

Energy Disaggregation with Aggregate Home Energy Signals

by

Dean Rodhouse

This thesis is presented in partial fulfilment of the requirements for the degree of
Bachelor of Computer Science with Honours at Monash University

Supervisors:

Dr. Reza Haffari, Dr. Lachlan Andrew, Dr. Ariel Liebman


Clayton School of Information Technology

Monash University

November, 2014

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the work of others has been acknowledged.

Signed by 

Name: Dean Rodhouse

Date: 14/11/2014

Acknowledgements

I would like to thank my supervisors, Dr. Reza Haffari, Dr. Lachlan Andrew and Dr. Ariel Liebman, whose support and guidance throughout the entire course of the year has helped me to understand this project and improve my research skills.

Abstract

Non-intrusive Load Monitors (NILMs) have shown to be useful in providing information on how electricity is used in a household. Simple feedback from this monitoring has shown to reduce energy consumption of everyday living. In this study we explore the use of the Hidden Markov Models (HMMs) in performing energy disaggregation with use of the Reference Energy Disaggregation Dataset (REDD) we disaggregate energy signals that are made up of a set of devices found within a household. Initial tests inspired us to further look into methods to increase the accuracy and efficiency of the Viterbi inference method. The experiments in this study looked into how the combination of different devices affected the accuracy of Viterbi. We further discuss how big noisy devices can affect the detection of small usage devices and provide a method of distinguishing between them. This method involves creating a split between such devices and running inference of them separately. Results from testing this method showed that it can help to increase accuracy of inference methods, but it can also decrease accuracy as well. This particular method requires further study into looking at the best way to create the divide between big and small devices.

Table of Contents

Chapter 1: Introduction	1
1.1. Overview	1
1.2. Research Scope and Contribution	1
1.3. Thesis Structure	2
Chapter 2: Literature review	3
2.1. Energy Disaggregation Background	3
2.1.1. <i>Outcomes of energy disaggregation</i>	3
2.1.2. <i>Non-intrusive monitors</i>	4
2.1.3. <i>Methods of energy Disaggregation</i>	5
2.2. Factorial Hidden Markov Models	7
2.2.1. <i>Hidden Markov Models</i>	7
2.2.2. <i>Viterbi Inference</i>	9
2.2.3. <i>Further extensions</i>	10
2.2.4. <i>Applications of FHMMs</i>	11
2.2.5. <i>Energy Disaggregation with FHMMs</i>	11
Chapter 3: Methods	13
3.1. Framework for FHMM	13
3.2. Building specific Models	13
3.2.1. <i>REDD</i>	13
3.2.2. <i>Reconstruction of state models</i>	14
3.3. Inference Techniques	16
3.3.1. <i>Viterbi</i>	16
3.3.2. <i>Naïve benchmarking</i>	17
3.4. Innovations	17
3.4.1. <i>Device separation for multiple passes of Viterbi</i>	17
3.4.2. <i>Methods for increasing efficiency</i>	18
3.5. Experiments, Output and Performance measures	19
Chapter 4: Results and Discussion	20
4.1. Preliminary results	20
4.1.1. <i>Justification of using a median filter</i>	20
4.1.2. <i>Benchmarking</i>	22
4.1.3. <i>Performance Limitations</i>	23
4.2. Results after changes to the framework	24
4.2.1. <i>Detecting a single devices</i>	24
4.2.2. <i>Combining devices with ground-truth labels</i>	26
4.2.3. <i>Detecting small usage devices amongst big devices</i>	28
Chapter 5: Conclusion	31
5.1. Summary and contribution	31
5.2. Future work	31
Chapter 6: References	33
Chapter 7: Appendices	35

Chapter 1: Introduction

1.1. Overview

The primary aim of this research is to create a tool to analysis the energy usage of individual devices in a circuit. The process of doing this is called Non-Intrusive Load Monitoring or Energy Disaggregation. This is the process of taking a whole aggregate energy signal from a household and using inference techniques to reconstruct the individual signals that come from each device in the household [1]. The further aims of this research are to evaluate the accuracy of this tool in performing energy disaggregation and discuss the improvements when alterations to the model are introduced.

Previous research in this field has shown that the knowledge of device level energy signals has incurred savings on power bills by considerable amounts [2]. While this is the main use of having this knowledge other applications have been considered, such as monitoring for faults in devices or even detecting when illegal devices are being used [3].

Methods for energy disaggregation started with attached monitors to circuits which detect power spikes that occur in the energy signal [1]. These power spikes were used to identify devices as they relate to a device being turned on. Research since this has introduced the monitoring of other electrical features including features such as current wavelengths and voltage noise [4].

This research introduces a technique to help to detect devices with low power levels when large noise exists from high powered devices. The idea was formed when initial tests showed a low accuracy of small powered devices when large devices existed in the model. Sometimes these small devices would be detected incorrectly as on when large positive noise was present. Other times these small devices get detected as off when negative noise was present.

1.2. Research Scope and Contribution

This research focuses on use of Hidden Markov Models for energy disaggregation. The main goal of this project was to create a framework for a HMM with the intended use of studying energy consumption. To fulfil this goal the study involves the creation of an automated classifier for the purpose of classifying an energy signal into separate device-level signals. An important distinction is that the tools used in this project which can commonly machine learning tools are not used for automated learning and any parameters for the HMMs are defined beforehand.

The research also looks into the idea of being able to split devices into two groups based on the size of their power usage. The actual choice of how to split devices is not contained in the scope of this research however tests to show that this choice is meaningful are undertaken.

This research contributes an analysis of the accuracy of general HMM methods and shows how minor alterations to the framework can help to increase efficiency and accuracy of the inference methods used. The research also provides future researchers in the field a platform for performing energy disaggregation for them to expand on. The main contribution of this study however is the introduction of a method to better detect small usage devices in a circuit where electrical noise is present.

1.3. Thesis Structure

The remaining chapters are organised as follows:

- Chapter 2 will present the literature review of the topic presenting the relevant concepts and terms for the research. This chapter will also introduce the framework used in this research
- Chapter 3 discusses the research methods used in undertaking this study. The framework creating is explained detailing how parameters for the model are found. Also discussed in this section are the inference methods used and the alterations made to the framework to help increase the accuracy of the model.
- Chapter 4 presents the results of initial tests to justify the use of certain simplifications of the framework. The chapter further discusses the results of benchmarking tests to demonstrate that complex methods are required to disaggregate an energy signal. This chapter further discusses the result of subsequent tests.
- Chapter 5 provides a complete summary of the research and discusses possible future work expanding from this study.

Chapter 2: Literature review

2.1. Energy Disaggregation Background

Energy Disaggregation is the process of taking an entire aggregate energy signal and breaking it up into device-level readings that contribute towards the total aggregate. For example in figure 2.1 below, the plot of the right is an example of a very simple energy signal, the idea of energy disaggregation is to take that reading and find when each device is on. The two plots on the right show what would be expected if you were to disaggregate this signal.

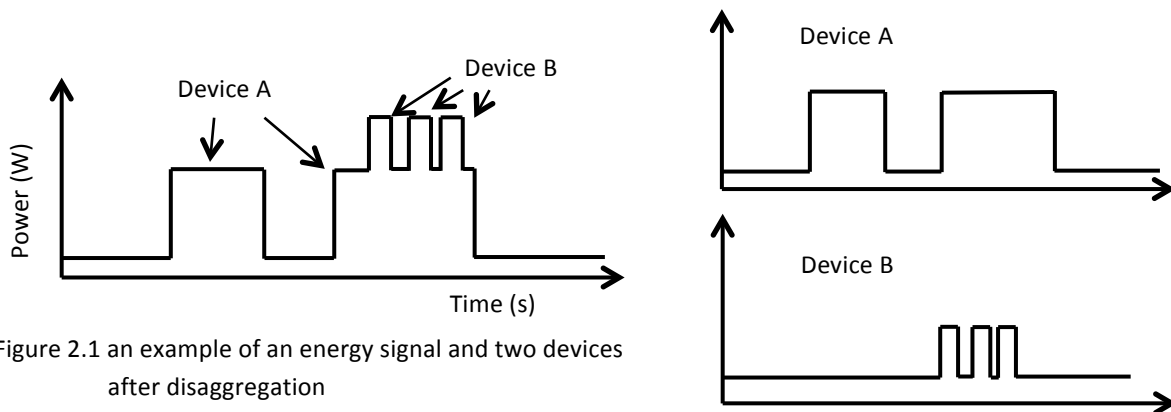


Figure 2.1 an example of an energy signal and two devices after disaggregation

2.1.1. Outcomes of energy disaggregation

Hart [1] outlines many uses of knowing how appliances are used within a house. The most obvious application of knowing where your electricity goes is to provide the information to consumers as feedback so they know exactly what is contributing to their electricity bill. Having this knowledge will help consumers save money and lower their energy usage in general. Previous research has indicated that when consumers are provided with feedback on their energy usage they can incur savings of up to 15% [2,3]. However this saving does depend on the type of feedback given, if it's done via the billing process than savings of up to 10% have been shown [2], whereas the study in [3] assumed there is a direct method to give feedback to consumers on demand which shows a saving of up to 15%.

Another major use of this tool also outlined by Hart [1] is the monitoring of devices to identify faults. This consists of checking for unusual power signals coming from devices, if an unusual signal is detected we can infer that there is a fault with some device in the circuit. Previous field tests have had success in monitoring for faults, in one situation an unusually low signal was found to be coming from an underground septic tank and discovered to be faulty [5]. Other tests have found defective fridges that were using too much power [1]. These particular findings show that this form of energy monitoring can help to lower energy consumption as well as find broken machines that may have had an impact of health and hygiene if not found otherwise. An alternative to identifying faults is to also monitor to predict when devices will need maintenance to prevent faults from ever occurring.

Hart [1] also mentions the use of this for load management control, which is a process used by energy utilities to reduce power usage and demand during peak times. These load management procedures might be to ask consumers to simply reduce the use of high powered devices during peak times, energy disaggregation can be used to monitor if consumers are correctly following these directions [6].

