Flowchart of dispatching	63
Figure 4.17.	Penalties for inaccurate forecast	63

LIST OF TABLES

Table 4.1.	Training MAPE, SMA model	49
Table 4.2.	$ARIMA_{(p,d,q)}$ model order for three cases	51
Table 4.3.	Forecast Evaluation, Case I	53
Table 4.4.	Forecast Evaluation, Case II	53
Table 4.5.	Forecast Evaluation, Case III	54
Table 4.6.	Model summary, Case I	54
Table 4.7.	Model summary, Case II	54
Table 4.8.	Model summary, Case III	54
Table 4.9.	Penalty for inaccurate forecast (\$)	61

ABSTRACT

DATA CENTER LOAD FORECAST USING DEPENDENT MIXTURE MODEL

MD RIAZ AHMED KHAN

2016

The dependency on cloud computing is increasing day by day. With the boom of data centers, the cost is also increasing, which forces industries to come up with techniques and methodologies to reduce the data center energy use. Load forecasting plays a vital role in both efficient scheduling and operating a data center as a virtual power plant. In this thesis work a stochastic method, based on dependent mixtures is developed to model the data center load and is used for day-ahead forecast. The method is validated using three data sets from National Renewable Energy Laboratory (NREL) and one other data centers. The proposed method proved better than the classical autoregressive, moving-average, as well as the neural network-based forecasting method, and resulted in a reduction of 7.91% mean absolute percentage error (MAPE) for the forecast. A more accurate forecast can improve power scheduling and resource management reducing the variable cost of power generation as well as the overall data center operating cost, which was quantified as a yearly savings of \$13,705 for a typical 100 MW coal fired tier-IV data center.

CHAPTER 1 : INTRODUCTION

1.1 Background

Data centers are the backbone of current world economy as they power up the organizations of all scales. For years data centers have been the key for modern software technology, performing demanding role for the business expansions. With high availability, reliability, efficiency and staggering scalability, more and more enterprises are moving towards cloud computing. All those error prone paper-pencil-documentation methods today have completely been replaced with the ease of usability of data centers as a whole. As a result the global electricity demand by the data centers is ever increasing day by day. In 2013, 1.5 MW per 100 MW production of electricity was consumed by the data centers to power up the business enterprises worldwide [1]. And the growth of data centers is showing no sign of slowing down. Cisco conducted a study to forecast the growth of global data centers [2]. It was reported that by 2019, 86% of the workloads will be processed by cloud data centers, a 3.3 fold increase from 2014. About 2 billion users will use personal cloud storage. Data created by the devices will reach 507 Zettabytes (ZB) per year by 2019, from just 135 ZB in 2014. To cope with this industry trend, giants like Google, Microsoft are coming forward, building new data centers. With 15 data centers worldwide, Google announced to open 12 more data centers by 2017 [3]. Microsoft announced 5 new, that will add to their 22 current data centers [3].

The growth of cloud is making the data centers significant electricity consumers. In 2013, data centers in USA only consumed 91 billion kW-hr electricity [4]. This is equivalent to 34 500 MW power plants, dedicated to power up the data centers with

\$10.92 billion annual production cost. By 2019 there will be 17 more required. With the current growth rate, 8% of the global electricity is forecasted to be consumed by data centers in 2020 [5]. As the power consumption and electricity prices are going up, energy cost is receiving more attention. Efficiency assessment became a priority in the industry for the high cost involved in the data centers. Substantial work have been done to increase the data center operating efficiency in the recent past [6], [7], [8]. Proper power scheduling and management can play a role to reduce the energy consumption and reduce the cost and carbon footprint. For proper energy management, one of the primary tasks is to forecast the load. In the era of deregulated energy market, load forecast is vital with applications like energy purchasing and on-site generation, maximize the use of renewable resource and thus direct towards reduced cost and carbon emission. Load forecast will also be significant since the industry are going for more renewable energy. Apple claimed to be already 100% green. Google announced its goal for 100% renewable power, purchasing 2.2 GW of clean energy to date. Amazon, by far the industry leader with 31% market share of the total cloud based services [9], also committed for 100% renewable energy. Microsoft is using wind and solar power to run their data centers. The industries are mainly focusing on wind, solar and hydro-electric power, which are intermittent in nature. Along with solar and wind forecast, load forecast is a key factor for proper hour by hour operating schedule, maximizing renewable penetration and minimizing cost and carbon footprint. Load forecast can also be useful to enable data centers to operate as virtual power plants (VPP) [10]. Accurate load prediction can also expedite the dynamic thermal load balancing by coordinating cooling units and improving the overall efficiency of the data center.

1.2 Datacenter overview

A data center is a facility to house large number of components for computing, networking, storage. In addition, it provides infrastructure for power distribution and cooling systems. In a broader sense, data center load can be divided into two categories, IT load and non IT load. Non IT load can further be categorized into cooling load, lighting, UPS and other miscellaneous load. According to the data provided by Lawrence Berkeley National Laboratory, in a typical data center 46% of the total load is the IT load, resulting in a power usage effectiveness (PUE) of 2.17 [11]. PUE is a metric used by the industries to measure how efficiently power is being used and given by the ratio of overall data center load to the IT load. The trend of designing facility has significantly changed over the last decade and industries are achieving PUE as low as 1.12 [12].

The IT/server load of the data center shows great variability as it depends on user activity. The server load fluctuates with high variance throughout the day. The cooling load includes chillers, pumps, radiators, air side economizers. These equipments are dedicated to maintain the environment temperature within limits. The cooling load also varies with the server load. All other loads are almost constant. Peak to average load ratio generally lies within the range of 1 to 4 [13].

1.3 Literature review

As the energy consumption by the data centers and electricity price are going up, energy efficiency is fast becoming a priority. Various ways to improve data center energy efficiency are presented in [6], [7], [8]. Many researchers focused on improved

mechanical designs, proper sealing, proper sizing, optimized air flow, using air-side economizers and strategic IT load distribution like distributed server power management, peak provisioning, server-virtualization, standardization, which can reduce the overall energy consumption. Power management is also another area which has been studied by the researchers looking for cost reduction. An adaptive server workload scheduling technique has been presented to optimize the data center profit in [14]. Regression tree was applied to predict the load with reported prediction and validation mean squared error (MSE) 9.9% and 12% respectively, to feed into the optimizer. A hierarchical server task scheduling method was proposed to optimize the application placement in [15]. This algorithm is applicable once the servers receive application requests; forecast of the tasks had not been investigated. An online control algorithm was developed to minimize the time average operating cost of the data center in [16]. In this work, the workload is considered as independent and identically distributed (IID) random variables and modeled via IID model . Although the algorithm was claimed to be applicable for non-IID load as well, no non-IID process was undertaken for modeling the load. It was assumed that enough number of servers can be activated instantly, which can greatly affect the optimal server on-off states, neglecting the provision of unit commitment.

An adaptive data center job scheduler was proposed in [17]. The authors showed that the use of green energy can be improved by proper forecasting of wind and solar energy. Statistical profiling based and power re-routing based power provisioning techniques were proposed in [18] and [19]. All these research works address the importance of forecasting the load, but a solution was not proposed.

Attention has been drawn on the load forecast in [20], [21] and [22]. A simple

exponential smoothing forecast mechanism has been used in [20]. A two step forecast technique has been discussed in [21] and [22]. In the first step, an extra layer of cubic spline was added with the exponential smoothing and then high peak was predicted by auto regression. A model had been presented for the prediction of cloud data using neural network blended with auto regression in [23]. A $3 \times 2 \times 1$ neural network has been used, where the input layer takes the previous three values as the input. Since the data center load possess high variance and seasonality over year, an auto regression of order 3 is just not appropriate. A load forecasting method was described to predict Google hostload using Bayesian model in [24]. It extracted 10 features out of the emulated time series data as covariates. The time series data was modeled as small packets over a chosen time duration and analysis was done to evaluate the mean forecast for the time duration. This forecast technique was reported to be performing better than eight other traditional forecast method based on auto regression (AR), moving average (MA) and artificial neural network (ANN).

Based on the study, it was found that researchers focused on increasing the data center energy efficiency and came up with wide range of ideas like improved mechanical design, oversubscribing, power provisioning, harnessing the green power. Most of them addressed the importance of load forecast, but a convenient solution was not proposed. In this research work, a technique to forecast the data center load was developed. The methodology was developed based on hidden Markov model (HMM) Bayesian Inference. Due to the high variance of the data center load profile, it was modeled as a finite mixture of multiple groups. It bypasses the complexity of extracting the features from the past history making the method relatively simple and robust. Along with the time stamp, only

temperature was used as exogenous variable for one of three cases. The model performed well even without the temperature or any climate data for the other two cases.

1.4 Contributions

The main contribution of this thesis is the development of a new method for data center load forecast using a dependent mixture model based on Bayesian inference.

1.5 Thesis outline

This thesis has been organized as follows: Chapter 2 describes the theory behind the forecast technique described in this thesis. In chapter 3, the technique was applied on three real data sets. It describes the step by step process of building the model for the specific data and assessed in terms of mean absolute percentage deviation (MAPE). Chapter 4 presents the results of other methods of forecast for the same data sets and compared with different other methods. The advantage of using the developed method was also quantified in terms of yearly savings. Chapter 5 concludes the work, with limitations and future development related to this study.

CHAPTER 2 : THEORY

Forecasting is making the best use of information in hand to make prediction about the future course. Electric load forecasting is immensely important in the planning and operation of power system. In the age of deregulated energy market it is even more important to direct towards optimal use of resources and planning for the future establishments of power plants, transmission networks. Usually the independent system operators (ISO) in the United States follow econometric model for long term and short term forecast [25], [26], [27]. MISO, PJM, New England ISO take the economic factors, temperature factors and historical load for up to 15 years to develop the forecast method. But load behavior of utilities and data center differ in a number of ways. First, the economic factor. The utilities usually incorporate economic variables to represent different sectors of customer class (residential, industrial and commercial). For the case of cloud, mainly three kinds of services, software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) are provided [28]. The first two kinds are related to the soft service, while IaaS also includes the hardware as a part of the services. The SaaS and the PaaS, both are deliverable over the web. But they have little difference in job nature. The SaaS applications are usually light and robust requiring comparatively less time than the PaaS applications. This makes sense that the PaaS applications would display greater variance than the PaaS applications. Including the portion of each type load would be appropriate while modeling the load of a mega data center, where all kinds of services are being provided. But in a case where the size of the data center is smaller and dedicated for a particular kind of service, then this fact can safely be ignored.

Geographic location is another key factor that has been traditionally taken into account for future load growth. But this is primarily essential in the T&D planning stage, where future load, in magnitude has to be predicted, which heavily depends on the geographic location. This is a part of long term load forecasting task, having little impact on the short term forecasting on day ahead basis. This work does not aim to build new infrastructure, rather it attempts to uncover the daily load behavior for an already existing data center. So this factor was also treated as irrelevant for the case.

Unlike the user class and geographic location, climate effect is obvious in modeling a data center load. Climate condition invokes a direct relation between the load. This makes sense as cooling is an integrated part of a data centers. Heat removal is one of the most essential tasks of data center. Heavy equipments like chillers, cooling towers, evaporators, condensers, water pumps are engaged to maintain the data center environment temperature and humidity. So when modeling the total data center load, temperature is essential to be taken into account. Higher ambient temperature raises the total system load, as the cooling system has more work to do to maintain the ambient temperature.

Making a temperature based load forecast model creates a minor problem. If one includes temperature as covariate, it has to be kept in mind that temperature is also another thing to forecast. Weather forecast is itself a vast topic involving complexity of chaotic nature and partial differential equations governing the atmosphere. With the advancements of data mining and supercomputers, models have been developed for predicting the weather up to six days with 90% accuracy [29]. Since in the real scenario there would be just forecast temperature, necessary transformation of the temperature was made to emulate forecast temperature.

This thesis presents the development of a data center load forecasting technique using dependent mixture model, which relies on a hidden Markov model (HMM), Bayesian inference in simple form. The underlying basis of the theory is explained in section 2.1. Section 2.2 describes how the HMM was employed to forecast load.

2.1 Dependent mixture model

Over-dispersion is common when dealing with a series having multiple seasonality. To deal with the problem, multimodal distribution is used [30]. The response is considered to be aroused from multiple subpopulations or states. This is the underlying basis of the mixture model. The model does not require that an observed data set should pinpoint the corresponding subpopulation, to which an individual observation belongs to. There are a number of approaches, attempt to derive the properties of the overall population. In HMM, the subpopulation or state, depends on the previous state. The transition of the states is modeled through a first order Markov process. Given that the subpopulations or states are hidden, the model is also called Hidden Markov Model (HMM). The HMM is the simple form of the dynamic Bayesian network. This offers a comprehensive tool for representing the probability distribution over sequence of observations.

The HMM gets the name from two of its properties. First, this is assumed that an observation (the output) at any given time t , is generated through some process, whose state is not visible to the observer, but inferred. Second, the invisible process is governed by Markov property. That is, at any given time the state envelops all the necessary historical information. An introduction of simple and hidden Markov process will be

discussed in the following two subsections:

2.1.1 Markov process

A Markov process is a stochastic process that satisfies Markov property. Markov property is a memoryless process of generating probability distribution of future states, that depends only on the present state of the process, not on the preceding states. If the random variables, known as states, undergoes the Markov property then the sequence generated is called simple Markov chain.

Suppose $Y_1, Y_2, Y_3, \dots, Y_T$ form a sequence, where each output belongs to a set of finite states $\{S_1, S_2, S_3, \dots, S_N\}$. Now the sequence will be called Markov chain if it holds the following Markov property:

$$Pr(Y_{T+1} = S | Y_1 = S_1, Y_2 = S_2, Y_3 = S_3, \dots, Y_T = S_T) = Pr(Y_{T+1} = S | Y_T = S_T) \quad (2.1)$$

The transition of the states at any given time, say from $t = T$ to $t = T + 1$ is defined by the transition matrix. The transition matrix A is a $N \times N$ matrix, where N is the number elements of the finite state space set, i.e the number of states. Each element of the matrix, a_{ij} , is a non-negative number and defines the probability of transition to state j from state i at any given time. Denoting the actual state at time t as r_t , this can be written as:

$$a_{ij} = Pr(r_{t+1} = S_j | r_t = S_i) \quad (2.2)$$

Any row of the transition matrix represents the probability of going to all the states

from a definite state. Since the number of states is finite, the summation of any row is unity.

The concept of simple Markov process can be explained by a simple example. Suppose, the random variable is the probability of a day being sunny or cloudy. From the history, it was inferred that given that currently the day is sunny, there is 70% chance for the next day being sunny. Also, if a day is currently cloudy, there is 60% chance for the next day to become cloudy. The state diagram and the transition matrix for the system is shown in Figure 2.1. Now, for this simple case, there are two definite disjoint states. Now to get the state probability distribution for the next day, one just needs the transition matrix and probability distribution of the current state. In this simple example, it was assumed that there is a fifty percent chance for a day to be sunny or cloudy. So the probability of next day being sunny can be given by conditional probability:

$$Pr(sunny)Pr(sunny|sunny) + Pr(cloudy)Pr(sunny|cloudy) = 0.5 \times 0.7 + 0.5 \times 0.4 = 0.55.$$

Similarly the probability of next day being cloudy can be calculated as 0.45. This task can be written in a compact form as:

$$st_{T+1} = st_T \times A \quad (2.3)$$

In equation (2.3), the left hand side is a row matrix, and the number of columns equals the number of states. This gives the probability distribution of the next state. Right hand side of the equation is the product of current probability distribution and the transition matrix. For the simple sunny-cloudy example, there are a 1×2 and a 2×2 matrix on the right hand side. The product is a 1×2 matrix, which is the probability

distribution of the next state. Now this simple system can be extended for N states. Then the transition matrix becomes an $N \times N$ matrix and the probability distribution is a $1 \times N$ row matrix. The state diagram then becomes a transition grid as shown in Figure 2.2.

Equation (2.3) can still be applied to find the next state probability distribution.