An alternative and somewhat concerning use for disaggregated energy knowledge is the detection of behavioural patterns in people. By having full knowledge of when devices in a household are being used you can monitor for habits and find out when a house is most likely to be empty, however the current data taken from houses is only smart meter readings, which are reported less frequently than the data used in most studies. While it could be used for criminals to detect when to burgle houses it can also be used to monitor for criminal activity in suspected or known criminals. For example a printing press may have a distinctive energy signal and you could monitor a known counterfeiter's house for this signal in order to check if they are forging currency [5], you could even monitor for chemical equipment being used by suspected drug dealers. These concerning uses do bring up a few privacy issues though, current research in this field looks into using non-intrusive load monitors as a way of not having too much knowledge of households.

2.1.2. Non-intrusive monitors

Non-Intrusive Appliance Load Monitor (NALMs) were first introduced by Hart [1] as a device designed to monitor entire aggregate energy signals from a circuit. This is in contrast to previous methods of load monitoring which required using separate monitors for each device in a circuit, this sort of monitoring was seen as intrusive and required too much prior knowledge of a house [5]. There are two main ways in which NALMs can be setup, manual and automatic [1]. The manual setup can be seen as an intrusive approach but only during the setup period. For this type of setup the signatures of each appliance are measured within a household by manually activating every state of the device. The automatic setup however is much less intrusive as it uses prior knowledge about the signatures of possible devices that can be found within a house. This prior knowledge is used to identify the important appliances within a house via the electrical signals without the need to actually go inside and check.

Using the nonintrusive methods have a few advantages over the other intrusive methods, with the obvious improvement being on the time and cost of installing the monitors in a house. Maintaining the monitors is another issue, having to return to a consumer's household to change around monitors whenever devices within the house are changed, since may not occur much but if it was required of every house then it becomes quite time consuming. There have also been studies in the past that show when a person is aware that there energy is being monitored they will unconsciously change their behaviour which gives skewed results when finding the general energy usage of houses [5]. Having the monitors present in a house all the time is shown to be a constant reminder that your energy is being monitored.

Hart [1] categorises consumer appliances into three possible groups that are dependent on the possible state setup of device. These groups are as follows:

Type 1: Devices with only two state or ON\OFF devices, examples of this includes basic lights, toaster, water pumps etc.

Type 2: Multi-state appliance with a finite number of states, known as finite state machines (FSM). Examples of these appliances might be washing machines, refrigerators, stoves and similar devices.

Figure 2.2 over the page shows examples of these two state devices.

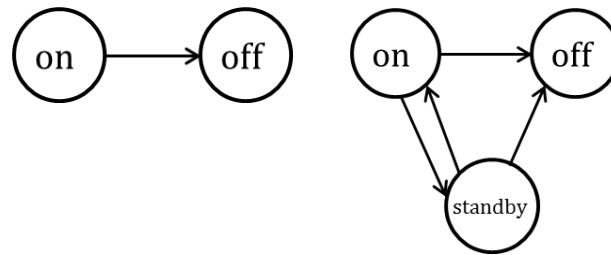


Figure 2.2 Left: an example of an on/off device,
Right: an example of a FSM

Type 3: Continuously variable devices (CVD), these devices have no fixed number of states, some examples of these are power drills, or light dimmers.

A further fourth type of device was introduced in [7], this type 4 device is the set of devices that remain active for long periods of time and consume power at a constant rate. Examples of these types of devices are clocks, smoke alarms or telephones. These devices however are often battery powered and will not be part of the aggregate signal [4].

The basic method of a NALM for energy disaggregation is done by connecting to a circuit to first read the aggregate electrical load, past studies take the readings between 1-3 seconds [3, 8]. The device can then monitor the load by detecting events in the energy signal that relate to some activity performed by an appliance. For example if a step increase is detected it can be seen that some appliance is turned on. The actual size of the increase can then be used to determine what device has been activated by finding what devices are likely to have that same energy signal. Similarly this can be done if a decrease is found, which in the same way relates to an appliances being turned off [1].

2.1.3. Methods of energy Disaggregation

Different methods of energy disaggregation work on different ways, the main differences between methods relate to the types of signals being used to infer appliance usage and the types of devices explained in the previous section. Different methods are also performed with the use of different models, which have their own strengths and weaknesses depending on the type of energy reading being used.

The two main types of device signatures are steady state and transient state signatures. Steady state features are the power signatures of a device when they are in a steady state i.e. non changing state. Transient signatures however use the signatures given by a device during the transitional state in which the behaviour is unstable [4]. Figure 2.3 over the page shows an example of a signature from a device, you can see at the start of each signature is a spike above the rest, then it trails off, the beginning of this signature is a basic example of a transient event as it's when the device turns on, however to properly measure the transient state a higher frequency measurement would be needed. The part that trails off in this graph would be used together to form a steady state signature of the device.

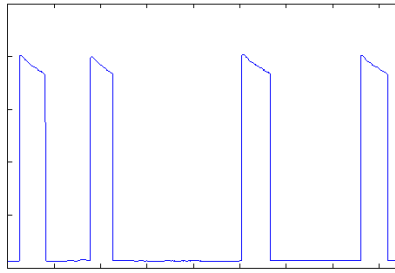


Figure 2.3 Signature of a device showing initial transient signature then steady state signature

Steady State Signatures

Both signature types then have their own individual types of readings as well. For the steady state methods there are three main types, the first we have already seen with the most basic NALM approach by considering the power change approach [1]. This method basically searches for spikes in the power signal and identifies which device caused that spike. However having only the one variable to identify a device limits how well the method can identify a device, if there were two devices with a similar power spike this method would not be able to tell the difference between the two devices.

An approach to remedy this uses more than just power data, these methods use voltage and current information as well to help to identify devices [4]. This uses the harmonic signatures of the voltage and current on a circuit, there are several characteristics of these harmonics that are used, such as the root mean squared or phase difference, these are quite technical and won't be expanded on. The use of all of these factors results in a greater accuracy of the disaggregation, however there are still some issues present with the overlap of signatures. There is also an issue when retrieving this data, to properly define every possible harmonic signal you need to find the harmonics for every combination of devices in the house, this is quite a large sample size and is very time consuming [9].

While these methods may be quite accurate when identifying operating devices, they don't work with all types of appliance. The simple power change approach is only set up for simple On\Off devices, whereas harmonic signatures will expand and work on FSMs and some CVDs with low accuracy [4]. A third steady state feature is defined as the voltage noise which can be used to correctly find CVDs. Voltage noise is measured by the EMI signatures made from devices that use a switch mode power supply (SMPS), this type of power supply generates high frequency EMI signatures that is used as an identification for the device. Methods using this signature type are very accurate however are only available for devices with SMPS, which is a small subset of devices [10].

Transient State Signatures

Transient methods take similar approaches, but only using readings during the transitional phases of a device. The simplest method is to simply monitor for power changes that relate to the transient state of devices, this method can work for any type of device as well as being able to differentiate overlapping signatures. Methods using this have issues when there is the rare instance of simultaneous state changes in multiple devices, the resulting spike from a simultaneous change may not relate to any device or be confused with another device [4]. Due to this issue to properly find the transient state of every device you need to find the transient state of every combination of devices, similar to that of voltage noise. Another issue with this that arises in many transient signature methods is the difficulty of measuring at a high enough frequency to capture every feature of the change.

Transient methods have other signature types similar to steady state methods, with categorising devices based on the initial spike in the current [11] or even measuring the voltage noise during the transient event of a device [12]. These feature many of the same strengths and weaknesses as the steady state methods, but are found to be more accurate, however are harder to measure as they are high frequency measurements [4].

Types of Models

Some past studies have looked into the use of additional features outside of the energy reading to increase the accuracy of the models used, these often include time of day data [8], current weather conditions or even the light and audio level readings in the environment at the time [13].

Quite a few different inference models have been used to perform energy disaggregation in the past. The models don't all work the same way, some working with different signatures, some working better on certain types of devices, all having a different range in accuracy [4].

The simplest of these models is a Bayesian network [4], these networks are steady state models and only work with FSMs. This model detects the most likely state sequence based on the probabilities contained within the model, the states are also assumed to be independent of each other.

Another model is a Hidden Markov Model (HMM) [7, 14], which is the model used in this research and will be the focus of the entire paper. They are similar to a Bayesian network working with the same devices, however the underlying probabilities and dependencies are a bit more complex, HMMs will be explained further in section 2.2.

Two other models that have been used in past studies that both use transient features of devices are Support Vector Machines and Neural Networks. These models both work with a large amount of devices, they are generally used for finding patterns in data which can be applied to finding patterns within an energy signal to isolate different devices [15]. Neural Networks can also work without prior knowledge of the environment, however they required exhaustive training when a new appliance is added to the system [7, 16].

While there are many inference models that have been used for energy disaggregation in the past there is one key difference with Hidden Markov Models. Past studies have used HMMs without supervised training, which gives the machines a minimal setup time and cost which when using other models with manual learning can be very cost and time expensive [4].

2.2. Factorial Hidden Markov Models

2.2.1. Hidden Markov Models

A Markov Model is a stochastic process made up of a number of states and transitions rates relating to the likelihood of moving from one state to another. Each transition rate depends only on the current state and does not depend on any past or future states. Markov models are often used in research for processes in which the state is always observable, in contrast to this however are Hidden Markov Models (HMMs). HMMs are much similar to Markov Models with the key difference that the active state at any given time is non-observable. Instead we make an observation based on some distribution within each states. HMMs still contain the regular transition rates like a regular Markov Model. The combination of the observation and the transition rate leads to a doubly-stochastic process with two underlying probability distributions [14, 17]. HMMs are chosen in studies where we do not know the state of a process and are required to find the active set of states. Ghahramani [17] represents a HMM with two sequences, one for states and one for observations, as defined above the sequence of states $\{S_t\}$ which is the active state at time t , is hidden from the observer. The other sequence which is observable is the observation sequence $\{Y_t\}$.

The two probabilities are defined as:

$$P(S_{t+1}|S_t) \quad (1)$$

$$P(Y_t |S_t) \quad (2)$$

Where (1) is the probability of the active state being S_{t+1} given that the active state at time t was S_t . And (2) is the probability of making the observation Y_t given that the current active state is S_t . Figure 2.4 below shows an example of these two sequences, the top row being the state sequence which only has dependencies on previous states in the sequence. Below the states is the observation sequence with dependencies only coming from each state at any given time.

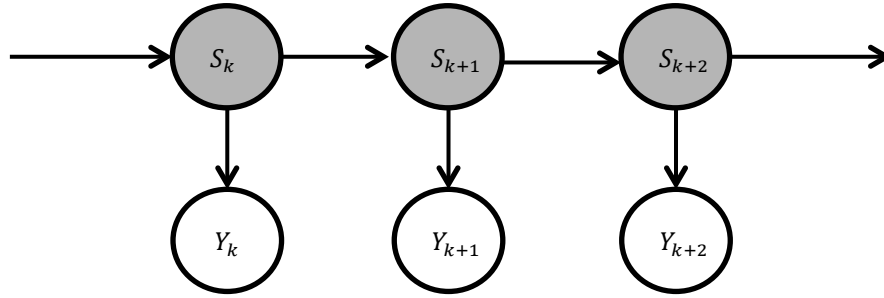


Figure 2.4 A directed graph showing the relations between the states and observation sequences

There are three tasks that HMMs are commonly used for, each task requires its own algorithm to solve. These problems are defines as:

Problem 1 – given a possible observation sequence, and a model with known probabilities, what is the likelihood of the observation sequence?

Problem 2 – given an observation sequence, what is the sequence of states that produced this output?

Problem 3 – How can we adjust the model to maximise the probability of a given observation sequence.

The process of energy disaggregation relates directly to solving problem 2.

Each of these problems uses a different algorithm to solce, For problem one the forward-backward procedure can be used [19]. This method defines two new variables in the model, the forward variable α_t , this is defined to be the probability of a partial observation sequence from $(1,t)$. The other variable is the backward variable β_t which is the probability of the other end of the observation sequence from $t+1$ to T , where T is the end of the sequence.

These two variables are defined mathematically as:

$$\begin{aligned} \alpha_t &= P(S_t, Y_{1:t-1}) \\ &= \left[\sum_{S_{t-1}} \alpha_{t-1} P(S_t | S_{t-1}) \right] P(Y_t | S_t) \\ \beta_t &= P(Y_{t+1:T} | S_t) \\ &= \sum_{S_{t+1}} \beta_{t+1} P(S_{t+1} | S_t) P(Y_{t+1} | S_{t+1}) \end{aligned}$$

Using these probabilities we can calculate the entire α_T which gives us the probability of the entire sequence up until the last state, where we can just marginalize over the final state to find the probability of the full sequence $P(Y)$.

Problem 2 is solved by use of the Viterbi algorithm, this algorithm works similar to the forward-backward procedure, however rather than having the summation process to find the probability, instead the max previous state probability is chosen for each state, this algorithm will be explored more in-depth throughout the paper, more background is given in section 2.2.2.

Problem 3 is a learning problem, one way to learn the model is done by using the Baum-Welch algorithm [14]. This algorithm defines two new variables of the model, these are γ_t the probability of being in any state at a particular time given the observation sequence. The other variable is ε_t , the probability of going from state S_t to S_{t+1} given the observation sequence.

$$\begin{aligned}\gamma_t &= P(S_t|Y) \\ &= \frac{\alpha_t \beta_t}{P(Y)} \\ \varepsilon_t &= P(S_t, S_{t+1}|Y) \\ &= \frac{\alpha_t P(S_{t+1}|S_t) P(Y_t|S_t) \beta_t}{P(Y)}\end{aligned}$$

These two variables are re-estimations of the probabilities in the transition and observation features of the model, the algorithm works by repeatedly running this re-estimation until a convergence is met.

2.2.2. Viterbi Inference

The Viterbi algorithm is used to find an answer to problem (2) in the Hidden Markov Model section. Using this algorithm we can infer the most likely state sequence in a HMM given an observation. This is a dynamic algorithm and defines two new equations to calculate the likelihood of states and the optimal previous state which maximises this likelihood, the equations are as follows [14]:

$$\begin{aligned}\delta_t &= \max_{S_{t-1}} [\delta_{t-1} P(S_t|S_{t-1})] P(Y_t|S_t) \\ \varphi_t &= \arg \max_{S_{t-1}} [\delta_{t-1} P(S_t|S_{t-1})]\end{aligned}$$

δ_t is the probability of a state and observation occurring given a previous state, the previous state is chosen as the state that maximises this probability, φ_t stores this state. The algorithm can be defined in 4 main steps these are:

Step 1 – Initialization, this set up the initial values where $t = 1$.

Step 2 – Recursion, this iterates over the given observation signal and calculates the probability of each state, and finds the previous state to maximise the probability,

Step 3 – Termination, when the recursion stops i.e. reaches the end of the observation sequence the max value at the end of the observation chain is chosen and is used as the final state in the model.

Step 4 – Path backtracking, this uses the second equation φ_t to create a path back to the beginning on the observation chain based on the chain of best previous states that was calculated. The path created is then considered to be the most likely state sequence.

2.2.3. Further extensions

Ghahramani [17, 18] defines an extension to the HMM called a Factorial Hidden Markov Model (FHMM). The model is defined as having each state represented as a collection of state variables, this is also classified as a model consisting of multiple independent HMMs working in parallel with a joint function on the observation distribution in each HMM [19]. The approaches to problem solving with FHMM are very similar for performing inference and learning, with the main difference being how the observation is computed. Figure 2.5 below shows an example of the state and observation sequence similar to that of the regular HMM, instead consisting of three underlying Markov chains. The figure shows that at each step the states of the three HMMs all contribute to the total observation at that point.

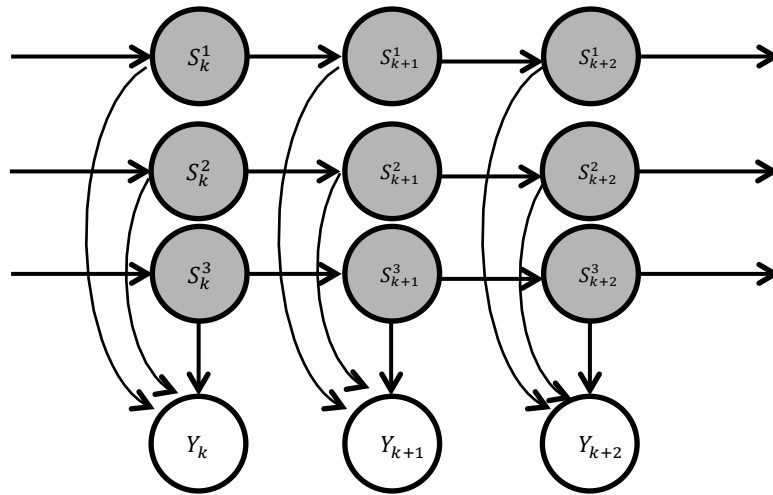


Figure 2.5 A FHMM built up of three independent HMMs

Other extensions look into added extra dependencies between HMM states and additional outside factors [8]. Extra data can be added to incorporate conditional dependencies of states occurring given what the additional data is, this could be information based on the time of day or year, or even environmental factors. There are two ways to incorporate these factors, the first way extends the FHMM again to create a Conditional Factorial Hidden Markov Model (CFHMM). In this model each state is dependent on the additional factors as well as the previous state in the state sequence.

Another extension to HMMs is known as the Hidden Semi Markov Model (HSMM) [8]. This extension introduces the idea that states in the model are likely to stay active for a fixed amount of time before transitioning to another state. An example of this is the cycle of a refrigerator, these cycles are likely to occur every few hours and last a fixed amount of time. Much like regular HMMs this model can also be given factorial and conditional factors.

2.2.4. Applications of FHMMs

Previous research have used HMMs in many other applications as well, some of these include:

Word Recognition

For isolated word recognition the model has a vocabulary of V words and a training set of K occurrences of each word, such that each occurrence is spoken by a different talker. Each occurrence of a word also represents an observation sequence for the model, the observations are a representation of the spoken word.

The speech recognition is performed by creating a set of HMMs based on the vocabulary of the model, and using the training set to estimate the parameters of each state. Once the model is learnt from the training set we can perform speech recognition on a test set, which is just another set of occurrences of spoken words. Using the test set $\{Y_t\}$ we can calculate $P(Y)$ for each word model and then choose the model with the highest probability to find the most likely word for the set. This can be done with just the forward-backward procedure, however the Viterbi algorithm is commonly used to find the maximum likelihood path for the words [20].

Gene Prediction

There have also been studies using HMMs for gene prediction [21]. The states in the model represent either coding or non-coding DNA, while the observations relate to an observed DNA sequence. The study in [21] focuses on the genes found in viruses, however this can be used for predicting genes in anything, in general the HMMs in this can be altered represent any sort of genome. The methods in this study use the Viterbi algorithm to predict the most likely sequence of genomes.

Ozone Level Prediction

In a previous study [22] with ozone level prediction HMMs have been used to represent different levels of ozone, different areas get separated into different amounts of HMMs but in general the level of ozone in ppb (parts per billion) is used to separate each HMM.

Unlike regular HMMs for the observation of the model there are multiple sequences rather than one, and hence more distributions for each state. The observation sequences for this model are observations of climate data such as wind speed, temperature, humidity, solar radiation, etc. The forward-backward procedure is used to predict which ozone level a location has based on the multiple observations made.

2.2.5. Energy Disaggregation with FHMMs

Factorial Hidden Markov Models are commonly used for energy disaggregation, they can be setup by having each individual HMM in the model to represent an particular device in the household [3]. It is quite simple to represent finite state machines as a HMM, having each state in the HMM relate directly to each state in the machine. The transition rates are simply the probability of a device changing state, and the observation distribution is simply the expected power usage the active states in each device. Generally the joint function for the observation is just the aggregate of the energy signals coming from each device. However there are some approaches that have looked at the observation being just the change in power consumption over a transition [19].