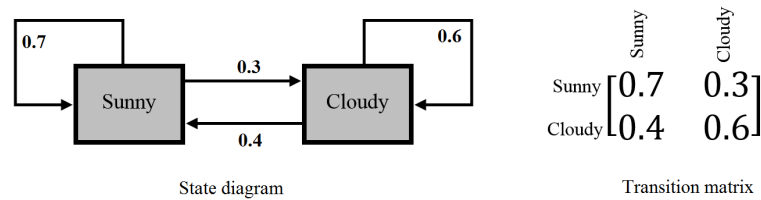


Figure 2.1. State diagram and transition matrix of simple Markov process (Sunny-Cloudy example)

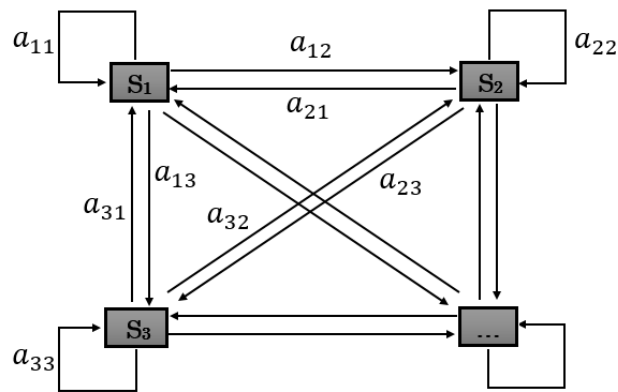


Figure 2.2. Transition grid for Markov process

Now equation (2.3) can be applied recursively if one wants to infer the state probability distribution more than one step ahead. The general form for this inference can be written as:

$$st_{T+n} = st_T \times A^n \quad (2.4)$$

In equation (2.4), st_T becomes the initial probability distribution for the Markov process.

To denote initial probability distribution, τ will be used in the rest of the literature.

2.1.2 Hidden Markov model

As stated earlier, in the hidden Markov model, the states are modeled as hidden to the observer. Unlike the simple Markov process, where the states themselves are the outcomes, in hidden Markov model the outputs are assumed to be generated from one of hidden or latent states. So the difference is made by adding one more layer keeping the states hidden. Each state has its own probability distribution to generate an output from a finite set. Hidden Markov model is thus a doubly stochastic process with an underlying stochastic process hidden/not observable.

The concept of hidden Markov model can be best described with the classic urn problem [31]. It is assumed that there is set of urns, each containing some colored balls. For example the first urn contains 5 red, 3 yellow, and 2 black balls, the second urn contains 12 red, 5 yellow, and 13 black balls. Each of the urns has its own probability for generating the output. In this case, the output is the color of the ball. This is called the emission probability. Now by some random process one urn is selected and one ball is drawn from the selected urn. The ball is replaced and again by the random process another urn is selected and a ball is drawn. It is to be noted that the selection process of the urn permits a repetition of urn. The output of the system, Y is a sequence of colors of balls.

Figure 2.3 illustrates the urn-ball system. Here each urn symbolizes one state. The observer only observes the color sequence of the ball. The information, each observation is coming from which urn, is hidden. For each urn, i.e state, the color probability is well defined. The selection of urn is governed by simple first order Markov process. Now the problem here is finding the sequence of the urns, that maximizes the likelihood of the

observed color sequence. This is a double stochastic process. First, within a state one from multiple possible values (colors) is observed. Second, the choice of the urn. This is another stochastic process and is hidden. So the problem is to find the sequence of the hidden states, which maximizes the likelihood of the observed color sequence.

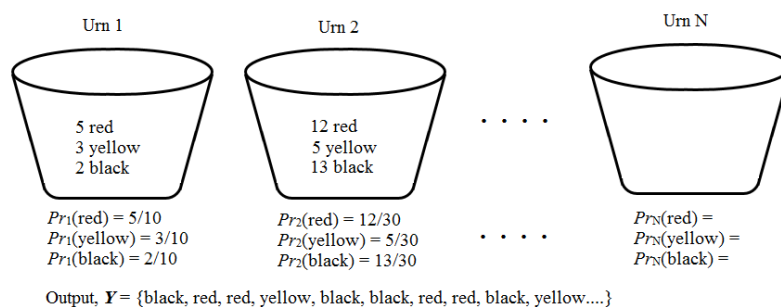


Figure 2.3. Urn-ball illustration of discrete symbol HMM

2.1.3 Hidden Markov model elements

In this subsection the HMM will be formally defined in mathematical terms. Most of the theory is adapted from the classical Rabiner tutorial paper on HMM [31]. The problems associated with the model and describe the classical methods of handling the problems will be discussed. First the discussion will be confined to discrete outputs. Then the idea will be extended for continuous output.

Suppose there are some states in a system $\mathcal{S} = \{S_1, S_2, S_3, \dots, S_N\}$. Some outputs $\mathbf{Y} = \{Y_1, Y_2, Y_3, \dots, Y_T\}$ are observed in T time segments. The outputs are coming from a finite set of distinct values, \mathcal{U} , where $\mathcal{U} = \{u_1, u_2, u_3, \dots, u_M\}$. The actual state at any time t is denoted by r_t . To model the system with HMM, five elements are required. Two of them are model specifications and the remaining three are model parameters.

2.1.3.1 Hidden Markov model specifications

1. **The number of hidden states:** This is denoted by N . Although the states are hidden to the observer, it might carry physical significance. For example, in the urn example, the state represents urn. So the number of urns becomes the number of states. The states undergoes a first order Markov process. Generally, the states are so interconnected that transition from any state to any state is possible. Since the states are hidden, there is no way of knowing this prior to the modeling. It has to be specified. Bayesian Information Criterion (BIC) will be used to choose the number of states.

2. **The number of distinct observations within a state:** This represents the discrete alphabet of the observation symbols and relates to the physical output. The discrete symbols will be denoted as $U = \{u_1, u_2, u_3, \dots, u_M\}$, where M is the alphabet size. For the urn example, the observation symbols are three colors, $colors = \{red, yellow, black\}$ and $M = 3$.

2.1.3.2 Hidden Markov model parameters

1. **Transition matrix:** This is a $N \times N$ matrix, portraying the state transition probability distribution. This will be denoted by $A = \{a_{ij}\}$, where

$$a_{ij} = Pr(r_{t+1} = S_j | r_t = S_i), \quad 1 \leq i, j \leq N \quad (2.5)$$

2. **Emission probability distribution:** Within each state, each symbol has some

probability to appear as output. Within state i , it will be denoted by $E = \{e_i(k)\}$, where

$$e_i(k) = Pr(u_k \text{ at } t | r_t = S_i), \quad 1 \leq i \leq N, 1 \leq k \leq M \quad (2.6)$$

3. **Initial distribution:** This represents the initial state distribution. It will be denoted by $\tau = \{\tau_k\}$, where

$$\tau_k = Pr(r_1 = S_k), \quad 1 \leq k \leq N \quad (2.7)$$

Given the model elements, HMM can be used to generate a sequence of observations. Figure 2.4 shows the algorithm for generating the observation using HMM. At the beginning of the process, it picks one state according to the initial state probability distribution and generates an output according to the emission probability. Then at the second time slot it switches to one state (may come back to the previous state as well) and again generates an output according to the emission probability distribution. This process goes on until the final time segment. This procedure can be used to model a given sequence of observation. Figure 2.5 illustrates a graphical representation of the generation of a sequence through HMM. Each state is associated with two probability measures. One relates to the conditional probability of generating an output within that state, other one relates to the conditional probability of switching to other state from that state. Thus the HMM creates a simple dynamic Bayesian network. The initial probability distribution it creates the root node, which is the state at the first time interval. Every other state becomes an intermediate node and every output is a leaf node. The network is a very simple one for the Markov property of the states. All the states are only directed by just

one preceding mode and all the outputs are connected to just one state.

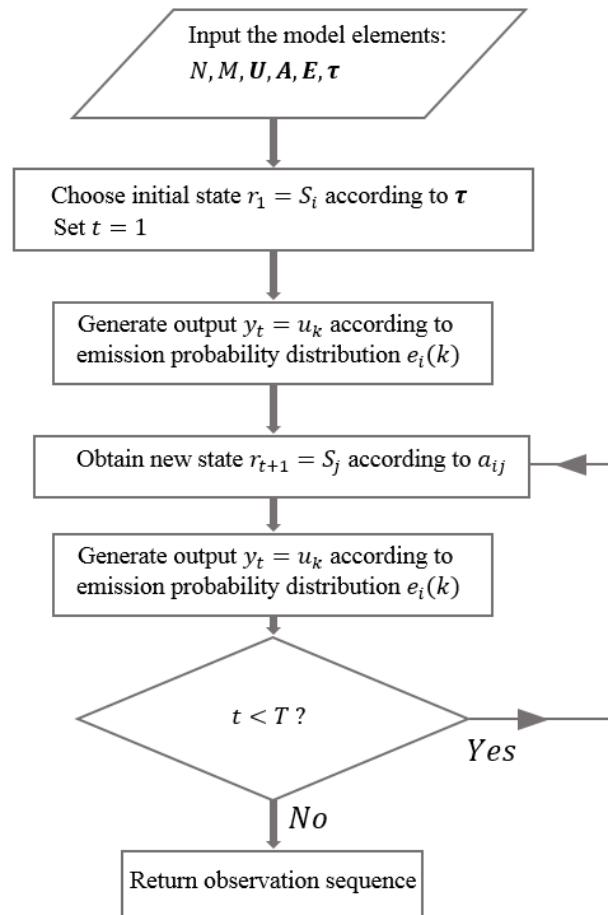


Figure 2.4. Observation sequence generation using HMM

2.1.4 Evaluating hidden Markov model parameters

For a complete description of HMM, two specifications, N and U and three parameters, A , E and τ are required. How the parameters of a HMM can be evaluated will be discussed in this section. For convenience, the compact symbol ϕ will be used to represent the model parameter. That is, $\phi = \{A, E, \tau\}$.

To model a sequence of observations by HMM, there are three fundamental

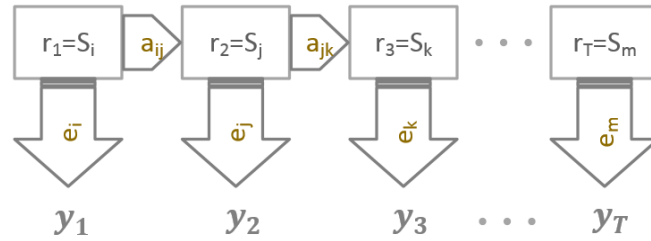


Figure 2.5. Graphical illustration of sequence generation by HMM

problems that must be solved:

- Problem 1: Given a HMM $\phi = \{\mathbf{A}, \mathbf{E}, \boldsymbol{\tau}\}$, how to compute the probability of a given observation sequence $\mathbf{Y} = \{y_1, y_2, y_3, \dots, y_T\}$.
- Problem 2: Given a HMM $\phi = \{\mathbf{A}, \mathbf{E}, \boldsymbol{\tau}\}$, how to choose the state sequences $\{r_1, r_2, r_3, \dots\}$, that will maximize the likelihood of a given observation sequence $\mathbf{Y} = \{y_1, y_2, y_3, \dots, y_T\}$.
- Problem 3: How to adjust the model parameters $\phi = \{\mathbf{A}, \mathbf{E}, \boldsymbol{\tau}\}$, that will maximize the conditional probability $Pr(\mathbf{Y}|\phi)$.

Problem 1 is known as the *Evaluation* problem of HMM. It is mainly associated with the computation of the probability of a given sequence according to the given model.

Problem 2, which is known as the *Decoding* problem of HMM, attempts to reveal the hidden part. It tries to find the state sequence that maximizes the optimality of the sequence based on some criterion. Problem 3 is the *Learning* problem of HMM. It adjusts the model parameters $\phi = \{\mathbf{A}, \mathbf{E}, \boldsymbol{\tau}\}$, so an optimal solution comes out through iteration.

In the following section, the mathematical formulation of and present state of art of

solving these three fundamental problems will be discussed.

Solution to Problem 1: The *Evaluation* problem deals with the computation of the probability of a given observation sequence $\mathbf{Y} = \{y_1, y_2, y_3, \dots, y_T\}$ for a given model $\phi = \{\mathbf{A}, \mathbf{E}, \boldsymbol{\tau}\}$. In mathematical notation, $Pr(\mathbf{Y}|\phi)$ is to be evaluated. From the preceding discussion, it is clear that this conditional probability is an outcome of a joint distribution. First, the probability of the emission; second, the probability of the states. An explicit state sequence $\mathbf{R} = r_1, r_2, r_3, \dots, r_T$ is considered. Now the probability of the observation sequence for this explicit state sequence can be given by:

$$Pr(\mathbf{Y}|\mathbf{R}, \phi) = \prod_{t=1}^T Pr(y_t|r_t, \phi), \quad 1 \leq t \leq T \quad (2.8)$$

Equation (2.8) can be written in terms of emission probabilities as follows:

$$Pr(\mathbf{Y}|\mathbf{R}, \phi) = e_{r_1(y_1)} \cdot e_{r_2(y_2)} \cdot \dots \cdot e_{r_T(y_T)} \quad (2.9)$$

The probability of the explicit state sequence is given by:

$$Pr(\mathbf{R}|\phi) = \tau_{r_1} \cdot a_{r_1 r_2} \cdot a_{r_2 r_3} \cdot \dots \cdot a_{r_{T-1} r_T} \quad (2.10)$$

The joint probability of the observation and the explicit state occurring simultaneously is simply the product of the two expressions:

$$Pr(\mathbf{Y}, \mathbf{R}|\phi) = Pr(\mathbf{Y}|\mathbf{R}, \phi) \times Pr(\mathbf{R}|\phi) \quad (2.11a)$$

$$\text{or, } Pr(\mathbf{Y}, \mathbf{R}|\phi) = \tau_{r_1} \cdot e_{r_1}(y_1) \cdot a_{r_1 r_2} \cdot e_{r_2}(y_2) \cdot a_{r_2 r_3} \cdot e_{r_3}(y_3) \dots a_{r_{T-1} r_T} \cdot e_{r_T}(y_T) \quad (2.11b)$$

$$\text{or, } Pr(\mathbf{Y}, \mathbf{R}|\phi) = \tau_{r_1} \cdot e_{r_T}(y_T) \prod_{t=1}^{T-1} e_{r_t}(y_t) \cdot a_{r_t r_{t+1}} \quad (2.11c)$$

Equation (2.11) gives the probability of the observation sequence for an explicit state sequence. The objective here is to find the probability of the observation sequence, unconditional to a definite state. Taking sum over all the states gives the necessary probability, irrespective of a definite state sequence:

$$Pr(\mathbf{Y}|\phi) = \sum_{r_1, r_2, \dots, r_T} \tau_{r_1} \cdot e_{r_T}(y_T) \prod_{t=1}^{T-1} e_{r_t}(y_t) \cdot a_{r_t r_{t+1}} \quad (2.12)$$

Although equation (2.12) gives the necessary probability in a straight forward manner, it is computationally exhaustive. The summation has to be performed over all the possible states, which is N^T , which is a very large number for even small values of N and T . The dimensionality problem is handled with forward-backward procedure. First the forward and backward terms α and β are defined, respectively:

$$\alpha_t(i) = Pr(y_1, y_2, \dots, y_t, r_t = S_i | \phi) \quad (2.13)$$

$$\beta_t(i) = Pr(y_{t+1}, y_{t+2}, \dots, y_T, r_t = S_i | \phi) \quad (2.14)$$

The forward term defines the partial probability of the observations up to time t and the

state S_i at that time. This can be calculated inductively by:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) \cdot a_{ij} \right] \cdot e_j(y_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N \quad (2.15)$$

Equation 2.15 is interpreted as follows: At time $t+1$ state S_j can be achieved from any of the N possible states. Since the forward term is the joint probability of observing y_1, y_2, \dots, y_t and having state S_i at time t , the product $\alpha_t(i)a_{ij}$ is the joint probability of observing y_1, y_2, \dots, y_t and reaching state S_j at time $t+1$ via state S_i . The summation of this joint factor over all the states gives the probability of observing y_1, y_2, \dots, y_t and reaching state S_j at time $t+1$. This reserves all the possible state scenarios in the preceding time frames. The forward term is initialized according to the initial state probabilities by $\alpha_1(i) = \lambda_i \cdot e_i(y_1)$ for all possible values of N . The terminal forward term, $\alpha_T(i)$ is the probability of observing \mathbf{Y} with S_i as the final state. Since the states are disjoint, summing all the forward term (over all the states) gives the necessary probability:

$$Pr(\mathbf{Y}|\phi) = \sum_{i=1}^N \alpha_T(i) \quad (2.16)$$

The backward term can similarly be utilized to calculate the observation probability.

Rather than calculating the partial probability by a forward induction, the backward term is used to calculate the partial probability through backward induction. The backward induction is given by:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} \cdot e_j(y_{t+1}) \beta_{t+1}(j) \quad (2.17)$$

The backward term is initialized arbitrarily by defining $\beta_T(i) = 1$ for all i . In the induction

procedure, which involves calculating $\beta_t(i)$ from $\beta_{t-1}(i)$, all the possible states are considered with associated transition. Either forward term or backward term can be utilized to solve the first fundamental problem. And both of them are used extensively to solve the remaining two.