There have been many past studies using FHMMs for energy disaggregation, in general they are used for steady state features as they are easier and cheaper to measure [8]. Some methods also choose to use approximate inference over an exact procedure for when models get too complex [23]. Other studies look into using multiple features of an electrical signal to increase the accuracy of identifying devices. These methods look into use real and reactive power along with current and voltage harmonics of an electrical signal. Using these features has allowed the past studies to increase the accuracy of more complex state devices, FSMs and CVDs [23].

Altered learning procedures similar to the Baum-Welch algorithm have been used to train the model to specifically increase the accuracy of the inference procedures used [24]. These alterations look into finding residuals between the real output of a device and the inferred output and modifies the model parameters in order to better detect the active states.

While the general approach is to use an additive function over the observation the study performed in [19] looks into exploiting the features of the aggregate total to find the power change made at any time, as a difference function of the observation. As a result of this the study also only considers the use of one device changing state at a time, to properly define a model that considers every state to change more information would be needed for the change in the power for every combination of state change.

A study by Kolter [19] uses exact MAP inference to derive methods for approximate MAP inference by considering only one device changing state at a time. Both the additive and difference models were considered, giving the resulting algorithm named Additive Factorial Approximate MAP (AFAMAP). This AFAMAP algorithm was compared to another approximate inference procedure known as structured variational mean field (SMF). AFAMAP was generally found to have greater accuracy than SMF, the downsides to these approximate algorithms is that they are only approximate, these algorithms may never find an exact solution. The assumption that only one device can change state also lowers the accuracy in the rare case when multiple devices change state.

An alternative to altering the inference techniques is to alter the model itself, this is commonly done with the consideration of outside factors such as the environment, or the time of the day/year [8]. The study in [8] introduced the CFHMM described in the previous section by incorporating this outside data that has no direct effect on an energy signal. The study also introduced the HSMM, taking into account that an appliance in a house may be used for fixed periods of time, i.e. refrigerator or a washing machine. The HSMM can also be used to include dependencies between devices, such as dryer is commonly used after a washing machine is used.

All these changes cause the model to become more complex, making it more computationally demanded than the regular FHMM. Due to the increased complexities the alternative methods of inference have been used. The study that introduced the conditional and semi models used simulated annealing to find an estimation of a state sequence, as iterating over every possible state giving all the dependencies can be very computationally intractable. The general accuracy of the CFHMM was greater than that of methods used with the FHMM, however being a lot slower in computation [8].

Chapter 3: Methods

3.1. Framework for FHMM

The research performed in this study requires the use of the previously mentioned Factorial Hidden Markov Model. The setup of the Factorial Hidden Markov in this study is similar to what has been done in past studies [3]. The FHMM contains n independent HMMs with each HMM representing a different device, each HMM then contains a different set of parameters specific to that device.

In each HMM there exists a set of states that describe the possible states of the device, usually having one state to say the device is 'off' then many states describing one or more possible 'on' states. Each possible state for each HMM also contains three probability distributions to describe the usage of the devices, first there is the initial probability of each state which describes the likelihood of the device being in a particular state. There is also the transition distribution which describes the probability of a device changing state, which can be seen as a user operating a device. The third distribution describes the energy output measured in watts that is generated from a particular state. This is represented as a Gaussian distribution. The observation sequence is simply the power reading of a device or aggregate reading from multiple devices at a particular time step.

3.2. Building specific Models

After describing the parameters we need to define a HMM, we will look into how we can calculate their values using the Residential Energy Disaggregation Dataset (REDD) [3].

3.2.1. REDD

This dataset from REDD was gathered in a previous study for the purpose of further research in Energy Disaggregation, this data was easily accessible and provided us with what was needed to undertake the study. The dataset was gathered on a range of different houses over the course of several months, this meant that the data wouldn't cover the full year and every season so the readings may be heavily influenced by the time of year it was gathered.

For each house in the set, there were multiple different circuit readings each showing the power usage of different device or a set of devices. The data acquired for this study contained readings for 6 houses with up to 25 individual circuits from each house. Only one of these houses was focused on during the experimental phase of the study, this house had 20 circuits, 2 of which were for the mains power the rest all representing different devices within the house.

The signals themselves consist of lists of timestamps accompanied by the energy reading made at that time. For the mains power they have taken measurements every second and for each separate device they have taken readings every 3 seconds. Some of the circuit readings relate to individual appliances, such as the fridge or oven. But there are other readings that relate to a group of devices such readings are called "Kitchen1" or "Lights1" which relate to a group of kitchen appliance (toaster, kettle, etc.) or a group of lights.

Using the channel data on each device we were able to find the relevant parameters for the FHMM that described the energy output of the device. The method behind this required us to perform multiple algorithms to manipulate the data. For the purpose of testing we only used the first half of each circuit reading when finding the parameters, leaving the second half to be tested on.

3.2.2. Reconstruction of state models

In order to create the state space of each device we took the channel data of this device used clustering techniques to assign each reading to an individual state. Before we could run a cluster algorithm we had to first make a few manual choices on the data. By observing the data we can see such things as found in figure 3.1 below that every instance of the usage of this device has approximately the same energy output, with a number of spikes lasting for about the same about of time. The assumption from this is that there would be one state for the off state, with a reading of 0 and one state for the on state with a reading of around 4000. The clustering algorithm would then be given the prior information that this device has two states, and a chosen energy reading for each state.

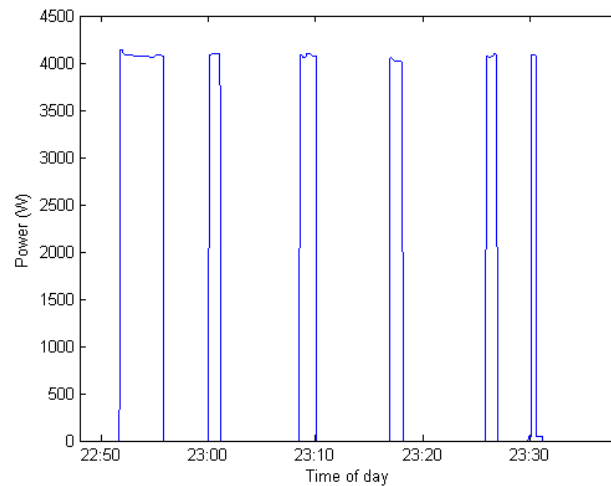


Figure 3.1 Part reading from the Oven, each spike representing the 'on' state of the device.

The algorithm using is the K-Means clustering algorithm [25]. This in short associates every observation point to a single cluster based on how far that point is from the centroid. After every point is assigned to a cluster the centroids of each cluster are then updated by calculating the average of the values within each cluster, after the centroids are updated the assignment and updating processes are then repeated until convergence i.e. The centroids are unchanged after updating. As the flowchart in figure 3.2 demonstrates the clustering process consists of continuously running the assignment and updating phases until a convergence property is met.

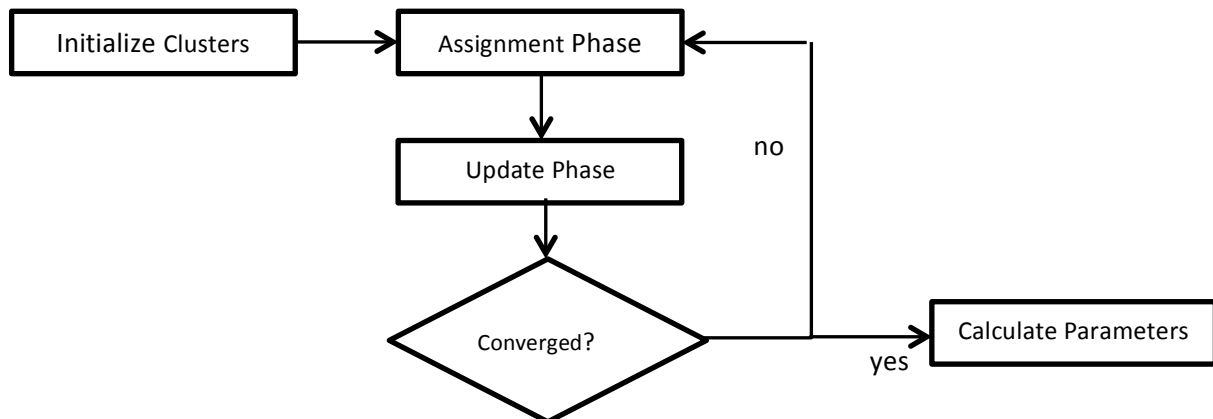


Figure 3.2 Flowchart showing the steps in the clustering and model building stage.

Once the clustering is done we would have the actual state sequence of the device signal, we can then use this to find the probability distributions needed for the HMMs. To calculate the initial probability we find how many points belong to each cluster and divide by the total number of points. We can then find transition rates by observing consecutive points and their clusters in which they belong to. To do this we found the frequencies of a transition from one cluster to another. This gave us the number of times the data stream would move from one state to another, from this we could then divide the transition frequency from one state to another by the total amount of transitions that came from one state. This would give us the transition rates from each state, resulting in a table of size $S \times S$.

The Gaussian for the observation in each state is found simply by taking the centroid of each cluster which is equivalently just the mean of each state, along with calculating the standard deviation of the data points within each cluster. This gives us the last parameters we need to define a device as a HMM.

After initial modelling of the data issues regarding random spikes that reduced accuracy of the inferred readings and increased complexity were found. In order to accommodate this we chose to perform some pre-processing techniques on the data. Before processing the data into states we performed a filter on the data to eliminate any random spikes and errors that would only last for a split second in the readings. This was inspired from the initial tests that were performed using non-filtered data, which would often infer incorrect states. The data was ran on a median filter [26], this process involved running over a data sequence and at each data point the neighbours of that point and itself are observed and the median of these values is put as the new value for that point in the data sequence. The amount of neighbours observed by the filter can be simply changed before the run, for this project we have used a filter of size 7, which would observe two neighbours on either side of a data point when calculating the median. In figure 3.3 below you can see the circuit reading of the fridge being put through the median filter. On the left is the signal before filtering and the right has the signal after filtering, you can clearly see large spikes being removed and a general smoothing of the signal.