Solution to Problem 2: The *Decoding* problem deals with finding the optimal states, which will maximize some score. This is different from *Evaluation*, where an exact solution can be possible. But in *Decoding*, multiple solutions are possible based on the criterion of optimality. One choice is to choose the states that will maximize the individual likelihood i.e $Pr(r_t|y_t, \phi)$, of the observations. But this criterion does not establish any bridge between states. So a problem could arise when an individual state comes with a zero or low probability, or makes transition to the next individually most likely state with zero or low probability. Although taking pairs or triplets could be a reasonable solution, the most used technique is to choose the best single path in terms of likelihood. In this case, one path is chosen so that $Pr(\mathbf{R}|\mathbf{Y}, \phi)$ is maximized. This is known as Viterbi algorithm [32].

The Viterbi algorithm is based on dynamic programming. This uses the fact that *If policy A has a best solution s, then another policy B must contain s in the best solution if A is a sub policy of B.* A score variable χ is defined to determine the highest probability along a single path for the first t observations ending at state S_i .

$$\chi_t(i) = \max_{r_1, r_2, \dots, r_{t-1}} Pr(r_1, r_2, \dots, r_t = S_i, y_1, y_2, \dots, y_t | \phi) \quad (2.18)$$

The variable is initialized by:

$$\chi_1(i) = \tau_i e_i(y_1), \quad 1 \leq i \leq N \quad (2.19)$$

and calculated inductively by:

$$\chi_{t+1}(j) = [\max_i \chi_t(i) a_{ij}] \cdot e_j(y_{t+1}), \quad 1 \leq j \leq N \quad (2.20)$$

It should be noted here that the Viterbi algorithm works almost in a similar way like in the forward procedure. In any time segment t , there are N "best" score variables up to that point. Now for the next time segment, each of the N states is reached via N best paths and the path having the best score is retained. When the procedure terminates when $t = T$, the best single path is retrieved by backtracking the states.

Solution to Problem 3: The third problem associated with HMM deals with finding the model parameters $\phi = \{\mathbf{A}, \mathbf{E}, \tau\}$ to maximize the likelihood of the observations.

Although there is no analytical way to find the model parameters for maximum likelihood [31], a local maximum can be achieved using Expectation Maximization (EM) algorithm, which is an iterative procedure, classically developed by Baum et al [33].

To begin with, for adjusting the model parameters, a new term $\omega_t(i, j)$ is introduced to define joint the probability of having state S_i at some time and state S_j at the next time, for the given observation sequence and model parameters:

$$\omega_t(i, j) = Pr(r_t = S_i, r_{t+1} = S_j | \mathbf{Y}, \phi) \quad (2.21)$$

Now $\omega_t(i, j)$ can be written in terms of forward and backward terms:

$$\omega_t(i, j) = \frac{\alpha_t(i)a_{ij}e_j(y_{t+1})\beta_{t+1}(j)}{P(\mathbf{Y}|\phi)} \quad (2.22)$$

The numerator of equation (2.22) gives the probability of seeing S_i at time t and S_j at time $t + 1$ for the given observation sequence and model. The denominator is to get the required probability measure. It can be written as a double summation of the numerator term, over all the states as $\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i)a_{ij}b_j y_{t+1} \beta_{t+1}(j)$. Now another variable $\mu_t(i)$ is defined as the probability of having the state S_i at time t for the given observation sequence and model parameter:

$$\mu_t(i) = Pr(r_t = S_i | \mathbf{Y}, \phi) \quad (2.23)$$

Now the two conditional probability terms, ω and μ can be correlated. The former term evaluates the joint probability of having two states at two consecutive times. Taking the summation of $\omega_t(i, j)$ over all the states at time $t + 1$ gives the probability of having some state at earlier time slot. Mathematically,

$$\mu_t(i) = \sum_{j=1}^N \omega_t(i, j), \quad 1 \leq i \leq N \quad (2.24)$$

Both the probability terms, ω and μ , give us very useful inceptions. Taking summation of $\mu_t(i)$ over the entire time interval, that is $t = 1$ to $t = T$, gives the expected number that the state S_i visited. Excluding t from the time interval, the summation can be interpreted as the expected number of transitions from state S_i . Also, taking the sum of

$\omega_t(i, j)$ over the time interval $t = 1$ to $t = T - 1$, it gives us the expected number of transitions that the a transition from state S_i to S_j occurs. That is,

$$\sum_{t=1}^{T-1} \mu_t(i) = \text{expected number of transitions from } S_i \quad (2.25)$$

$$\sum_{t=1}^{T-1} \omega_t(i, j) = \text{expected number of transitions from } S_i \text{ to } S_j \quad (2.26)$$

Using the above formulas and the concept of counting, method can be developed for readjusting the HMM model parameters. There will be a set of formulas for reestimating the model parameters as follows:

$$\bar{\tau}_i = \text{expected number of times state } S_i \text{ visited at time } (t = 1) = \mu_1(i) \quad (2.27a)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to } S_j}{\text{expected number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \omega_t(i, j)}{\sum_{t=1}^{T-1} \mu_t(i)} \quad (2.27b)$$

$$\bar{e}_j(k) = \frac{\text{expected number of times in state } S_j \text{ and onserving } u_k}{\text{expected number of times in state } S_j} = \frac{\sum_{t=1}^T \mu_t(j)}{\sum_{t=1}^T \mu_t(j)} \quad (2.27c)$$

Based on the equations (2.27a)-(2.27c), model parameters can be readjusted iteratively. It has been shown by Baum et al. that if a new set of model parameters $\bar{\phi} = \{\bar{\mathbf{A}}, \bar{\mathbf{E}}, \bar{\tau}\}$ is computed through equations (2.27a)-(2.27c), then it will 1) be equal to the existing model parameters $\phi = \{\mathbf{A}, \mathbf{E}, \tau\}$, or, 2) would give a better likelihood in

terms of probability [34]. That is, $Pr(\mathbf{Y}|\bar{\phi})$ greater than $Pr(\mathbf{Y}|\phi)$. The model converges better parameters through successive iterations.

2.2 Data fitting and n-step ahead forecast

When a given model is fitted with covariates, it returns model parameters. N sets of coefficients with intercepts associated with the covariates are returned, along with 1 $N \times N$ transition matrix \mathbf{A} and T probability densities of dimension $1 \times N$ for T time slots. The states take the form of N functions $f_{i,1 \leq i \leq N}$, each associated with 1 set of coefficients. These parameters are estimated from training data and used forecast future data.

2.2.1 Data fitting

Let us denote the coefficients by $\theta_{i,1 \leq i \leq N}$ and the state probabilities densities by $st_{j,1 \leq j \leq T}$. If \mathbf{X}_T are T covariates for T time slots, N fitted values are found for each time slot using the state functions and coefficients,

$$\hat{Y}_{ij} = f_i(\theta_i, \mathbf{X}_j), \quad 1 \leq i \leq N, \quad 1 \leq j \leq T \quad (2.28)$$

Left hand side of equation (2.28) contains two indexes. They indicate states and time slots respectively. So $\{\hat{Y}_{ij}\}$ is a $N \times T$ matrix, with columns indicating the states. Now for each time slot, N values are aggregated using the state probability densities. The aggregate estimated value for time slot j is written as \hat{Y}_j and is found by:

$$\{\hat{Y}_j\} = [st_j] \times \{\hat{Y}_{ij}\}, \quad 1 \leq j \leq T \quad (2.29)$$

2.2.2 n-step ahead forecast

Make forecasts for the future data, equations identical to those stated in the preceding subsection were used. The only difference is, the state probability densities had to be calculated, which are used as weights of N forecast values for each time slot.

Suppose T data is used to train the model. Setting $j = T$ in st_j gives the last state probability densities of the given data, which is used as initial state probabilities for forecasting, denoted as st_{init} . For n-step ahead forecast, w first it is necessary to calculate N forecast values for n time slots using equation (2.28), taking j withing the interval of $1 : n$. Each j is associated with each future step. Then N estimated values are aggregated for each step using equation (2.29), where the state probability density is calculated using:

$$[st_j] = [st_{init}] \times \mathbf{A}^j, \quad 1 \leq j \leq n \quad (2.30)$$

2.3 BIC: criterion of number of states

When a dataset is modeled through a statistical tool, model selection is a fundamental task. Different covariates, different distributions, different powers of covariates result in a number of candidate models. When a dataset is modeled through HMM, it is possible to go for a infinite number of states. Bayesian Information Criterion (BIC) criterion was invoked to determine optimal number of states [35]. Mathematically, it is given by:

$$BIC = -2 \times \log_e \hat{L} + k \times \log_e(\text{number of data points}) \quad (2.31)$$

where, $\hat{L} = Pr(y|\hat{\theta}, M)$, maximum likelihood of the model M with parameters $\hat{\theta}$, and k is the number of free parameters to be estimated, including the number of states. The first term in Equation (2.31) is a measure of goodness of fit and the second term is associated with the complexity of the model. Complex model with more parameters often leads to better goodness of fit, but leans itself to overfitting as well. The second term deals with model complexity. It penalizes for adding more states and safeguards against overfitting.

2.4 MAPE: metric of prediction accuracy

There exists a number of metrics to assess time series forecasting models e.g Mean Absolute Error (MAD), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE). In this thesis, MAPE was used consistently. Mathematically, it is given by:

$$MAPE = \frac{1}{T} \sum_{i=1}^T \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \quad (2.32)$$

where \hat{y}_i is the predicted value and y_i is the actual one. Although there is no absolute criterion for a "good" value of MAPE [36], it gives a metric for making comparison of several models.

CHAPTER 3 : PROCEDURES

Dynamic Bayesian based HMM method was applied on three cases. In the first case, the NREL RSF data center hourly load for the year 2011 was used to validate the proposed method. In the second case, load from Microsoft Data center was used. This data set also contains hourly load, but for just a month. For the third case, again Microsoft data center load of one month was used, but with a higher resolution (1 observation/2 minutes, 30 observations/hour).

3.1 Case Study I

Using the depmixS4 package of the statistical computing programming language R [37] [38], a HMM model was made to forecast the load. The NREL RSF data center hourly load for the year 2011 was found in [39]. Hourly data is given for 2011, with a total of $24 \times 365 = 8760$ observations. There are date and time stamps. Total data center load and the breakdown of the load into cooling load, mechanical load, heating load, plug load, server load is also provided. In this case the total data center load was considered for modeling.

Figure 3.1 gives the plot of the response variable as a time series. There is a yearly trend in the load. The average monthly load increases from January to July. The increase rate increases in April. In August, there is a decrease in the load and it goes high again in September. From September through rest of the year, the load profile possesses some constancy.

To check daily seasonality in the load pattern, 15 days were randomly selected.

The load behaviors for these days are depicted in Figure 3.2. For most of the cases the load profile is almost constant and does not show any periodicity.

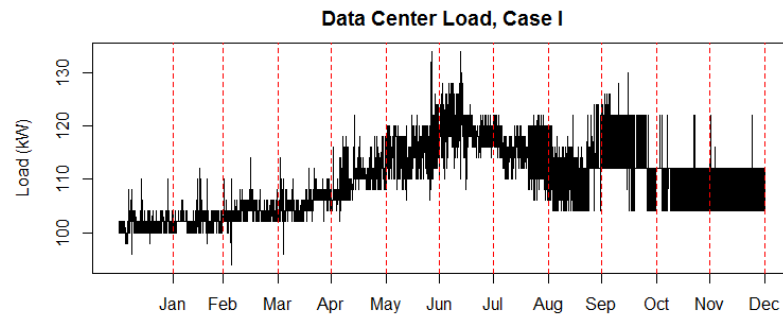


Figure 3.1. Data center load, Case I

The next three sub-subsections describe the methodology of model selection, model assessment, and model validation for this data set.

3.1.1 Model Selection

For this specific case of data set, two HMM models were considered:

$$\text{Model 1 : Load} = a + b \times \text{temperature} + c \times \text{hour} + d \times \text{month} + \varepsilon$$

$$\text{Model 2 : Load} = a + b \times \text{temperature} + c \times \text{hour} + d \times \text{monthly Fourier term} + \varepsilon$$

Temperature is incorporated as a linear covariate in both the models. This is reasonable as the data centers constantly strive to maintain the temperature and humidity within limits.

More HVAC load are expected to be running during the high temperature. The original NREL data was time stamped. The temperature data was collected from [40]. The correlation of load and the temperature was found to be 0.64.

In both the models, hour is incorporated as a simple linear covariate. From Figure 3.2, most of the time the load was almost constant. In few occasions there were trends. Hour

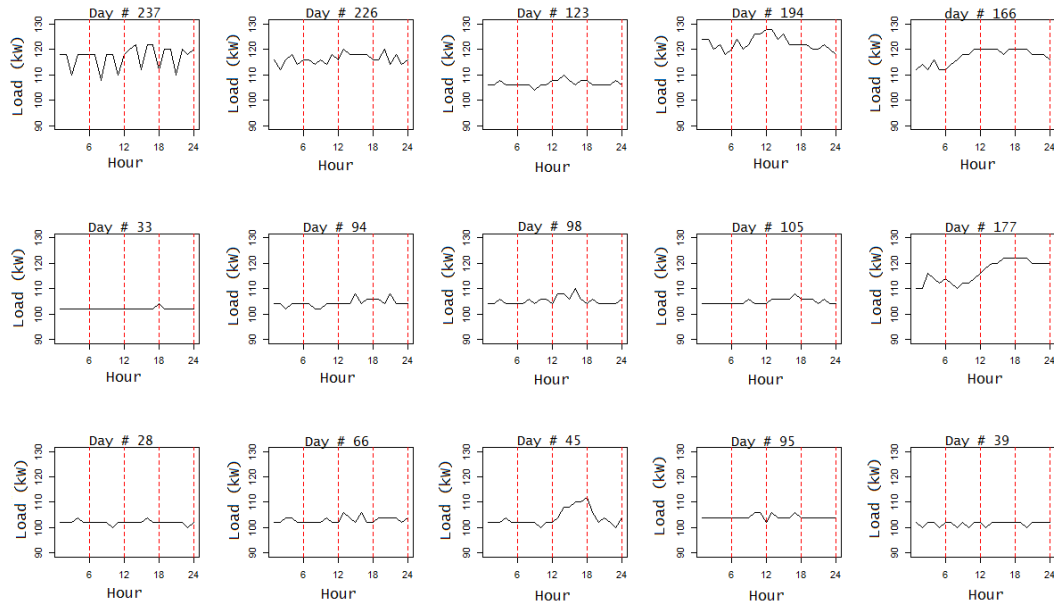


Figure 3.2. Load profile of 15 selected days, Case I

was kept as linear covariate. This is not restricted by the apparent constancy. The model can adjust parameters based on the response. Moreover, since HMM is a stochastic process, including this variable would widen the scope to capture any or no dependency on hour of the day in different states.

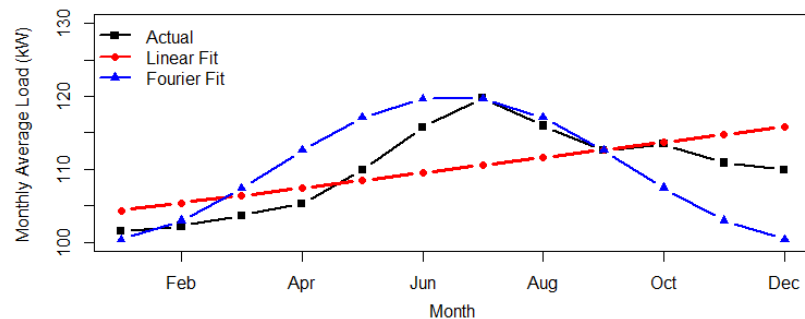


Figure 3.3. Monthly average load with linear and first order Fourier fit, Case I

To reflect the yearly seasonality of the load, two different approaches have been adapted. In the first model, month is taken as a linear covariate, whereas in the second model, it is transformed in a Fourier term. Figure 3.3 shows the monthly average load

with linear fit and first order Fourier fit. The correlations were 0.65 and 0.68 respectively.

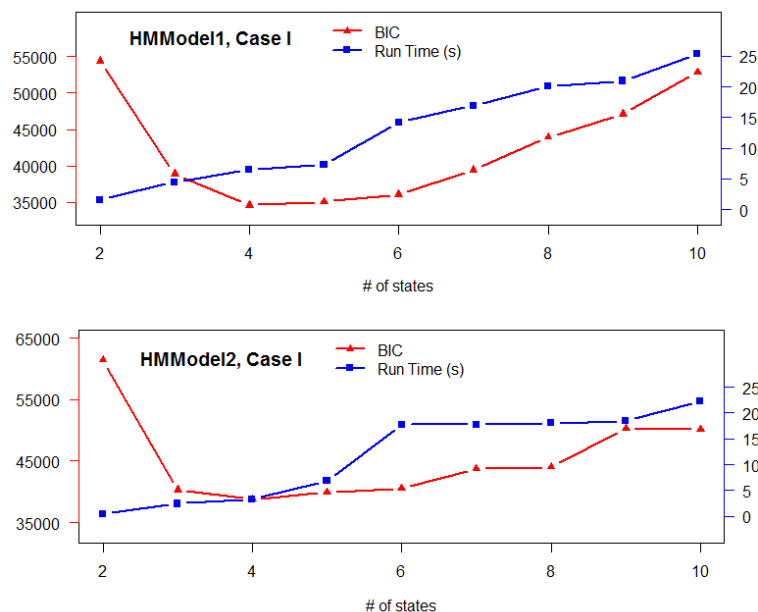


Figure 3.4. BIC and execution time vs number of states, Case I

A critical part to model a dataset with HMM is to select the number of states associated. Although often there is physical significance attached to the states, for the case of data center load, there is no straightforward significance that can be used in selecting the number of states. BIC criterion was used to find the optimal number of states. Simulations were performed varying the number of states from 2 through 10. Figure 3.4 shows the BIC values associated with each number of states. For both the models, taking 4 states leads to the lowest BIC score. Moreover, the first model, where the yearly seasonality was modeled as a linear fit, gives the lowest BIC. In this thesis, the analysis will be performed with 4 states for both the models.

Figure 3.5 shows the NREL data fitted with two models. It can be observed that both models were able to follow the average trend featuring high load in the mid months of the year and comparatively low loads towards the beginning and the end of the year.

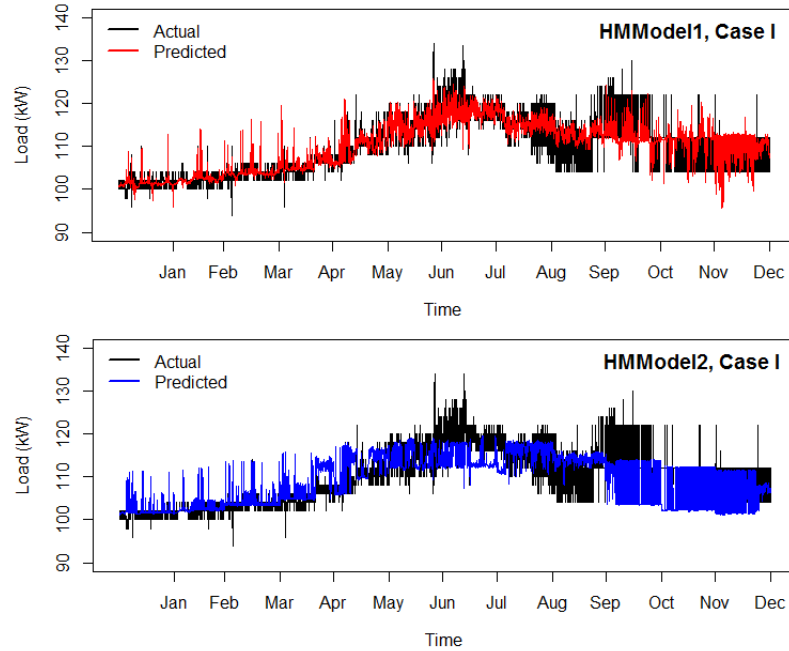


Figure 3.5. Fitted values from two models, Case I

The first model fits better than the second one as the mean estimates follow the actual trend more intently. From May to July, the second model is constantly under-forecasting. Also in October, the actual load is mostly populated above 110 kW, but the fitted values from model 2 is settled under 110 kW. In model 2, the monthly trend was modeled as Fourier term, which has lower values at both tails. Including the monthly Fourier as a linear covariate forced the estimates to take lower values. The goodness of fit of the models are reflected in their training MAPEs. The MAPEs were found to be 1.49% and 3.89% respectively for models 1 and 2.

3.2 Case Study II

In this case hourly load data from a Microsoft data center was used. The data is provided for a time period of one month. The resolution of the data is similar to that of

Case I, observations are made on hourly basis. There are $24 \times 30 = 720$ observations. Figure 3.6 shows the hourly load as a time series.

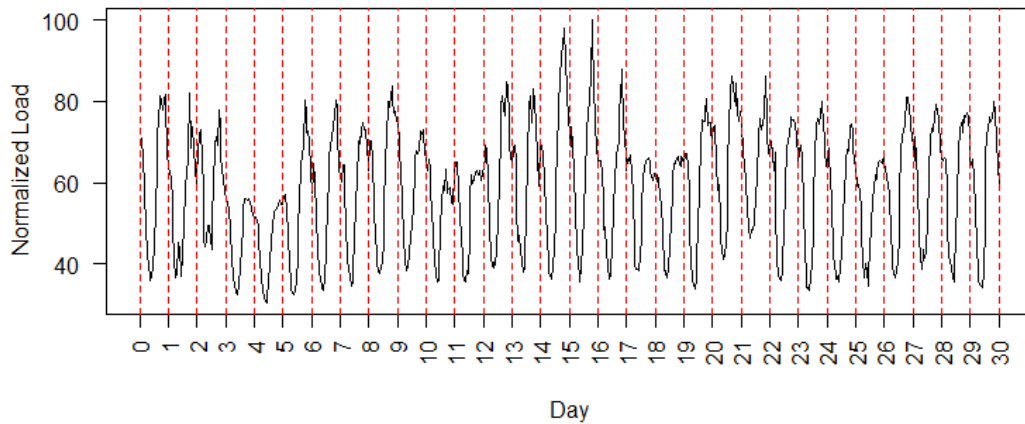


Figure 3.6. Normalized Load vs Time, Case II

Based on the data set, below are couple of observations associated with the dataset:

- From Figure 3.6, a daily seasonality can be observed. The first observation is assumed to be made at midnight of the starting day of the month. The load tends to slightly decrease at midnight and reaches the minimum point at around 8/9 am in the morning. Then, it starts to rise again. This rise continues up-to 5 pm. However, the rate of rise slowly decays over time. Until 10 pm, a significant variation is not observed. Then, it starts to decay. The decay continues up-to 9 am next morning. The rate of decay, again decreases slowly over time.
- Along with the daily cyclic pattern, a weekly seasonality is also observed. From Figure 3.6, there are smaller peaks for two consecutive days in every seven days. It was assumed that those two days represent the weekend. Data centers are used by mostly the corporations, along with personal usage. Naturally there will be

decreased load from corporations in the weekend. For those days, there is still a daily cyclic pattern, but reduced in magnitude.

Data preparation: The time variable possesses repetition. Every 24 observations have the values 1:24. So a hour variable was created.

Regarding the daily cyclic behavior, it was observed that for some period the load tends to increase and for some period it decreases. Within the increase period the rate of increase slowly rises and again shows some decay. After reaching the peak, the load starts to decrease. The decrease rate also rises over time and again decays before reaching the minimum. So the rate of change also exhibits a cyclic pattern, resembling a sinusoidal wave. In Figure 3.7, the load and a pure sinusoidal curve were plotted together. The phase shift of the curve was determined using the R package `harmonic.regression()`. The amplitude and bias were adjusted so that it matches the actual pattern. It is worth to note that this manual fixing of amplitude and bias does not affect the correlation of the sine curve and the load pattern. The correlation between the given time series and the first order Fourier expansion was found to be 0.855. Since this suggests a strong correlation, a Fourier variable was created to be used as a linear covariate.

3.2.1 Model selection

Since the hour variable was converted into a Fourier term, it would be redundant to use both hour and Fourier term in a model. The Fourier term was used in the model. To represent the weekday (and weekend), a binary variable was created, encoded with 1 and 0 representing weekend and weekday respectively. Two models were considered for this

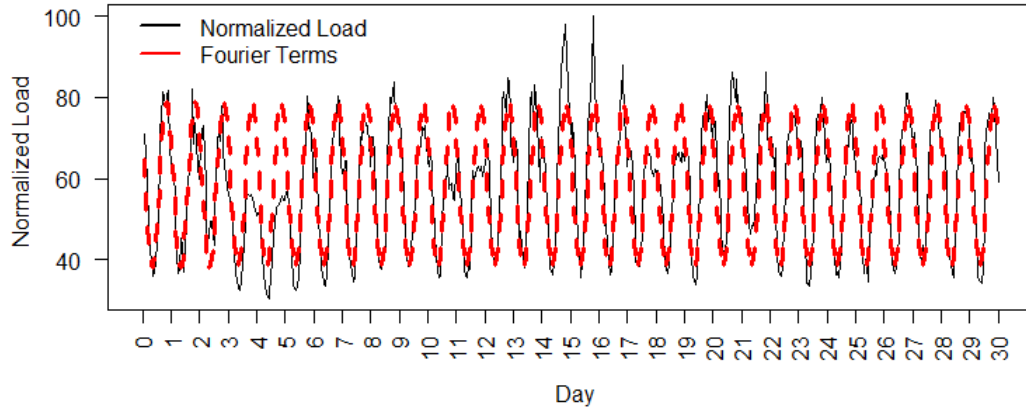


Figure 3.7. Load and pure sinusoidal curve, case II

case:

$$\text{Model 1 : Load} = a + b \times \text{fourier term} + c \times \text{weekend} + \varepsilon$$

$$\text{Model 2 : Load} = a + b \times \text{fourier term} + \varepsilon$$

In the first model, the weekend was incorporated as a linear covariate. It is expected to get a negative coefficient corresponding to this term as the average load is lower in the weekends. The second model is a simpler one, where just the hourly Fourier term was considered.

A hidden Markov model was applied with multiple number of states. The number of states were varied 2:10 like the previous case to find the optimal number. This is depicted in Figure 3.8. It is observed that taking the number of states=4 gives the lowest BIC score for both the models. So 4 states were taken into consideration for this case. Figure 3.9 shows the fitted values. The training MAPE values were found to be 5.04% and 6.05% respectively.

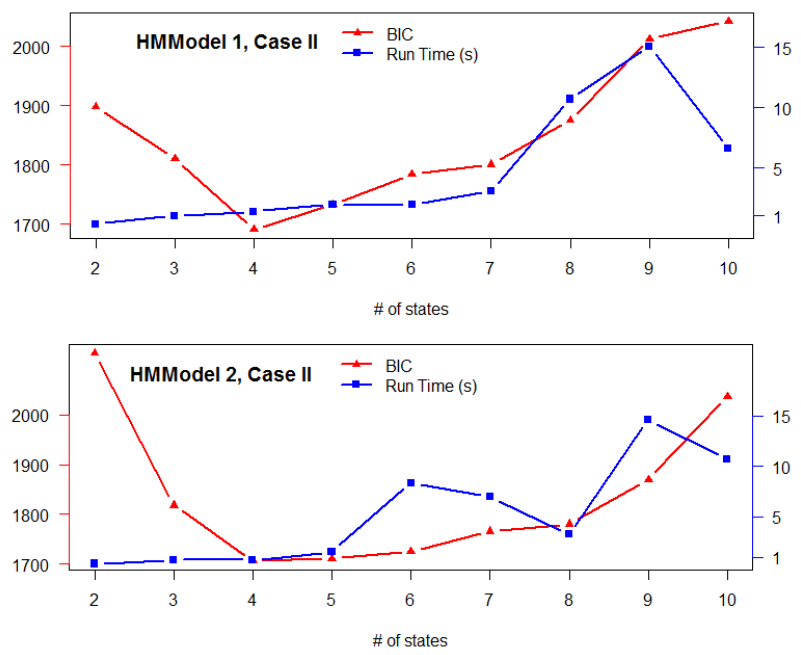


Figure 3.8. BIC and execution time vs number of states, Case II

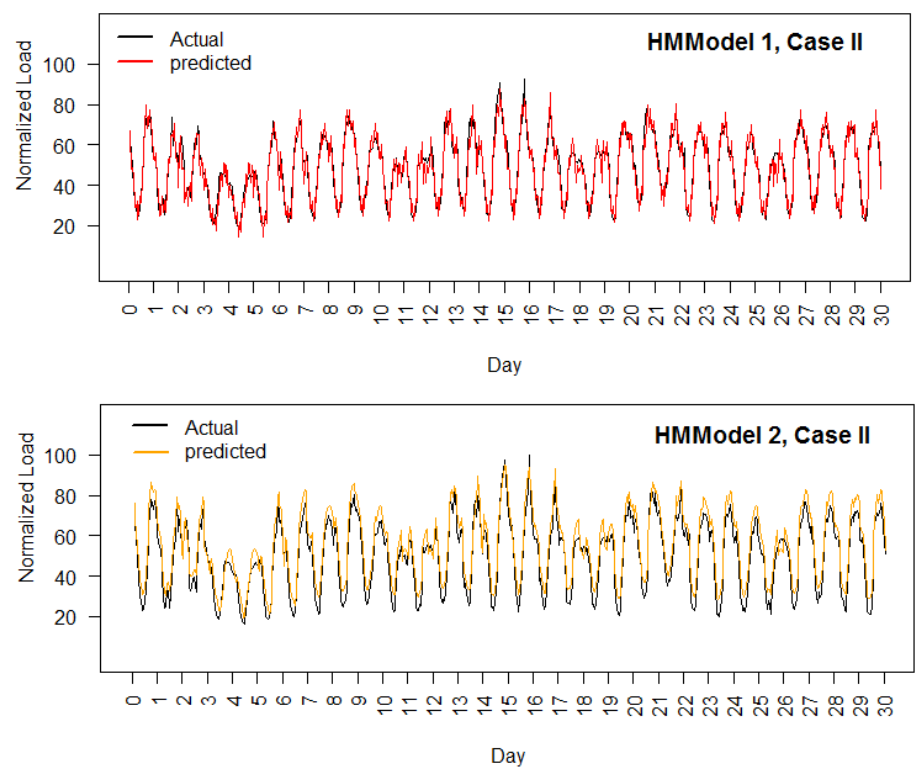


Figure 3.9. Fitted values from two models, Case II

3.3 Case Study III

In this case, the same dataset is used in the previous case. The only difference is that the data resolution is higher in this case with 1 observation in 2 minutes. Data provided for 1 month. Figure 3.10 presents the response variable.

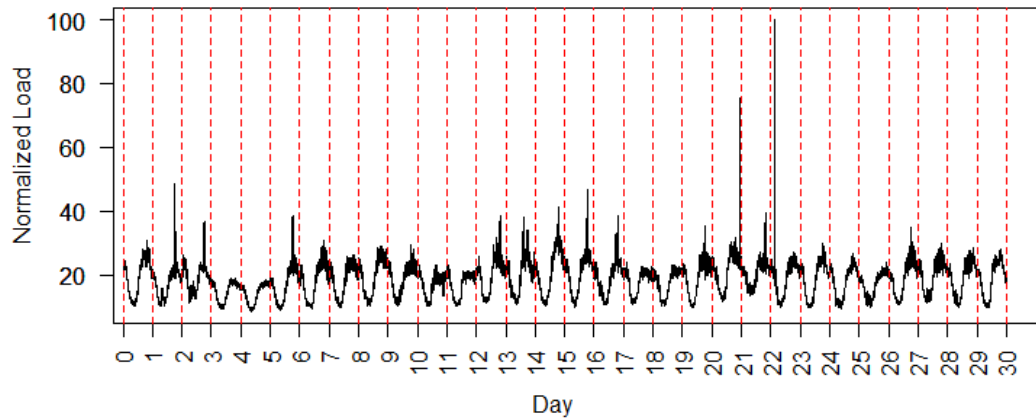


Figure 3.10. Normalized Load vs Time, Case III

Based on the case 3 load patten, below are the observations associated with the data set:

- From the plot in Figure 3.10, a daily cyclic pattern is observed like in the previous case. The same assumption of starting at midnight holds for this case as well.
- There are smaller peaks for consecutive two days in every seven days. Again, these days with smaller peaks are assumed to be weekends.

Before starting the model, a new variable was added in the dataset for simplicity. Due to the evidence of the cyclic pattern, again an hourly Fourier variable was added into the dataset. This is illustrated in Figure 3.11. The linear correlation of the Fourier variable and load was found to be 0.85.