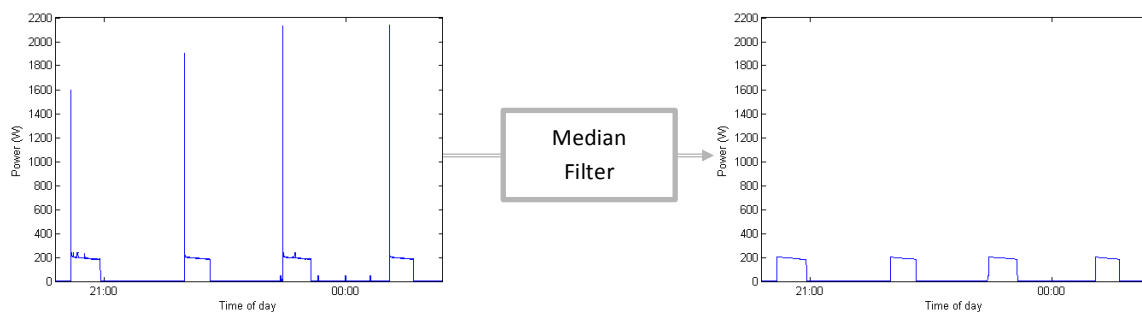


Figure 3.3 Reading from the fridge being put through the median filter.

3.3. Inference Techniques

3.3.1. Viterbi

The implementation of the Viterbi algorithm, defined in section 2.2.2, requires two tables to store the likelihood of each state occurring at each time step and the state from the previous time step that maximises this likelihood.

The actual process of the algorithm is as follows:

```
//This loop finds the initial probabilities at the state of the observation sequence.
For each state s in the state space S
    ViterbiTable (0,s) = Initial probability of s
    Backpointer(0,s) = -1

//This loop fills in the tables with the probability of each state at each time step.
For each time step t in the observation sequence O
    For each state S
        ViterbiTable(t,S) =  $\max_{S_{t-1}} [ViterbiTable(t-1, S_{t-1}) P(S_t | S_{t-1})] P(O_t | S_t)$ 
        Backpointer(t,S) =  $\operatorname{argmax}_{S_{t-1}} [ViterbiTable(t-1, S_{t-1}) P(S_t | S_{t-1})] P(O_t | S_t)$ 

//This section finds state most likely to represent the end of the sequence and backtracks over the
//backpointer table.
maxS = max(ViterbiTable (end))
For each time step t = end to 0
    Path(t) = maxS
    MaxS = Backpointer(t,maxS)
Return Path
```

The first loop performs the initialization state finding the starting probabilities of each state, iterates over every position in the observation sequence to find the probability of that observation occurring in each state as part of the recursion step of the algorithm. After the loop is completed the value maxS is chosen as the end of the state sequence as part of the termination step. After this the backpointer table is iterated over to find the full state sequence for the observation as part of the final phase of the algorithm, the path backtracking.

The two tables for this algorithm become quite large having $|O| * |S|$ values. $|O|$ being the size of the observation chain and $|S|$ is the total amount of states in the FHMM. The size of the observation chain is quite easy to alter being entirely dependent on how many time steps are being observed, however the size of the state space brings up problems. The amount of states in a FHMM is equal to the product of the amount of states of all the HMMs within the model. As the number of devices included in the model is increased the total number of states is increased quite quickly resulting in quite a few issues that are brought up in chapter 3.4.

3.3.2. Naïve benchmarking

As a way to show that the inference techniques used in the project are actually meaningful and worth the complexity we have created a very basic method of energy disaggregation as a benchmark to compare results with. This method is quite simple and runs on the same parameters used in the rest of the project. The algorithm runs through the observation sequence and at each point the energy level observed is compared to the mean power signal that comes from each state in the FHMM. Simply put if the observation is higher than the mean signal of a state then the algorithm will assume that that state is active and will add it onto the state sequence. This will also only choose the state with the highest signal that fulfils this requirement so every state is not chosen to be active. This doesn't require any sort of probability distribution and the choice of states is independent of any other state. Due to the simplicity of the method it is expected that similar devices will get confused and readings that come from certain devices that are much lower than the mean will also be misclassified.

3.4. Innovations

Preliminary tests revealed a couple of issues regarding the space and time required when performing the inference procedures. This inspired multiple methods for the purpose of simplifying the model

3.4.1. Device separation for multiple passes of Viterbi

The largest contribution that was made to the framework was to separate devices based on their power usage. Then run the inference on each set of devices. This was motivated by the idea that if a device with a high power usage has a high variance was to run at the same time as a device with a power signal that would be encapsulated within the high variance then the small usage device would be incorrectly detected or not detected at all. Another factor that influences this choice was the complexity issues that were faced when running the program on the entire set of devices. To fix this we needed to find a way to have a smaller amount of states in the model without a significant lose in accuracy.

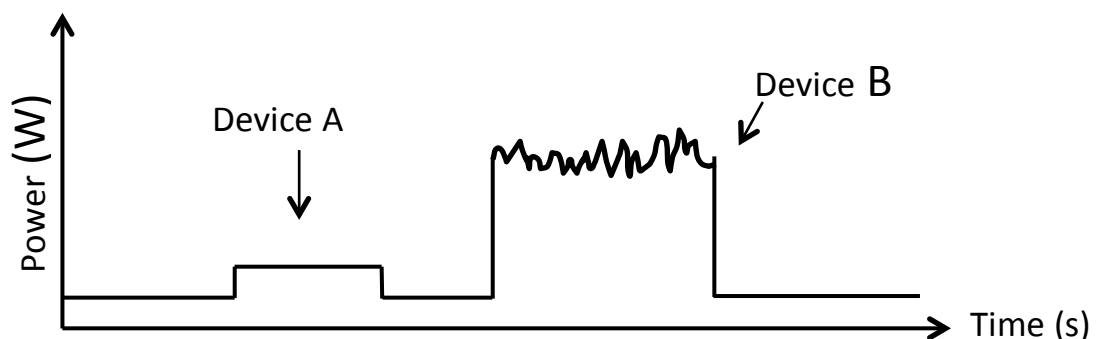


Figure 3.4 Example of a big and a small device, the noise in the big device is larger than the signal of the small device

As a result of this choice a dividing line between 'big' and 'small' devices is chosen, then each device with a power signal above this line is chosen as a big device the rest are considered small devices. After this split is made the inference is ran on the big devices followed by another run on the small devices. However during the run for the small devices the algorithm would ignore any

time steps when a big device is considered on. Having this ignore when big devices are being used lets us remove any mistakes caused by the variance of the big device. When a time step is ignored for a small device the previous state at the first non-skipped point is assumed to be the first state before the time steps that were skipped. One drawback of this method is that a big device may not have a very big variance and the split may not actually affect the entire accuracy of the method.

3.4.2. Methods for increasing efficiency

Further methods were also used in order to reduce the complexity of the model and increase the efficiency of the inference methods.

Change 1. This change was made by collapsing the probability table used in the Viterbi algorithm. This was done purely to save space in the program as it would not run due to the size of the table being too large. The collapsing is quite simple, in the algorithm it iterates over every column in the table, having each column relate to a single observation. Each column is filled with the probabilities of each state which require the previous column to calculate them. However each column itself is only used in another calculation once. This means we can delete previous entries to the table once a column has been completely filled, this lets us change the table which would be of size $|O| * |S|$ to instead be of size $2|S|$. This shortens the table by $1/|O|$ which for larger test is a significant amount. As a result of this change we were able to run more complex models of a larger set of data, however this didn't fix the issue entirely.

Change 2. Another method to try to increase the efficiency of the Viterbi algorithm and reduce the state space was by simplifying devices and reducing the amount of parameters that describe each device. For some devices this could not be done without significantly reducing the accuracy of the model. This was more aimed at the more complex devices which contain a large number of transitions between states. Some similar states were able to be combined with a minor reduction in the accuracy of the device. In figure 3.5 below the signal for a Dishwasher is shown, this contains many changes in the signal that could be seen as a change of state. If we were to model this device perfectly we could define every one of these transitions as a new state, however this would create a huge number of state 30+ for just this device. To reduce the state space for devices like this a fixed number of states were chosen to be approximate to the size of each level. For example in the graph below you may have one at 0, one at about 200, one at about 400 and another at 1100, which are the main levels shown just in this plot. However to properly define it a close look may be needed, some devices have very small usage states and are hidden in larger plots like this.

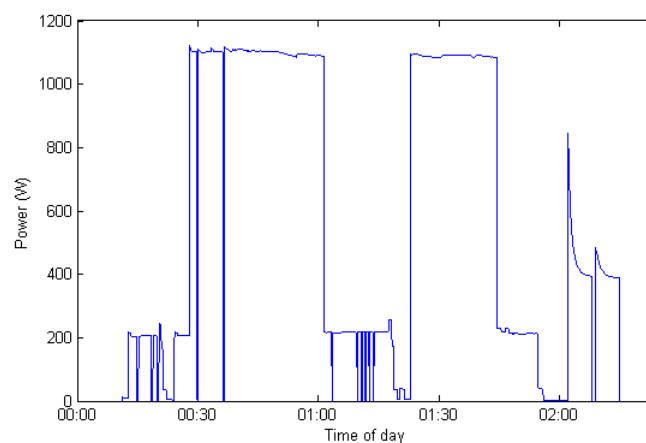


Figure 3.5 Plot of the Dishwasher signal, the most complex signal modelled.

Change 3. The final alteration to Viterbi that was made is during the recursive phase of filling the tables. When the probability of a state is calculated the algorithm looks at every state as a possible previous state and selects the maximum from this. In this change instead of every state being seen as a possibility instead only the states that relate to one device in the model changing are considered. This is under the assumption that no two devices are able to change state at once, which is quite a common assumption in past studies in Energy Disaggregation [19]. The issues with this are that in rare cases it may be possible for multiple states to change at the same time, this leads to slight inaccuracies having devices being assumed to be on about one step longer than they actually are.