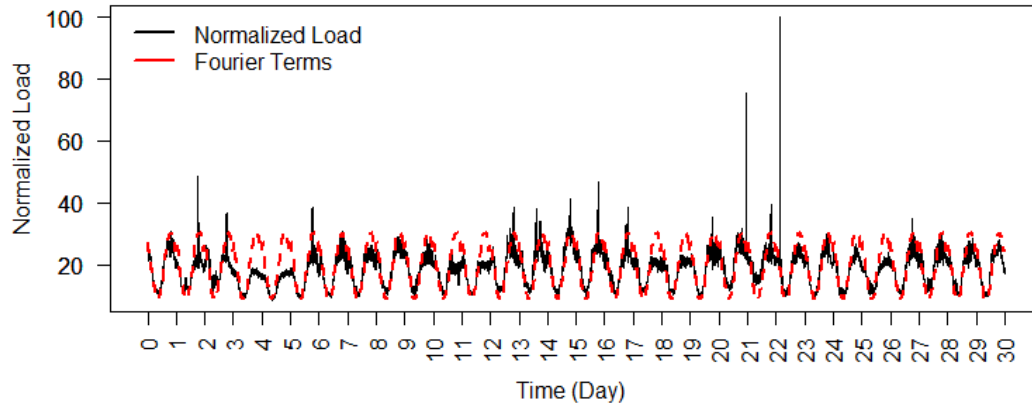


Figure 3.11. Load and pure sinusoidal curve, Case III

After adding the Fourier term, formal procedures to build the hidden Markov model is described. The succeeding subsections describe the methodology of model selection, assessment, and validation for this data set.

3.3.1 Model Selection

The hour variable is converted into a Fourier term, only Fourier term was included in the model to avoid redundancy like in the previous case. Apart from this, the minute variable was also included in the model. The weekends and weekdays were binary coded like before. Starting with a simple model, $Load = a + b \times minute + c \times fourier\ term + \epsilon$ HMM was applied to find the optimal number of states.

Figure 3.12 shows the BIC values against the number of states. Number of states was varied in the range 2:10. The BIC curve has a decreasing trend, suggesting better model with higher number of states. But it does not come free. It incurs the cost in terms of computational time. Along with the BIC, the execution time is also plotted in Figure 3.12. The execution time drastically increases if over 7 states. And the difference of BIC between state 6 and 7 is not very significant. So 6 was chosen as the number of states.

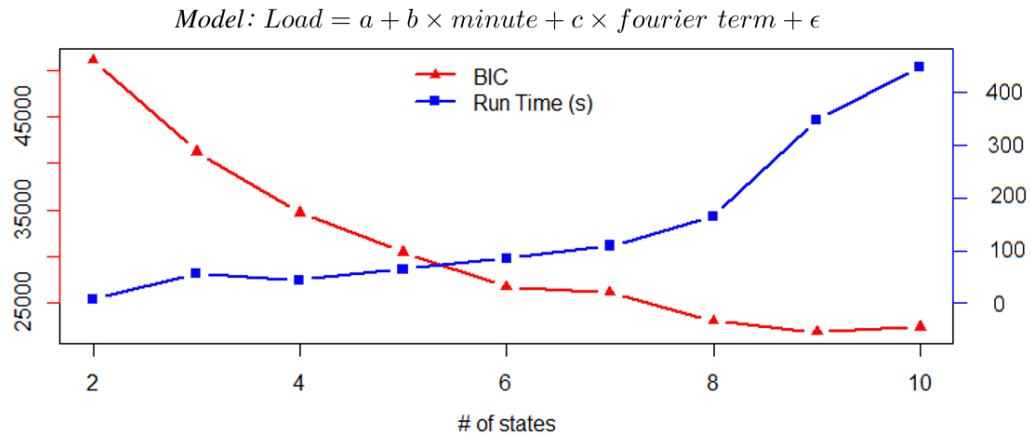


Figure 3.12. BIC and Execution time vs number of states of model 1, Case III

With 6 states, three models were considered:

$$\text{Model 1 : Load} = a + b \times \text{minute} + c \times \text{fourier term} + \varepsilon$$

$$\text{Model 2 : Load} = a + b \times \text{minute} + c \times \text{fourier term} + d \times \text{wk} + \varepsilon$$

$$\text{Model 3 : Load} = a + b \times \text{minute} + c \times \text{fourier term} + d \times \text{wk} + e \times \text{wk} \times \text{fourier} + \varepsilon$$

where, *wk* stands for weekday. The first model is straight forward, keeping just minute and the hour variable (mapped to Fourier terms). In the second model the weekday was included as a covariate since some weekend effect was observed. In the third model, an interaction of weekday and hour is also included.

Figure 3.13 shows the refitted normalized load. In the first model weekday was not incorporated. Model 1 failed to detect the reduced magnitude of the daily cycle for the first weekend. However, in the next three weeks, the reduced magnitude was detected for the weekends. The training MAPEs were 3.51%, 2.98%, and 3.13% respectively.

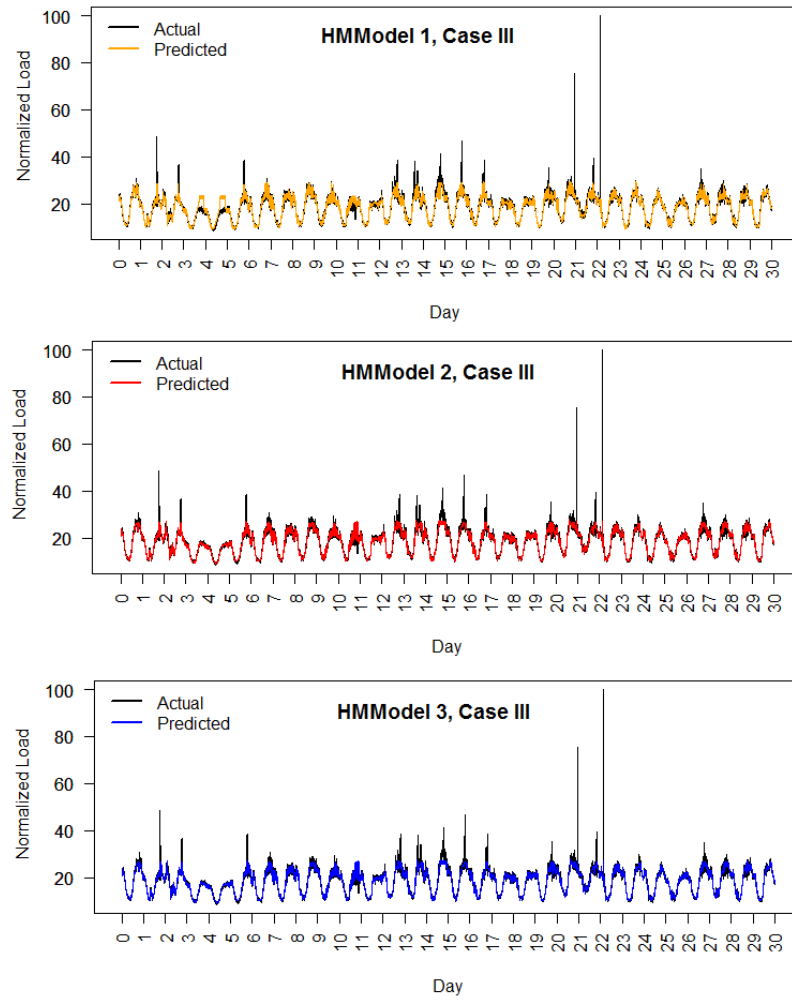


Figure 3.13. Fitted values from three models, Case III

CHAPTER 4 : LOAD FORECAST AND PERFORMANCE EVALUATION

4.1 Load Forecast

The ultimate goal of the models developed above is to predict the future load of the data center. The models were used to forecast loads for the next 24-hour period for the three cases.

4.1.1 Forecast: Case I

To make forecasts of d^{th} day, data up-to $(d - 1)$ days were used to train a model. After fitting the data, four (equal to the number of states) distinct functions were returned to find four estimates $\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4$ for the n^{th} hour. These estimates were aggregated by their respective weights $\mathbf{w}_n = [w_{n1}, w_{n2}, w_{n3}, w_{n4}]$.

To calculate the weight vector \mathbf{w}_n for the n^{th} forecast, initial state probability density and transition probability matrix were used. If $\mathbf{st}_{init} = [st_{init1}, st_{init2}, st_{init3}, st_{init4}]$ and $\mathbf{A} = a_{ij}$ denote the two quantities respectively, then $\sum_{i=1}^4 st_{initi} \times a_{ij}$ gives the probability of being at state j at the next hour. Setting $j = 1, 2, 3, 4$ gives probabilities for all the states. The new state probabilities can then be used to obtain state probabilities for the next time interval, inductively. Mathematically, this can be written as

$$\mathbf{w}_n = \mathbf{st}_{init} \times \mathbf{A}^n \quad (4.1)$$

For validation, thirty days were picked: the first five days of July and October, five days from the middle of August and November, and the last five days of September and

December. The forecast MAPEs from the two models are shown in Figure 4.1. Both models display similar patterns with a noticeable difference for the October forecasts. This can be explained with the help of Figure 3.1. In October the linear fit model closely matches the actual trend line, which made the second model worse than the first one for October. In the last two months both lines deviated from the actual trend which is reflected in the MAPE distribution. The mean forecast MAPEs were found to be 2.93% and 3.52% from two models.

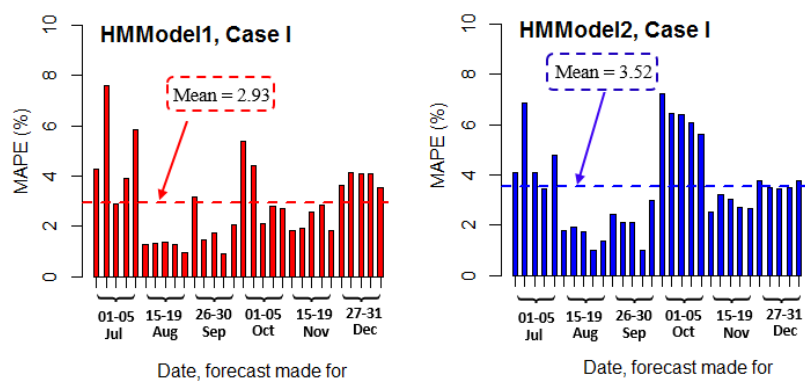


Figure 4.1. Forecast MAPE values, Case I

For graphical representation of the forecast, selected 3 days were randomly selected. Figure 4.2 shows the forecast of 15th Aug, 5th Oct and 19th November. Similar comment can be made for the model 2 forecasts as well, except for the constant under forecast of 5th October.

Both the models failed to pick up the non periodic downward spikes of November forecast. Observing the yearly data trend, it can be seen that the load approaches an almost constant value towards the end of the year with some random, equal, and non-periodic spikes, possibly from switching the heating load on and off. Because the model is learning from all the previous hourly loads, the training data becomes inadequate

for learning. Looking at Figure 3.5, these spikes are picked by the fitted values. So, it can be concluded that using more data to train the model would result in a better learning and better forecasting.

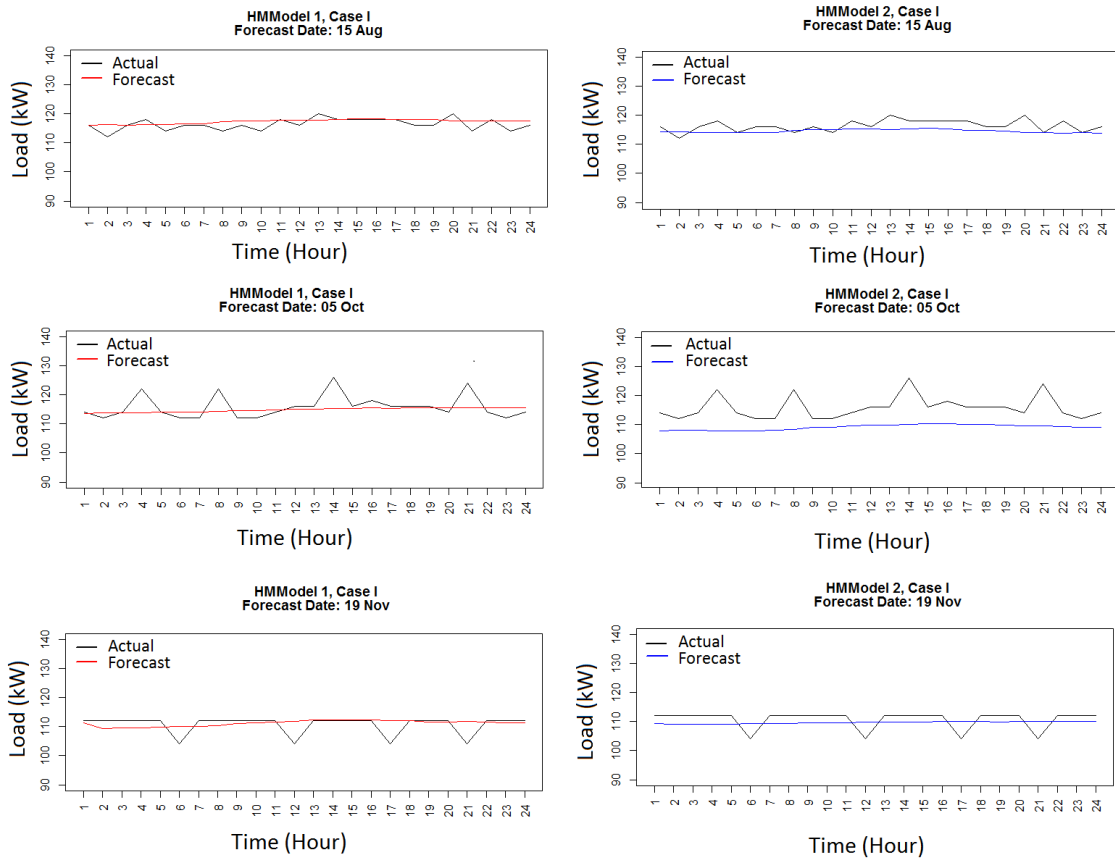


Figure 4.2. Load forecast, Case I

4.1.2 Forecast: Case II

In case II, exactly the same procedure was followed for forecasting like in the previous case. However, since the data window is different for this case, the choice of the validation days was also different. Out of the 30 days, the last 15 days were selected for validation purpose. The forecast MAPE values are presented in Figure 4.3. The simpler

model performed better than the complex model. The simpler model had significant higher training error than the complex one. But when it comes to the matter of forecast, the simpler model had lower MAPE. The mean forecast MAPE from model 1 was 15.10%, whereas model 2 had a mean forecast MAPE of just 9.40%. From Figure 4.3, in

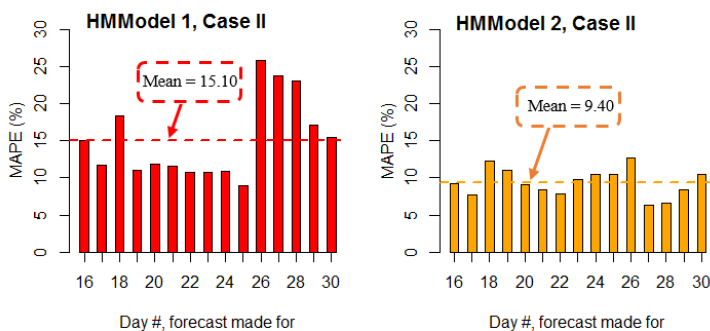


Figure 4.3. Forecast MAPE values, Case II

model 1, there is a jump on error at day 18, and again at day 26. Both of them are weekends. Model 1 incorporated the weekend as a variable. In training data, there are mostly four weekends, which might be insufficient for training. So using model 1 would be more viable if there were more data including more weekends.

For graphical representation, two days were randomly selected. Figure 4.4 shows the actual and forecast load for randomly chosen days (day 17 and day 23). Model 1 makes a day ahead forecasts with some spikes, which is wiped out in model 2. The average forecast trend from both the models follows the actual load pattern.

4.1.3 Forecast: Case III

Like in the previous two cases, similar method was used for load forecasting. Since for this case the resolution is different and there are 30 observations in one hour,

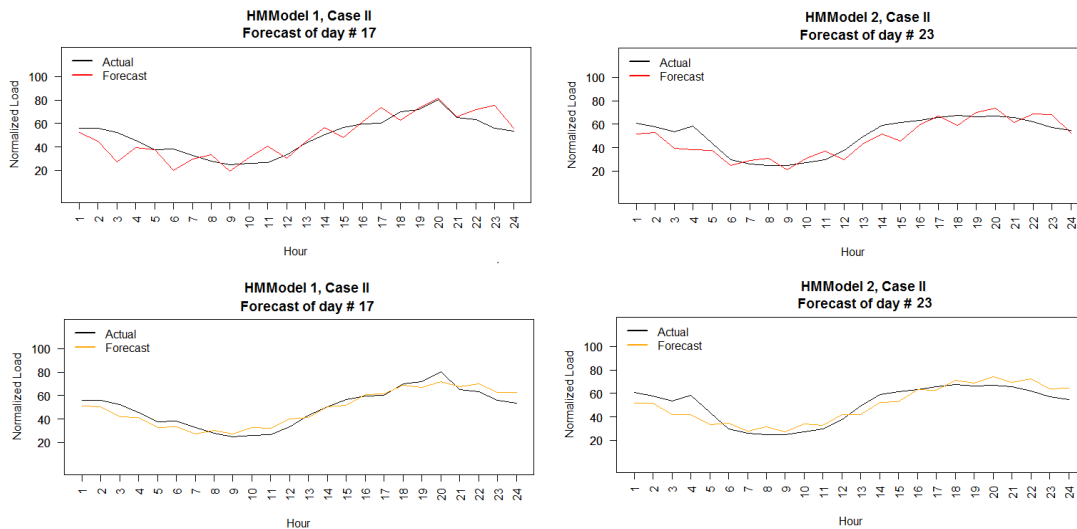


Figure 4.4. Load forecast, Case II

$30 \times 24 = 720$ future data had to be predicted for day ahead forecast. Like the previous cases, forecast had been done with the three models described above for the same 15 days of case II. The average forecast MAPEs were found to be 10.03%, 10.03%, and 9.67% respectively. Although the training error was higher than the second and the third, the forecast MAPE for all the models are almost consistent. Model 3 returned slightly lower error. All the MAPE distributions are shown in Figure 4.5.