3.5. Experiments, Output and Performance measures

The experiments performed in this algorithm use the second half of the circuit readings from each device, as each device is modelled on the first half of the circuit data. The experiments themselves in order to preserve time are mostly performed on subsets of devices to show that the issues explored actually exist in the project and show how the methods introduced help to remedy them.

The setup of any experiment performed used a subset of the devices in one house, with the observation simply being the aggregate of their respective channel data. These subsets were chosen where there were similarities between devices, such as device that are often used at the same time, or devices with similar power levels.

The output of any experiment is inferred reading from each device, to get this we use the expected state sequence found in the model and plot the mean power usage of each device based on its expected state at each point in the observation

To check how accurate our inference is we can observe how often an incorrect state is chosen. This is simply the amount of times a state in the inferred sequence matches the state in the actual sequence divided by the total number of states. We can also check the accuracy of the inferred energy signal by observing the differences between the expected state sequence and the actual state sequence, from this. However this is likely only be the different from the observation and the mean power level of the predicted state. We can use this difference of the inferred and actual signals to check how accurately the states in the constructed device relate to the real device. While checking the accuracy of the correct state being chosen is used to actually check if the correct state is found. Even if there is a high accuracy for finding the right state it may be just because the device is modelled too simply, with too few states.

Chapter 4: Results and Discussion

4.1. Preliminary results

The preliminary tests were performed using small subsets of the devices at a time in order to check that the algorithm could work properly before performing more intensive tests. When doing these tests a couple of performance issues arose, the following two chapters will discuss this and the attempts to fix it.

4.1.1. Justification of using a median filter

The first tests performed were based on models that were derived from the unaltered version of the data. This data contained many random spikes that would last less than one time step (3 seconds), these made the devices harder to model and were often put in the incorrect state leading to large inaccuracies in the inferred usage, this led us to use a median filter to remove this short random spikes. The graphs below in figure 4.1 and 4.2 show the real and inferred usage before and after median filtering was used, the graph in before the filtering was performed clearly shows when these random spikes occurred and the estimated usage at this times is quite off, while after the filtering was used there is a much more smoother plot and the inferred usage is much closer.

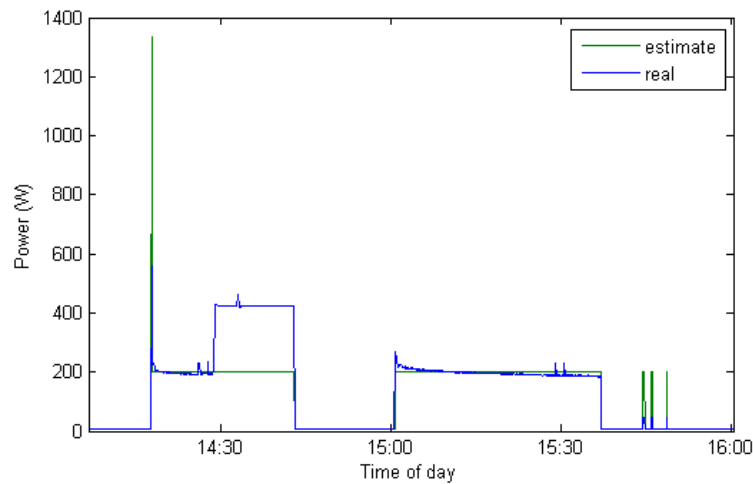


Figure 4.1 Real and inferred usage of the fridge before median filtering.

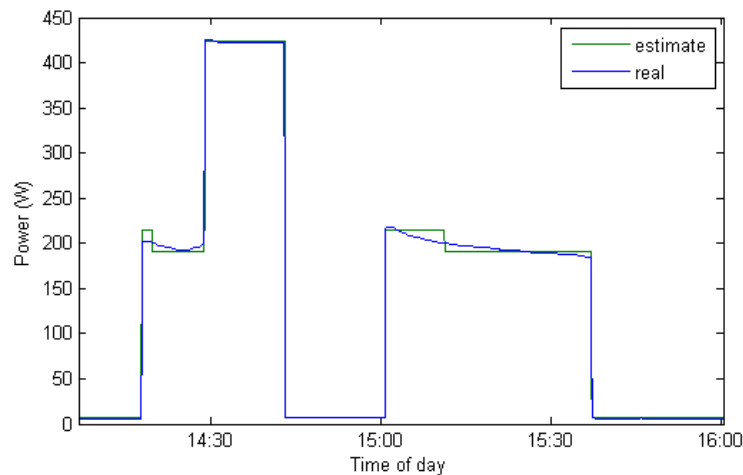


Figure 4.2 Real and inferred usage of fridge after median filtering.

The general accuracy of the inference in the above tests was quite good, actually better in the test before the median filtering having an accuracy of 99.84%, while the test after median filtering is 99.01%. Still very close together, this difference cause is simply because the model created after median filtering ignored the random spikes and was able to be modelled to be much closer to the actual output for the device. An example of this in the above graphs is when the real usage is at 400W, the first test didn't pick this up as a state was not placed for that, while it is found in the second test. So the higher accuracy of the inferred usage is only found because the model representing the device is too simple and there does not exist enough states that they could be confused.

Since comparing the accuracy of the correct state being found doesn't seem to show which of these tests was better we can then check to see the plots of the absolute error in these devices which shows us the actual difference in the energy we think is being used by the device. The accuracy of the correct state however can be used to properly check whether devices can be identified in more complex tests where many devices are present. The graphs below show just that, in graph 4.3 it's shown that there is quite a large error in the predicted output and the actual output, the total range of this error over the total test was (0.19, 933.46)W, with a mean error of 4.50W. The error over for the test after the median filtering had a range of (0.01, 229.01)W with a mean error of 2.14W. This comparison shows a large change after the median filtering allowing us to more accurately predict the usage of devices without random spikes interfering with the modelling and inference processes. As a result of this increase in accuracy it was decided that the rest of the data would be put through a median filter as a part of the modelling phase.

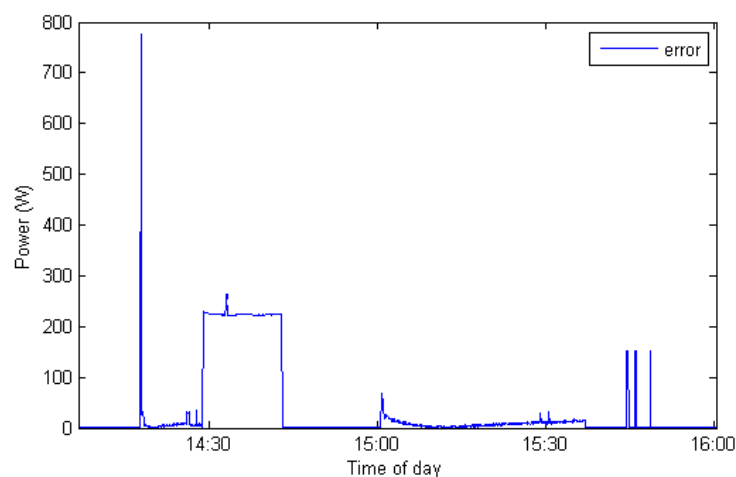


Figure 4.3 Error between the real and inferred usage before median filtering.

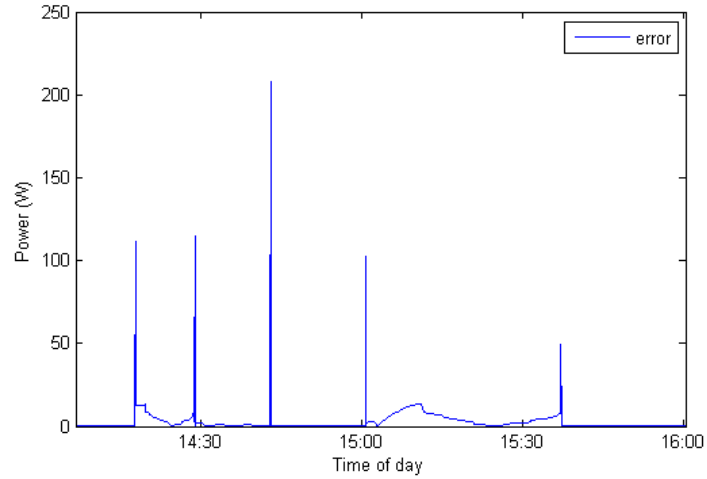


Figure 4.4 Error between the real and inferred usage after median filtering.

4.1.2. Benchmarking

The Benchmarking process as described in section 3.3.2 is a simple method of performing energy disaggregation, the algorithm simply goes over the observation input and if the reading at a particular time is greater than the usage of a state then that state is chosen to be active at that time. The reasoning behind this test is to simply show that our method can achieve a greater accuracy and is worth the complexity.

The test below in figure 4.5 was done on the same fridge input as the previous test found in figure 4.2 however with a different section of the output. It's quite clear in the plot that when the device is being used it finds the correct state only when the reading is above the mean usage of the state leaving a large chunk in each cycle of the device to be incorrectly assigned to a state. A trivial solution to this would be to increase the range of deciding when a state is active, for example rather than having the algorithm check if the power level is above the mean usage it would instead check if the reading is within three standard deviations of the mean, which would cover 99.7% of the readings in a particular state. However when there are two states that have overlapping Gaussians then any reading contained within the overlap may be incorrectly assigned.

The accuracy of find the correct state in this test was 84.14%, the error range is (0.01, 216.51)W with a mean error of 20.71W, so not only is it less accurate but the mean error is larger, meaning the error in the predicted output is much larger.