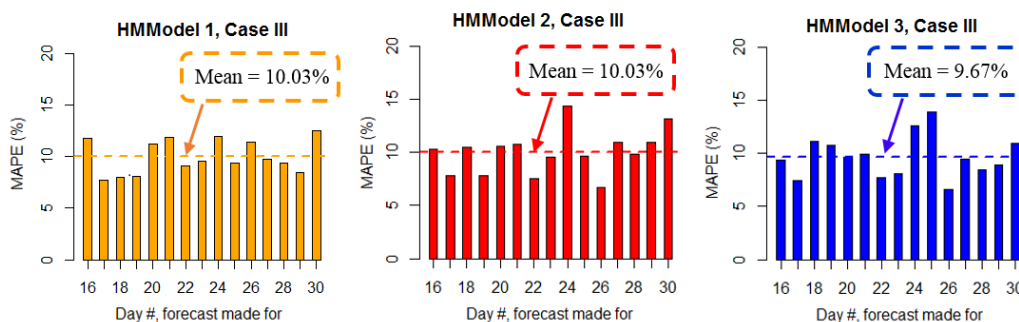


Figure 4.5. Forecast MAPE values, Case III

For graphical representation, again 2 days were randomly selected. These are shown in Figure 4.6. Even though the models did not capture all the minute by minute

spikes, but there is success in grabbing the average seasonality. The MAPE values were proved to be better than four other different methods, which is discussed in the next chapter.

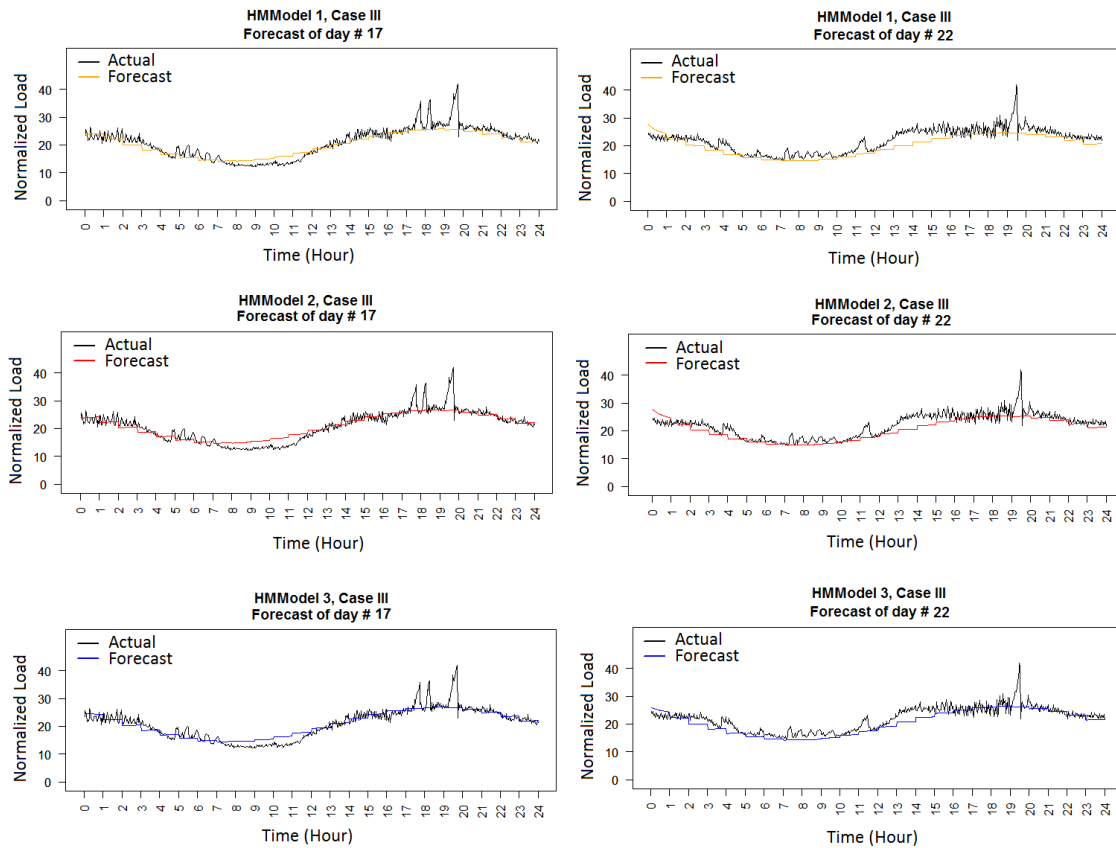


Figure 4.6. Load forecast, Case III

4.2 Forecast by other methods

In addition to the hidden Markov method, four other forecast methods were also implemented for comparison. Classical forecasting methods like simple moving average, exponential moving average, auto-regression integrated moving average and a hybrid of auto-regression and neural network described in [23] were used. A brief description of each of the methods is provided in the following subsections.

4.2.1 Simple Moving Average (SMA)

SMA is a basic, and widely used technique for analyzing data. It is arguably the most popular method used for trend detection [41]. It works as a finite impulse filter smoothing out the data by taking average of previous data. The mathematical expression to forecast y (response variable) is given by:

$$\hat{y}_{t+1} = \frac{y_t + y_{t-1} + \cdots + y_{t-m+1}}{m} \quad (4.2)$$

where m is the order of SMA and indicates how many previous terms are being used to take the mean. Selecting m is a crucial part of modeling a data set with SMA. There is a trade off choosing this too high or too low, which is analogous to bias variance trade off. Using fewer previous values to take the mean, it leads to lack of inertia and responds too heavily to the changes in data. On the other hand, selecting too high m overweights the inertia, resulting in a poor or no response at all to the data trend.

The training MAPE was used as the criterion of choice of m . The data were fitted with SMA using a range of m values and calculated the training MAPE for all of our cases. The results are tabulated in Table 4.1. Looking at the MAPE values, using $m = 1$ always leads to perfect data fitting for all the cases. SMA model with order 1 is equivalent to random walk model, where the forecast is simply equal to the last value. Results from random walk model shifted one step backward simply returns exactly the same values as in the original data. The reason of backward shifting is that the average age of SMA forecast is given by $(m + 1)/2$, that is whatever is being predicted, is what one would have

Table 4.1. Training MAPE, SMA model

<i>m</i> value	Training MAPE (%)		
	Case I	Case II	Case III
1	0.00	0.00	0.00
3	1.21	2.00	1.97
5	1.33	3.79	2.10
7	1.38	5.77	2.33
9	1.41	7.74	2.61
11	1.44	9.79	2.63
13	1.47	11.88	2.82
15	1.51	13.98	2.90
17	1.55	16.04	3.00
19	1.59	18.03	3.11
21	1.62	19.86	3.13
23	1.66	21.47	3.20
25	1.69	22.87	3.23

expected $(m + 1)/2$ step earlier.

Looking at Table 4.1, it can be seen that taking three terms for averaging out the data gives the minimum training error for all the cases. So $m = 3$ was used for all the cases.

4.2.2 Simple Exponential Moving Average (SEMA)

The idea of SEMA is similar to that of SMA, with an exception on the weights put on the previous values to take the average. In SMA, all the previous terms are equally weighted, while in SEMA, more weight is given to the more recent value to discount the past data in more gradual fashion. In the simplest form, the mathematical expression is given by:

$$\hat{y}_{t+1} = w_1 \times y_t + w_2 \times y_{t-1} + \cdots + w_m \times y_{t-m+1} \quad (4.3a)$$

$$\sum_{i=1}^m w_i = 1 \quad (4.3b)$$

Using $m = 3$ resulted the weights for three terms to be 0.65, 0.25 and 0.1. These values were determined by integrating the exponential curve e^{-x} , covering 94% region, over three uniform time intervals $[0, 1]$, $[1, 2]$, and $[2, 3]$. This is depicted in Figure 4.7. Initially the weights were found to be 0.63, 0.23 and 0.08. To meet the stochastic criterion as described in equation (4.3b), 0.02 was added with all the weights.

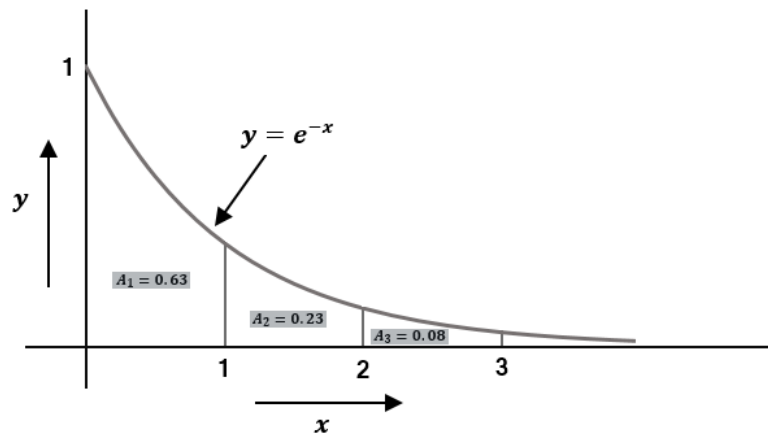


Figure 4.7. Determining weights, SEMA model

The average age for this case is determined by calculating the value, which divides the area under the exponential curve equally. Since $\int_0^{\infty} e^{-x} dx = 1$, the average age can be calculated intuitively by solving $\int_0^x e^{-x} dx = 0.5$ for x . For the third order SEMA model, the average age was found to be $\log_e 2 = 0.69$.

4.2.3 Auto Regression Integrated Moving Average (ARIMA)

Autoregressive Integrated Moving Average (ARIMA): ARIMA is time series modeling tool described by two polynomials, one for auto-regression (AR) and other one for moving average (MA). The AR process specifies the response as a linear combination

Table 4.2. $ARIMA_{(p,d,q)}$ model order for three cases

	p	d	q
Case I	1	1	1
Case II	2	0	2
Case III	3	1	4

of its own previous values as specified by equation (4.4):

$$\hat{y}_{t+1} = c + \sum_{i=1}^p \varphi_i y_{t-i+1} + \varepsilon \quad (4.4)$$

where p is the order of AR process. The generalized ARIMA is described as of order (p, d, q) , where they refer to the order of AR, differencing and MA part. The ARIMA model was implemented for three cases using the R function `auto.arima()` which chooses the best (p, d, q) based on BIC. For the three cases, the ARIMA orders were found to be $(1, 1, 1)$, $(2, 0, 2)$ and $(3, 1, 4)$ respectively. This is shown in tabular format in Table 4.2.

4.2.4 Auto Regression Neural Network (ARNN)

This method is described in [23]. This is a blend of AR and Neural Network (NN). This model basically is a $3 \times 2 \times 1$ NN architecture, which had been applied to forecast the load by training in a supervised manner with back propagation algorithm. The neural model receives own three previous data as input and links the output response through the hidden layer. Figure 4.8 provides the synopsis of the ARNN model.

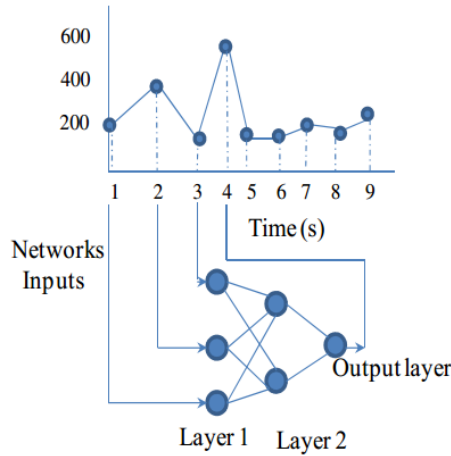


Figure 4.8. Synopsis of ARNN model [23]

4.3 Results Comparison

For case I, 30 days were selected for forecast. For the other two cases, last 15 days of the month were selected. Forecast MAPEs are tabulated in Table 4.3, 4.4, and 4.5.

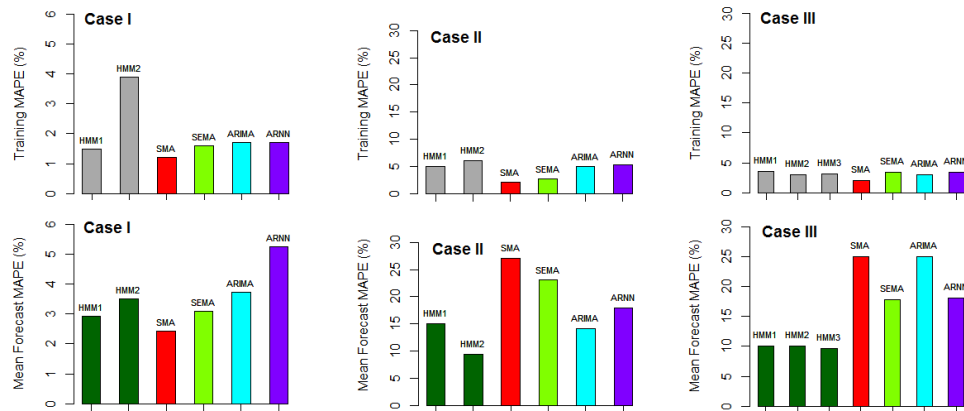


Figure 4.9. Models comparison

The training MAPEs and mean forecast MAPEs from different models are tabulated in TABLE 4.6, 4.7, and 4.8 for three cases respectively. The forecast summaries are graphically presented in Figure 4.9. For case I, where the NREL data was modeled, the ARIMA model performed better than all other models including the proposed two

Table 4.3. Forecast Evaluation, Case I

Forecast Date	Forecast MAPE (%)					
	HMM 1	HMM 2	SMA	SEMA	ARIMA	ARNN
01-Jul	4.28	4.09	2.10	2.80	2.86	9.65
02-Jul	7.62	6.88	2.75	3.65	4.44	12.51
03-Jul	2.89	4.10	2.97	3.52	4.10	10.58
04-Jul	3.93	3.44	3.22	3.31	5.31	10.67
05-Jul	5.84	4.79	2.74	3.09	3.21	11.49
15-Aug	1.27	1.78	1.04	2.34	2.49	6.42
16-Aug	1.34	1.92	1.45	2.21	3.15	5.92
17-Aug	1.36	1.73	1.51	2.41	3.48	5.88
18-Aug	1.26	1.01	1.46	1.94	1.59	4.85
19-Aug	0.93	1.36	1.51	1.10	2.76	5.01
26-Sep	3.17	2.42	1.50	2.50	3.99	4.41
27-Sep	1.47	2.10	2.34	2.40	3.31	3.98
28-Sep	1.75	2.09	1.91	2.60	2.81	4.38
29-Sep	0.90	0.98	1.77	1.51	2.46	2.23
30-Sep	2.08	3.00	2.80	2.69	3.07	4.19
01-Oct	5.39	7.22	2.35	3.10	4.47	4.59
02-Oct	4.44	6.45	2.35	5.32	4.71	4.77
03-Oct	2.09	6.40	2.46	3.15	3.04	5.44
04-Oct	2.80	6.06	2.19	3.07	3.44	6.18
05-Oct	2.69	5.62	3.21	3.04	3.81	5.06
15-Nov	1.83	2.51	2.24	1.63	3.65	2.38
16-Nov	1.92	3.23	2.16	1.72	3.61	2.56
17-Nov	2.59	3.01	2.17	4.94	2.59	2.38
18-Nov	2.83	2.71	2.25	3.47	5.14	2.56
19-Nov	1.85	2.66	2.38	4.60	4.34	2.38
27-Dec	3.62	3.78	3.54	4.23	4.28	3.42
28-Dec	4.13	3.50	3.57	4.40	5.16	3.43
29-Dec	4.10	3.47	3.56	4.39	4.29	3.25
30-Dec	4.10	3.50	3.52	3.52	4.74	3.43
31-Dec	3.56	3.75	3.57	3.91	5.36	3.61