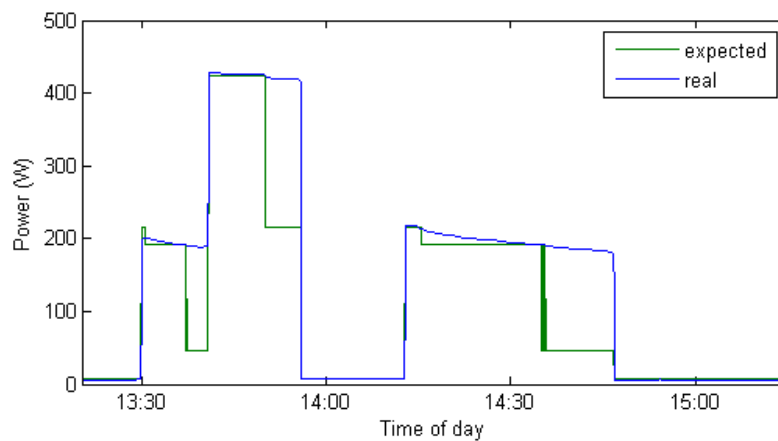


Figure 4.5 the real and expected output from the benchmarking test

The other benchmarking test performed involved two devices, the washing machine and dryer, these were chosen together as they are common appliances to use at the same time. The benchmarking tests showed that there was an accuracy of 99.69% for getting the correct state for the dryer and 99.39% for getting the correct state of the washing machine, however these devices aren't used much so finding the correct 'off' state boosts these devices quite a bit. The chance of finding the 'on' state of these devices when the device was in the 'on' state was 82.91% for the dryer and 66.09% for the washing machine. Due to the lack of accuracy of these tests we have the need of a more complex method of disaggregation, which we will be doing with the FHMM described in chapter 3. Our proposed method when used on this test with the combined dryer and washing machine showed an accuracy of 97.22% for detecting the on state of the dryer and 99.27% for finding when the washing machine was on.

4.1.3. Performance Limitations

Some of the other performance limitations, outside of accuracy was the complexity of the model. Put simply a test on a large subset of devices required too much memory to run. The first test that this occurred on was a test with 6 devices, totalling 648 states in the FHMM, with an observation test size of 372939, half of a circuit reading.

Without being able to run this test it was instead run on smaller inputs, it was decided to use a third of this input instead, however to check how long it would take it was run on an observation chain of size 1000.

As a summary, a test with 648 states, observation chain of 1000 took 50 minutes to compute. This meant if it were run it on the third of the planned data we could estimate it at around 80 hours and all of the planned data (372939 time steps) it would take about 310 hours. It's quite clear that this would take too long to compute.

Because of these issues a couple of changes were made to the model, as previously mentioned we collapsed the Viterbi table to save space, and altered the algorithm to only consider one state change at a time.

With the change of only considering one state change at a time it made the algorithm much more efficient having the above test last for only 30 minutes, rather than the estimated 310 hours. The only downside to only considering one device to change state at a time is the rare occurrence when two devices change at once; so far there has been no evidence of this in tests. These comparisons can be seen in figure 4.6.

Test type, with 648 states	t = 1000	t = 100000	t = 372939
Multiple state changes	50 minutes	~100 hours	~310 hours
One state change	< 5 seconds	~10 minutes	~30 minutes

Figure 4.6 Table showing runtime on different size input, t = number of time steps.

The collapsing of the table did well to save space now being able to run the test mentioned above on the full 372939 size input, this issue occurred again at around 8000 states, but it was suitable for the rest of the tests undertaken. A possible way to remedy this is to collapse the backpointer table as well and save it to file every time it is filled.

4.2. Results after changes to the framework

Once the framework was made and adequate changes were made to fix performance and accuracy issues, a number of tests were performed. Firstly every device was put into the FHMM by itself to see how accurate it is to find one device. Combinations of devices were then tested on. The combinations chosen were carefully chosen to be devices that overlap or have similar power levels.

4.2.1. Detecting a single devices

The first set of tests done was to check if one device by itself could be detected, the model would just consists of the single device and the observation input was the second half of that devices circuit reading.

The performance measure from this test looks at how often the correct state is detected, then further on how much the correct 'on' state is detected, the reasoning for this is devices are more likely to be in the off state which contains a close to zero usage level, so this should never be incorrectly detected as an on state. Further than this the results look at the min and max error in the inferred observation and the mean of the error.

The table below in figure 4.7 shows all these results for each device in one of the houses. The general correctness of the chosen states is quite high. The devices that have less accuracy all have states that use a very low amount of power the BathGFI for example has an off state with 0.9W usage with a 0.2W standard deviation and the smallest on state is 12W, but with a 10W standard deviation. You can see these values in the appendices in chapter 7. These values result in an overlapping Gaussian of the two states and it is expected that they will be incorrectly assigned. The devices showing a lower accuracy indicate that they may need to be modelled again and more carefully. However the downside to remodelling the devices is that the state space will have to be increased as well to accommodate for the current errors.

Device	Correct (%)	Found when on (%)	Min Error (W)	Max Error (W)	Error Mean (W)
Oven	98.604%	99.984%	0.04	3338.30	1.71
Fridge	99.010%	96.305%	0.01	229.01	2.14
Dishwasher	99.809%	99.845%	0.02	1088.60	1.95
Washing Machine	100.000%	100.000%	0.02	187.98	0.56
Dryer	99.999%	100.000%	0.01	2074.80	0.59
Kitchen1	97.528%	97.481%	0.18	6.54	0.68
Kitchen2	100.000%	100.000%	0.06	46.93	0.19
Kitchen3	98.509%	99.962%	0.02	69.71	0.14
Lights1	99.254%	98.570%	0.09	67.76	1.75
Lights2	99.955%	99.998%	0.00	35.88	0.54
Lights3	99.296%	99.930%	0.01	21.38	0.46
BathGFI	93.295%	100.000%	0.04	319.26	0.93
Electric Heating	99.393%	100.000%	0.00	6.87	0.06
Microwave	99.423%	99.981%	0.04	621.72	0.86
Stove	99.338%	99.725%	0.01	558.86	0.15

Figure 4.7 Accuracy of finding correct state and size of errors for single devices

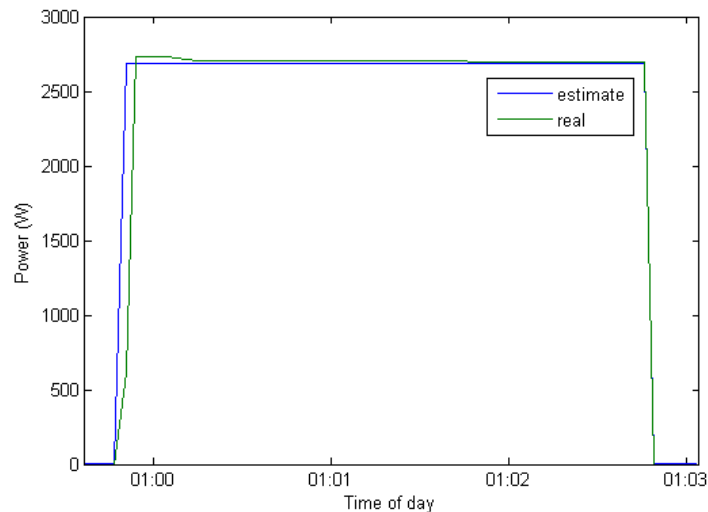


Figure 4.8 Cycle of the dryer to showing the max error occurring

The actual size of the errors is caused by a couple of issues. One the main issues is during these tests happened during the times when a device is turned on or off, rather than it going straight to the state usage level it may instead have one point halfway between the on and off. This was a very rare occurrence in the data, in figure 4.8 above you can see the dryer and when the max error in the inferred usage occurs. This happens right at the start of the cycle, you may be able to notice that the inferred line (blue) reaches the state level slightly earlier, at this same point you can see the real usage having a point around 500W. A likely cause of this may be due to the readings being every three seconds, my expectation would be that if we had readings of every second we would see this sort of behaviour at every instance of use.

The max errors in all of the other large devices occurred in the same way as the dryer, but the errors the size of the max error only occurred once in the entire test. Considering the next highest error in the large devices would give a max around half of the original. In order for us to change this, we would need to remodel the devices similar to fixing the general accuracy, the issue with this is that the energy readings that occasionally appear between the state transitions aren't all the same. For example in the dryer this state of 500W occurred once, the other errors that were similar to this were around 1000W or 2000W, each occurring once, if we were to update the model to include these levels it would be much more complex and we would not be able to run larger tests. Some of the devices also had different states active in the second half of the data that weren't active in the first half, this attributed to less accuracy and larger errors, as a way to resolve this some of these devices were modelled on the second half and tested on the first, however not every device was like this.

4.2.2. Combining devices with ground-truth labels

The second round of tests performed consisted of taking subsets of the devices with the observation being the aggregate of their respective circuit readings. These tests were designed to see how well it detects devices when other devices are active as well. The combinations of devices were carefully chosen so there would be conflicts, looking for devices with similar energy signals that might get confused or looking for devices that are active at the same time.

The table over the page in figure 4.9 shows the accuracy of the 7 tests performed. Tests 1,4, 5 and 7 were chosen as devices that may be used at once. While tests 2 and 3 were chosen as the devices have similar energy signals. Test 6 on the other hand was chosen as a small device is active at the same time as a large device as a way to check if they can both be detected, this test will be further explored in section 4.2.3.

While the accuracy of finding the correct state is high in these tests, it isn't that meaningful as most of the states of a device will be off and using no energy, the accuracy of finding the correct active state however is shown to drop quite significantly when multiple devices are present. For the tests aimed at devices with similar energy signals the accuracy is quite low for some devices, this is due to how close each device signal is. In test 2 the accuracy for the BathGFI is 40%, this would mean that 60% of the time when the BathGFI is active the model instead thinks the Microwave is being used, vice versa for the times when it doesn't detect the microwave being on. This causes the errors in the inferred usage for these devices to be quite large as well, about as equal as that devices biggest state.