Table 4.4. Forecast Evaluation, Case II

Day	Forecast MAPE (%)					
	HMM 1	HMM 2	SMA	SEMA	ARIMA	ARNN
16	15.11	9.26	24.47	33.73	13.94	19.19
17	11.73	7.67	24.93	29.33	12.01	18.49
18	18.36	12.26	21.48	22.48	9.03	12.81
19	10.98	11.09	16.21	19.38	15.74	14.10
20	11.81	9.15	19.31	27.47	22.18	21.09
21	11.61	8.43	27.63	23.28	15.79	18.14
22	10.77	7.88	21.80	23.15	10.79	14.91
23	10.71	9.84	16.18	25.80	14.25	17.69
24	10.95	10.48	24.13	30.96	13.46	21.82
25	8.94	10.48	28.42	28.31	11.32	18.10
26	25.91	12.65	26.18	28.13	13.74	15.51
27	23.83	6.32	23.25	26.60	20.81	20.31
28	23.16	6.66	25.77	25.62	11.50	16.85
29	17.13	8.39	22.13	28.60	13.49	19.20
30	15.48	10.51	24.02	34.06	14.71	20.93

Table 4.5. Forecast Evaluation, Case III

Day	Forecast MAPE (%)						
	HMM 1	HMM 2	HMM 3	SMA	SEMA	ARIMA	ARNN
16	11.80	10.33	9.37	27.42	21.09	28.19	19.74
17	7.77	7.79	7.48	25.19	20.49	24.44	18.97
18	8.02	10.52	11.19	17.36	18.53	17.04	13.11
19	8.07	7.78	10.75	19.11	13.36	18.92	14.61
20	11.20	10.56	9.63	28.01	13.08	27.42	20.03
21	11.86	10.77	9.97	23.17	20.63	23.95	18.15
22	9.07	7.57	7.71	22.07	18.85	22.45	16.19
23	9.59	9.57	8.11	25.33	12.13	24.93	18.23
24	12.00	14.37	12.64	29.40	18.12	29.42	21.22
25	9.36	9.64	13.91	28.26	22.29	29.23	18.26
26	11.45	6.67	6.61	23.50	19.82	23.83	16.12
27	9.75	10.96	9.45	26.40	15.17	26.14	20.41
28	9.38	9.86	8.43	24.83	19.12	24.87	17.47
29	8.50	10.92	8.92	26.92	16.76	26.21	18.65
30	12.57	13.21	10.92	28.55	18.67	28.75	21.18

Table 4.6. Model summary, Case I

	HMM 1	HMM 2	SMA	SEMA	ARIMA	ARNN
Training MAPE (%)	1.49	3.89	1.21	1.59	1.72	1.70
Mean Forecast MAPE (%)	2.93	3.52	2.42	3.09	3.72	5.25

Table 4.7. Model summary, Case II

	HMM 1	HMM 2	SMA	SEMA	ARIMA	ARNN
Training MAPE (%)	5.04	6.05	2.00	2.65	4.92	5.31
Mean Forecast MAPE (%)	15.10	9.40	27.13	23.06	14.19	17.94

Table 4.8. Model summary, Case III

	HMM 1	HMM 2	HMM 3	SMA	SEMA	ARIMA	ARNN
Training MAPE (%)	3.51	2.98	3.13	1.97	3.44	2.99	3.39
Mean Forecast MAPE (%)	10.03	10.03	9.67	25.03	17.87	25.05	18.16

hidden Markov models, in terms of both training and forecast. However, one of our proposed two models, where the month was used as a linear covariate, performed better than all other models except for the ARIMA in both training and forecasting. In other two cases, our proposed models lacked in performance while training the data. The other models described in the preceding subsection performed better than the proposed hidden Markov models in most cases. However, looking at forecast MAPEs demonstrates the superior performance of the proposed hidden Markov models over the other methods. All the different hidden Markov models returned lower MAPE values than the other methods. Considering the best one from different proposed hidden Markov models for all the cases, there are $3 \times 4 = 12$ comparisons. On 11 occasions out of these 12, HMM was proved better forecast method. To find a global metric for comparison, a reduction index was formulated as I_r , which is given by:

$$I_r = \frac{\sum_{j=1}^3 \sum_{k=1}^4 [m_{kj} - \min(m_{ij})]}{\sum_{j=1}^3 \sum_{k=1}^4 1} \quad (4.5)$$

where m indicates the mean forecast MAPE. Index k denotes one of the other methods implemented for comparison, j denotes the three cases and i denotes the several hidden Markov models developed for corresponding cases. The reduction index indicates how well the proposed model performed on average. Using equation (4.5), I_r was found to be 7.91. that is, on average the proposed model resulted 7.91% lower forecast MAPE than the other four methods.

4.4 Economic Impact

A good load forecast can be significant for the operational cost of a datacenter. Inaccurate forecast leads to sub optimal set points and incurs penalty. In this section, the penalty associated with inaccurate forecast will be investigated. For this, a 100 MW grid connected datacenter with wind power, solar power, battery reserve and on site generator was considered. Optimal set points were solved for the actual load (perfect forecast) and forecasted load. Set points for forecasted load were used to operate the datacenter for actual load. Optimal cost was subtracted from the actual operational cost to find the penalty.

Based on the [10], a 100 MW datacenter benchmark was developed, including 40 MW peak solar power, 20 MW peak wind power, 192 MWhr AGM PbA battery, 10 MW on site natural gas generator and and grid power. The hourly data from second case was used as the load. It was scaled to have a peak of 100 MW load. Figure 4.10 shows the solar and wind power profile for a typical day. For simplicity, these were considered as free energy. The wear costs associated with these renewables were not included in the operational cost.

The cost of grid power varies throughout the day. Figure 4.11 shows the locational marginal price of a typical summer day, which was collected from PJM. The generator cost was adapted from [VPP]. It composes of startup cost and running cost and governed by the following equation:

$$C_{gen}(P_{gen}) = 0.07 \times P_{gen}^2 + 20.2 \times P_{gen} + 7.27 \quad (4.6)$$

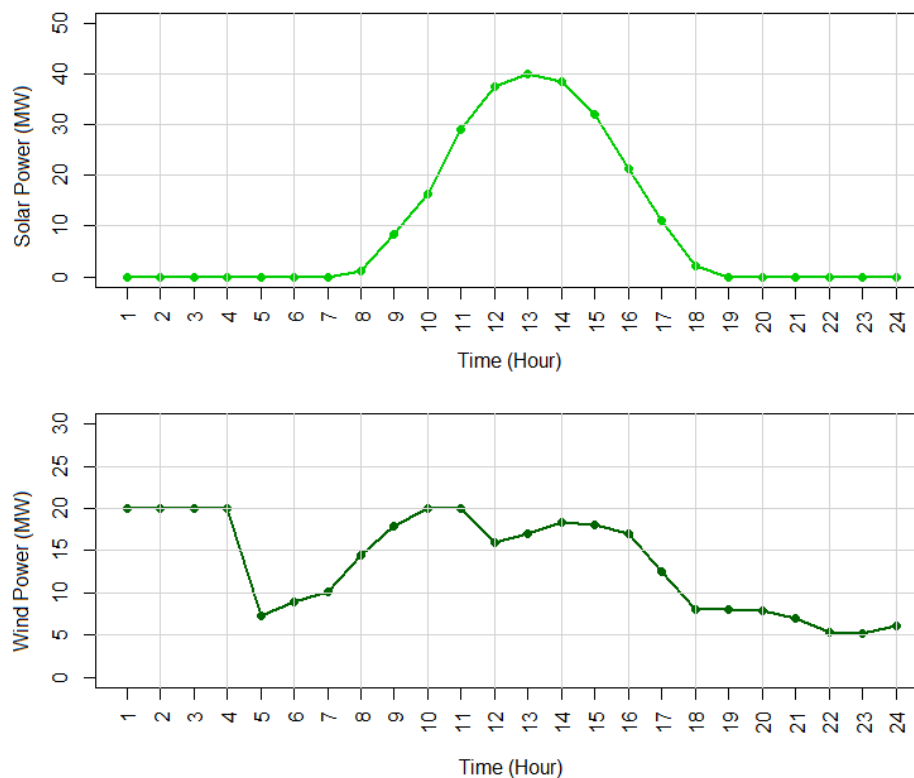


Figure 4.10. Solar and Wind power profile for a typical day

where, C_{gen} is given in $\$/hour$ and P_{gen} is given in MW. The generator cost is plotted in Figure 4.12. Along with the actual plot, a linear fit was also plotted dropping the quadratic term of the cost function. Being the quadratic coefficient small compared to the linear coefficient and the constant term, two curves almost overlap. In the simulation, the simpler linear cost model was considered.

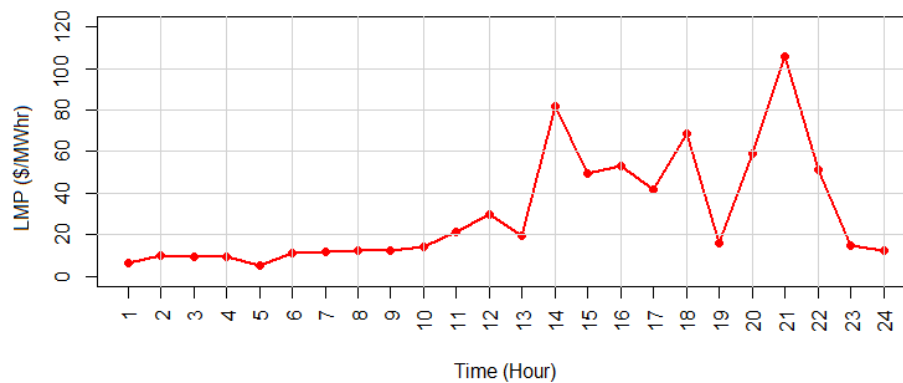


Figure 4.11. Locational Marginal Price (LMP) for a typical day

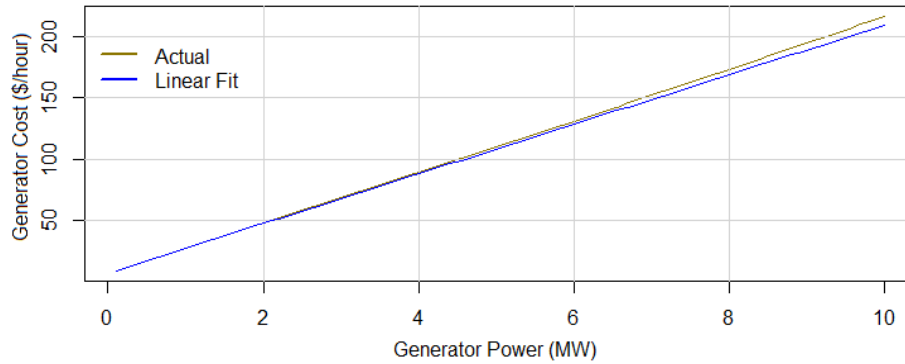


Figure 4.12. Natural gas generator cost curve

Generator efficiency is also a function of output power. High efficiency is achieved with higher loading. Figure 4.13 shows the efficiency of the generator as a function of percentage loading of rated power. Below 40% loading, generator operates in low efficiency and shows a greater degree of variability. Above 50% loading, there is still a trend, but with lower variability. For high efficiency, the on-site generator was constrained to operate at least at 50% of its maximum efficiency.

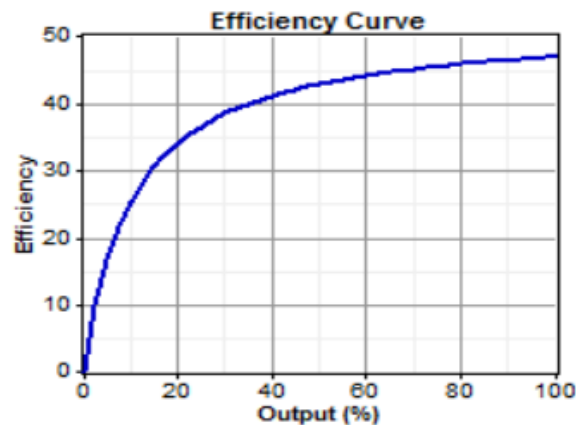


Figure 4.13. Natural gas generator efficiency curve [10]

The battery wear cost was taken as $\$0.506/MWhr$ while discharging, considering AGM PbA battery [42]. Two constraints were imposed while charging/discharging power from battery bank. First, the charging/discharging rate should not exceed $45MW/hr$ and

battery state of charge (SoC) remains within [0.5 1] [43].

With the system benchmark described above, an optimized day ahead schedule of the grid power, battery power and the on-site generation was calculated to minimize the cost. The optimization objective is formulated as:

$$\text{Minimize } Cost_{total} = \sum_{i=1}^{24} (P_{grid,i} \times LMP_i + P_{gen,i} \times C_{gen_i} + P_{bat,i} \times C_{batwear,i}) \quad (4.7)$$

subject to,

$$P_{solar,i} + P_{wind,i} + P_{grid,i} - P_{bat,i} + P_{gen,i} - P_{load,i} = 0 \quad (4.8a)$$

$$\sum_{i=1}^{24} [P_{batcharging,i} \times \eta + P_{batdischarging,i} / \eta] = 0 \quad (4.8b)$$

$$0.5 \leq SoC_{bat,i} \leq 1 \quad (4.8c)$$

$$-45 \text{ MW} \leq P_{bat,i} \leq 45 \text{ MW} \quad (4.8d)$$

$$P_{gen,i} = 0 \text{ or } 5 \text{ MW} \leq P_{gen,i} \leq 10 \text{ MW} \quad (4.8e)$$

$$P_{grid,i} \geq 0 \quad (4.8f)$$

The first constraint (Equation (4.8a)) ensures the load balance; total power generation would equal the total load. The negative sign of the battery power means battery power is treated positive when charging, and negative when discharging. The next three constraints (Equation (4.8b)-(4.8d)) relate to the operation of battery and ensure that the battery would come back to initial reserve level and SoC and charging/discharging rate remain

within the defined limits. The fourth constraint (Equation (4.8e)) secures at least 50% loading of the generator, otherwise it would not run. Finally, the last constraint (Equation (4.8f)) guarantees one-way power flow from the grid. It was assumed that no power can be sold to the grid operator.

First the schedules of the generator, battery and the grid power was made using deterministic optimization method. The actual load of day 16 was used for the scheduling. This is illustrated in Figure 4.14. Based on this schedules, the optimal cost was found to be \$22,036.62. However, in the real scenario, the load profile will be different since forecasted load will differ from the actual load. This would make a shift of the set points, as shown in Figure 4.15. Black bars represent the dispatch schedules for the HMM forecast load. However, although these set points are optimal for the forecast load, this is not optimal for the actual load. While dispatching in real time, the operator would require to continuously adjust the power from resources to meet the load balance resulting from under-forecast or over-forecast.

As stated earlier, in real time dispatching, the residual load must be absorbed. While doing so, care must be taken so that the constraints used in the optimization must satisfy. This is critical as there are minimum and maximum limits for the generator. Also there are limits for battery SoC, governed by complex equations. So while dispatching, the battery set points were not perturbed in this case. The actual and forecast mismatch would be compensated by the generator and grid only. Two kinds of load mismatches could can be emerged. One from under-forecasting and another one from over-forecasting. In this work, the dispatching algorithm relied on these two cases, which is shown in Figure 4.16. Using the algorithm, the operational cost was found to be

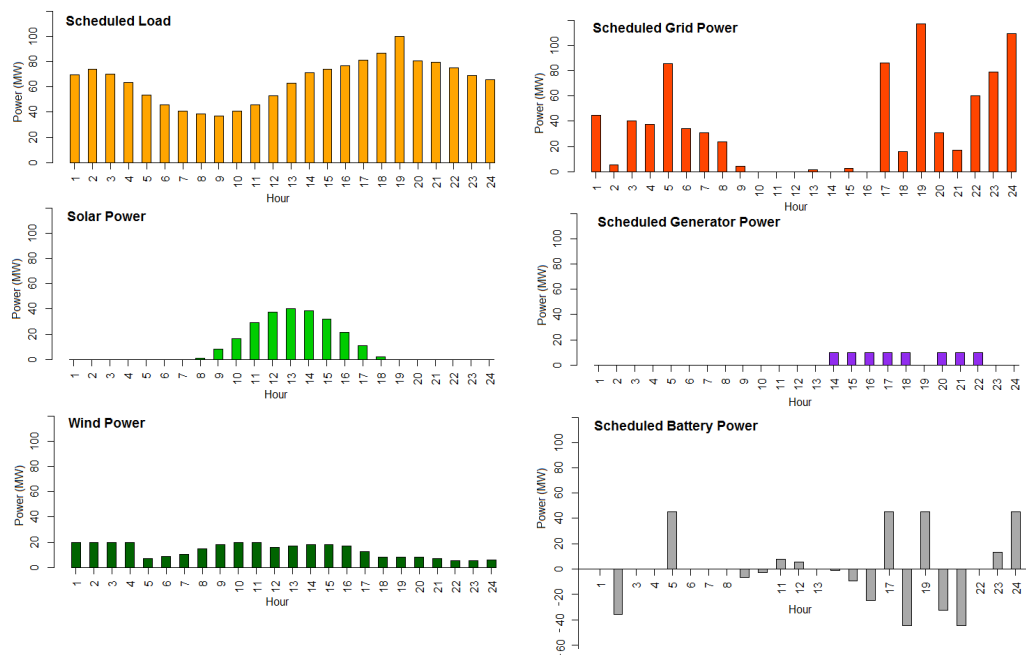


Figure 4.14. Profiles for actual load

Table 4.9. Penalty for inaccurate forecast (\$)

	HMM	SMA	SEMA	ARIMA	ARNN
Day 16	590.21	1092.00	928.86	1033.01	974.73
Day 20	655.18	1243.25	960.65	1337.64	967.74
Day 22	394.52	629.42	1168.10	1179.14	798.05
Day 27	450.06	1736.39	1364.79	1221.71	942.63
Day 28	409.86	816.58	944.95	912.78	931.24
Mean	499.97	1103.53	1073.47	1136.86	922.88

\$22,626.83. The difference is \$590.21, which is the penalty for inaccurate forecast.

Four days were randomly picked to find the operational cost penalty.

Optimizations were performed with forecast loads from all methods discussed previously.

Table 4.9 shows the results, and graphically presented in Figure 4.17.

From Table 4.9 it can be seen that non-perfect forecast always leads some penalty, resulting from sub-optimal set points of scheduling. However, penalty can be reduced by better forecast. For forecasting in Case II, it was found that HMM performed better than

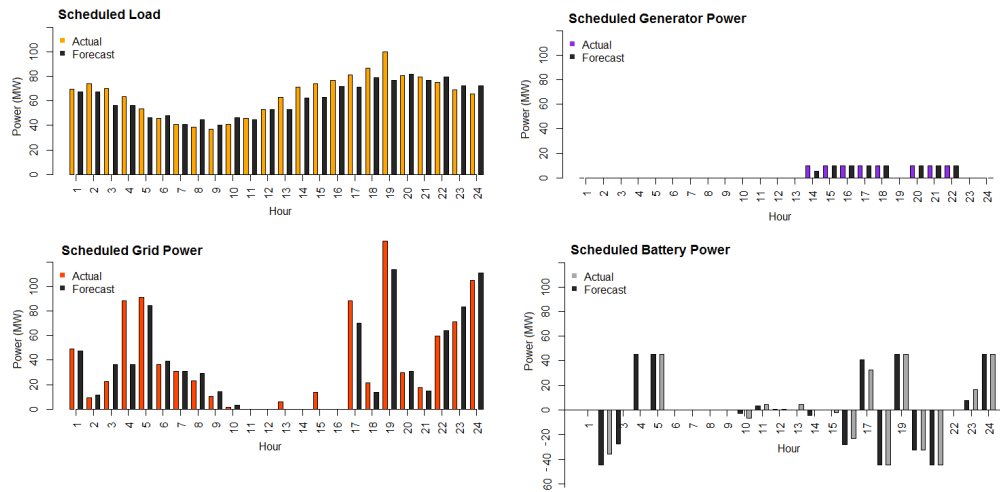


Figure 4.15. Profiles for actual and forecast load

the other forecast methods. This is reflected in the penalty cost. For each of the selected days, the penalty from HMM forecast scheduling was lower than the penalties from other forecast scheduling. The mean penalty for HMM forecast scheduling was found to be \$499.97. The mean penalties from SMA, SEMA, ARIMA and ARNN forecast based scheduling were \$1,103.53, \$1,073.47, \$1,136.86 and \$922.88, 121%, 115%, 127%, and 85% higher than the mean penalty from HMM forecast based scheduling respectively.

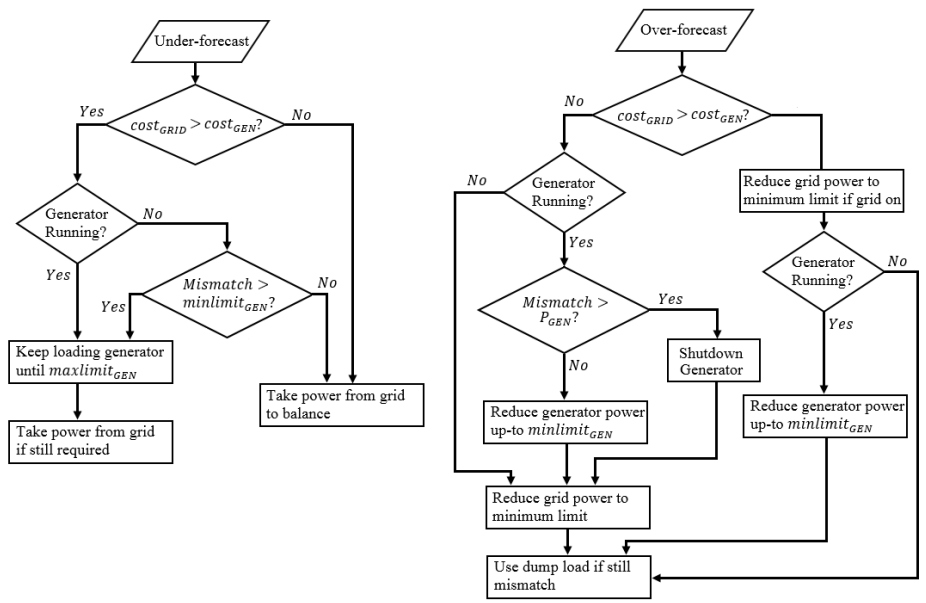


Figure 4.16. Flowchart of dispatching

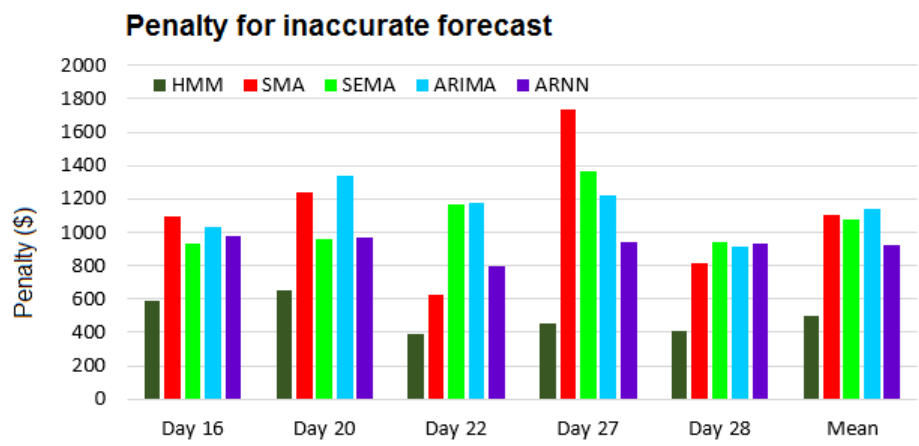


Figure 4.17. Penalties for inaccurate forecast

CHAPTER 5 : CONCLUSION

With the increase of dependency on cloud computing the cost involved in datacenters operation is also increasing. Industries are coming up with new ideas and technologies to optimize the cost by reducing the energy used by datacenters. Load forecasting plays a vital role in both efficient scheduling and operating a data center as a virtual power plant. In this thesis work a stochastic method based on dependent mixture and Bayesian inference is developed to model the datacenter load and is used for day-ahead forecast. The model was used to predict the unforeseen day-ahead load and was cross validated using three data sets from National Renewable Energy Laboratory (NREL) and one another data center. The proposed method was proved better than classical auto-regression, moving average and neural network based forecasting systems and resulted on average reduction of 7.91% forecast MAPE. This leads to a yearly savings of \$13,705 for a typical 100 MW coal fired tier-IV data center [44].

Datacenter load forecast can have economic impact. The variability and uncertainty of load affects the unit commitment decisions of the generators. A good forecast can also help the utilities and regulatory commissions for integrated resource planning to meet the energy requirement cost-effectively. This was investigated by setting up a 100 MW datacenter benchmark. It was found that inaccurate forecast always leads to sub-optimal set points for resource scheduling, which further leads to operational penalty. However, improved accuracy tends to decrease the cost of penalty.

Currently datacenters are incorporating such as PV and wind by either having own sources or buying from existing off-site generation to reduce the carbon footprint. This

potentially widened the scope of Demand Response (DR) program, which reduces the electricity usage during peak periods. Prediction of load is one of the preliminary tasks to schedule load shifting. It has been shown that there is a potential to reduce the peak electricity by as much as 20% in the US through DR [45].

An accurate load forecast can help strategic scheduling, optimal unit commitment decision and DR. These approaches reduce not only the cost, but also the carbon footprint, from which the society can greatly be benefited.

REFERENCES

- [1] A. Rallo, *Industrial outlook: Data center energy efficiency*, www.datacenterjournal.com/, published: Nov 15, 2014.
- [2] Cisco, *Cisco global cloud index: Forecast and methodology, 2014–2019 white paper*, www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html, Updated: Apr 21, 2016.
- [3] S. Shead, *Google announces plans to open more data centers*, www.inc.com/business-insider/google-revamping-cloud-platform-data-centers.html, published: Mar 23, 2016.
- [4] J Whitney and P Delforge, “Data center efficiency assessment,” *Natural Resources Defense Council, New York City, New York*, 2014.
- [5] N. Deng, C. Stewart, and J. Li, “Concentrating renewable energy in grid-tied datacenters,” in *Sustainable Systems and Technology (ISSST), 2011 IEEE International Symposium on*, IEEE, 2011, pp. 1–6.
- [6] *Five strategies for cutting data center energy costs through enhanced cooling efficiency*, <http://www.emersonnetworkpower.com/>, A White Paper from the Experts in Business-Critical Continuity, published by Emerson Network Power, 2007.
- [7] Anthesis, *Data center efficiency assessment*, <https://www.nrdc.org/sites/>, published by Natural Resource Defense Council (NRDC), Aug 2014.
- [8] N. Rasmussen, *Implementing energy efficient data centers*, 2006.
- [9] Synergy, *Aws remains dominant despite microsoft and google growth surges*, <https://www.srgresearch.com/articles/>, published: Feb 03, 2016.
- [10] S. R. Awasthi, S. Chalise, and R. Tonkoski, “Operation of datacenter as virtual power plant,” in *Energy Conversion Congress and Exposition (ECCE), 2015 IEEE*, IEEE, 2015, pp. 3422–3429.
- [11] L. B. N. Laboratory, *Benchmarking data centers - case study reports*.
- [12] Google, *Efficiency: How we do it*, <https://www.google.com/about/datacenters/efficiency/internal/>.
- [13] S. Chalise, A. Golshani, S. R. Awasthi, S. Ma, B. R. Shrestha, L. Bajracharya, W. Sun, and R. Tonkoski, “Data center energy systems: Current technology and future direction,” in *2015 IEEE Power & Energy Society General Meeting*, IEEE, 2015, pp. 1–5.
- [14] J. L. Berral, R. Gavalda, and J. Torres, “Adaptive scheduling on power-aware managed data-centers using machine learning,” in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, IEEE Computer Society, 2011, pp. 66–73.

- [15] G. Wen, J. Hong, C. Xu, P. Balaji, S. Feng, and P. Jiang, “Energy-aware hierarchical scheduling of applications in large scale data centers,” in *Cloud and Service Computing (CSC), 2011 International Conference on*, IEEE, 2011, pp. 158–165.
- [16] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, “Optimal power cost management using stored energy in data centers,” in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, ACM, 2011, pp. 221–232.
- [17] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing, “Utilizing green energy prediction to schedule mixed batch and service jobs in data centers,” *ACM SIGOPS Operating Systems Review*, vol. 45, no. 3, pp. 53–57, 2012.
- [18] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, “Statistical profiling-based techniques for effective power provisioning in data centers,” in *Proceedings of the 4th ACM European conference on Computer systems*, ACM, 2009, pp. 317–330.
- [19] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood, “Power routing: Dynamic power provisioning in the data center,” in *ACM Sigplan Notices*, ACM, vol. 45, 2010, pp. 231–242.
- [20] X. Ren, R. Lin, and H. Zou, “A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast,” in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, IEEE, 2011, pp. 220–224.
- [21] P. Saripalli, G. Kiran, R. R. Shankar, H. Narware, and N. Bindal, “Load prediction and hot spot detection models for autonomic cloud computing,” in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, IEEE, 2011, pp. 397–402.
- [22] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, “Prediction-based dynamic resource allocation for video transcoding in cloud computing,” in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, IEEE, 2013, pp. 254–261.
- [23] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, “Prediction of cloud data center networks loads using stochastic and neural models,” in *System of Systems Engineering (SoSE), 2011 6th International Conference on*, IEEE, 2011, pp. 276–281.
- [24] S. Di, D. Kondo, and W. Cirne, “Google hostload prediction based on bayesian model with optimized feature combination,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1820–1832, 2014.

- [25] *Load forecasting model whitepaper*, <http://www.pjm.com/~media/planning/res-adeq/load-forecast/2016-load-forecast-whitepaper.ashx/>, A White Paper, published by Resource Adequacy Planning Department, PJM Interconnection, April 27, 2016.
- [26] *2015 miso independent load forecast*, <https://www.misoenergy.org/Planning/Pages/IndependentLoadForecasts.aspx>, Prepared for MISO, November 2015.
- [27] *Forecast model structures of the iso new england long-run energy and seasonal peak load forecasts*, http://www.iso-ne.com/static-assets/documents/2015/05/forecast_model_structures_2015.pdf, Prepared 2015 CELT report and 2015 Regional System Plan.
- [28] B. Kepes, *Understanding the cloud computing stack: Saas, paas, iaas*, <https://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/>.
- [29] J. Ferrell, *The secrets of weather forecast models, exposed*, <http://www.accuweather.com/en/weather-blogs/weathermatrix/why-are-the-models-so-inaccurate/18097>.
- [30] K. Poortema, “On modelling overdispersion of counts,” *Statistica Neerlandica*, vol. 53, no. 1, pp. 5–20, 1999.
- [31] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [32] G. D. Forney Jr, “The viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
- [33] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.
- [34] L. E. Baum and G. Sell, “Growth transformations for functions on manifolds,” *Pacific Journal of Mathematics*, vol. 27, no. 2, pp. 211–227, 1968.
- [35] G. Schwarz *et al.*, “Estimating the dimension of a model,” *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.
- [36] *What’s the bottom line? how to compare models*, <http://people.duke.edu/~rnau/compare.htm>.
- [37] I. Visser, M. Speekenbrink, *et al.*, “Depmixs4: An r-package for hidden markov models,” *Journal of Statistical Software*, vol. 36, no. 7, pp. 1–21, 2010.
- [38] I. Visser, M. Speekenbrink, *et al.*, “Depmixs4: An r-package for hidden markov models,” *Journal of Statistical Software*, vol. 36, no. 7, pp. 1–21, 2010.
- [39] *Nrel rsf measured data 2011*, <http://en.openei.org/doe-opendata/dataset/nrel-rsf-measured-data-2011>.

- [40] *Nrel rsf weather data 2011*,
<https://en.openei.org/datasets/dataset/nrel-rsf-weather-data-2011>.
- [41] *Moving averages*, <http://www.onlinetradingconcepts.com/>.
- [42] M. H. Ullah, S. Chalise, and R. Tonkoski, “Feasibility study of energy storage technologies for remote microgrid’s energy management systems,” in *2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, IEEE, 2016, pp. 689–694.
- [43] S. Chalise, J. Sternhagen, T. Hansen, and R. Tonkoski, “Energy management of remote microgrids considering battery lifetime,” *The Electricity Journal*, 2016.
- [44] B. F. Hobbs, S. Jitprapaikulsarn, S. Konda, V. Chankong, K. A. Loparo, and D. J. Maratukulam, “Analysis of the value for unit commitment of improved load forecasts,” *IEEE Transactions on Power Systems*, vol. 14, no. 4, pp. 1342–1348, 1999.
- [45] M. Schillebeeckx, B. Maricque, and C. Lewis, “The missing piece to changing the university culture,” *Nature Biotechnology*, vol. 31, no. 10, pp. 938–941, Oct. 2013, ISSN: 1087-0156. DOI: 10.1038/nbt.2706. [Online]. Available: <http://dx.doi.org/10.1038/nbt.2706>.