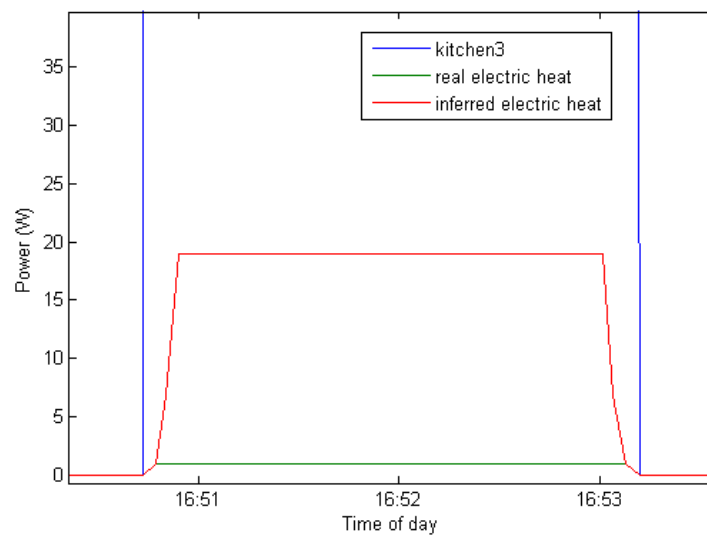
Test 4 with the Washing Machine and Dryer seems to do the best, these devices were chosen as they are often used at the same time and they are very distinct both having two states with their active usages being completely different. As these devices are so different it is expected that they would do just as well as if they were separate.

Test 7 was performed as a combination of test 1 and test 5. Test 1 and 5 show a decrease in the accuracy originally, then when we combine them the accuracy is even lower, showing that as more devices are added to the model the lower the accuracy. Kitchen2 in these tests proves to have amazing accuracy even when combined with other devices. The Kitchen2 model similar to the Washing Machin and Dryer only has two states are the active state is nowhere near the other states of the devices in test 7 so it was expected that Kitchen2 would have the highest accuracy.

This confusion found between devices with similar power levels is quite an issue with disaggregation, in order to find a more distinctive difference between two similar devices more information is needed, such as outside parameters that affect the usage of device. These might be the current weather conditions, or the time of day or season, anything that really affects people using certain devices.

Device Test	Correct (%)	Found when on (%)	Min Error (W)	Max Error (W)	Error Mean (W)
1					
Oven	99.859%	95.545%	0.04	3,338.30	1.65
Stove	97.727%	94.869%	0.01	558.86	0.26
2					
BathGFI	97.702%	39.618%	0.04	1,643.00	2.85
Microwave	99.640%	78.878%	0.04	1,619.00	3.97
3					
Kitchen3	99.748%	78.866%	0.02	1,589.00	3.96
Microwave	99.539%	76.103%	0.04	1,578.70	5.45
4					
Washing Machine	99.998%	100.000%	0.02	441.98	0.57
Dryer	100.000%	100.000%	0.01	1,222.50	0.58
5					
Kitchen1	96.040%	95.993%	0.18	11.18	0.75
Kitchen2	100.000%	100.000%	0.94	1,068.10	32.23
Kitchen3	98.298%	64.351%	0.02	1,518.30	19.58
6					
Electric Heating	99.851%	99.301%	0.00	18.96	0.04
Kitchen3	98.929%	92.774%	0.02	69.71	0.15
7					
Kitchen1	94.634%	95.424%	0.18	11.18	0.82
Kitchen2	100.000%	100.000%	0.06	1069.10	0.19
Kitchen3	96.997%	34.186%	0.02	1514.30	0.30
Stove	97.998%	69.058%	0.01	1404.30	0.28
Oven	98.702%	56.464%	0.04	3338.30	1.78

Figure 4.9 Accuracy of the test run on multiple devices



Below: Figure 4.10 Confusion of electric heat being on when it is not.

4.2.3. Detecting small usage devices amongst big devices

Test 6 from figure 4.9 as mentioned was chosen to see if the model can find a small usage device amongst a big signal, the small device being the Electric Heat may have high accuracy but the actual error found in the inferred usage is around the same as the energy signal for the device. This suggests that the device is not found one of the times where it is turned on, or it is mistaken as on when there is large noise from the device.

Figure 4.10 on the previous page shows the time of when the largest error in the heat occurs. The green line indicates the actual usage of the Electric Heat, the red is it's inferred usage, it's quite clear that this point has been assigned the wrong state. The cause of this is from the Kitchen2 device which is active at the same time, represented as the blue lines. The standard deviation of Kitchen2 in it's on state is 17.5W which encapsulates most of the Electric Heat device, this confusion suggests that the noise during the time that Kitchen2 is active is larger than the Electric Heat signature and so it is assumed to be active as well.

Another way these devices may have been confused is if the noise from Kitchen2 at the time the Electric Heat was active was quite low, then the model would assume the Electric Heat signal was completely part of the Kitchen2 signal, however this scenario did not occur in the test.

The method introduced for us to fix this issue and properly detect small devices requires us to separate small usage devices from big usage devices. The model is run on first the big devices, then the small devices while ignoring when the big devices were turned on. Below in figure 4.11 is an example of the output of this method the graph on the left shows the output before the splitting method is used, having inferred usages at the same time. The graph on the right shows after the splitting method is used, the Washing Machine is considered to be a small device and is run separately on the same input ignoring the times when the Dryer is turned on.

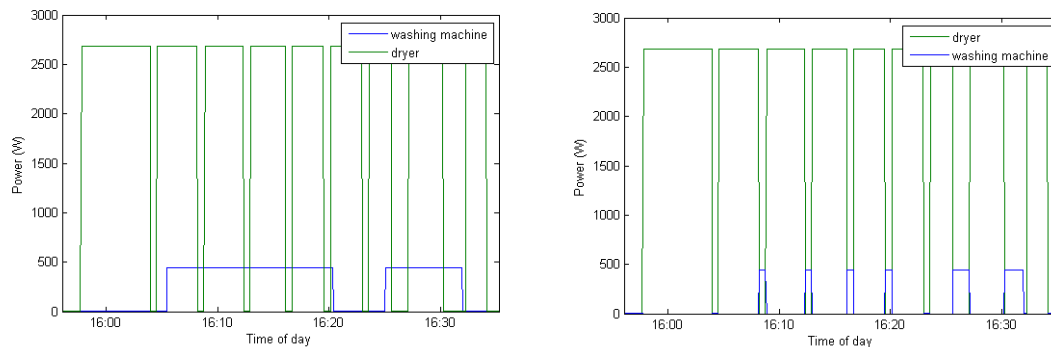


Figure 4.11 Example of before and after implementing the splitting method. Devices used are the washing machine and dryer.

To show that this method increase the accuracy of small devices it was run on tests 4 and 6 from figure 4.9. The table over the page in figure 4.12 contains the accuracy from these tests. Note that the accuracy calculated for these tests only consider the non-skipped time steps. It is quite clear that without the big devices interfering the small device gets detected with no problem, and the errors are shortened.

Device	Correct (%)	Found when on (%)	Min Error (W)	Max Error (W)	Error Mean (W)
Heat	97.56%	100.00%	0.003	7.875	0.209
Kitchen3	100.00%	100.00%	0.000	69.714	12.735
Washing Machine	100.00%	100.00%	0.003	172.980	0.422
Dryer	99.97%	100.00%	0.003	2074.800	28.793

Figure 4.12 Accuracy and Error size from tests 4 and 6 with splitting method implemented.

Further tests with a larger set of devices were performed as well. These tests were done using the same subset of devices but with a different energy level to split the big and small devices on. The previous tests have only looked at splitting the devices above and below a 1000W reading. These last few tests use different levels for this split ranging between 50W to 2000W.

The eight devices chosen for this test were: Dryer, Washing Machine, Kitchen1, Kitchen2, Lights1, Lights2, Electric Heat and the Oven. There were 6 tests done each with different levels for the split. The 6 tests were split as follows:

Split level = 0W – All devices considered big

Split level = 50W – Dryer, Washing Machine, Kitchen2, Oven, Lights1 & 2 considered big, other two devices are small.

Split level = 100W – Dryer, Washing Machine, Kitchen2, Oven, Lights1 Considered big, three others devices are small.

Split level = 300W – Dryer, Washing Machine, Kitchen2, Oven considered big, other four are small.

Split level = 1000W – Dryer, Kitchen2, Oven are considered big, other 5 are small.

Split level = 2000W – Dryer, Oven considered big, other 6 are small.

The results of these tests can be found below in figure 4.12. The results show that the choice of the split level has a definite change of the accuracy of the model. On first inspection there doesn't seem to be a level that works best for every device. With the exception of the Electric Heat every device performs best when there is no split between the devices. There appears to be a large variation as the size of the split is changed. This could be the cause of different devices interacting with one another and it only happens when the devices are in the same side of the split.

Dryer Accuracy		Kitchen1 Accuracy		Kitchen2 Accuracy		Electric Heat Accuracy	
0	100.00%	0	62.76%	0	100.00%	0	59.22%
50	99.75%	50	16.69%	50	100.00%	50	96.92%
100	99.35%	100	32.37%	100	100.00%	100	59.55%
300	99.99%	300	51.50%	300	99.98%	300	55.32%
1000	78.47%	1000	61.41%	1000	98.35%	1000	55.14%
2000	73.94%	2000	61.56%	2000	100.00%	2000	55.10%

Washing Machine		Oven		Lights1		Lights2	
0	100.00%	0	99.29%	0	86.51%	0	83.69%
50	99.74%	50	95.83%	50	26.66%	50	5.02%
100	96.47%	100	48.35%	100	18.62%	100	60.99%
300	53.32%	300	68.37%	300	81.51%	300	79.60%
1000	100.00%	1000	16.55%	1000	81.13%	1000	78.56%
2000	100.00%	2000	20.06%	2000	81.15%	2000	78.72%

Figure 4.13 Set of tables for each device used in the final tests, shows accuracy of inferring the correct state given the level of split.

The graph below in figure 4.14 shows the relation between split level and accuracy of the device. There is no obvious link between the two variables, showing large variation of the accuracy at each split level. It's possible some of the errors found at certain levels is a cause of devices interfering with each other. For example in the 50W test only the Electric Heat and Kitchen1 are considered small devices, these devices have a very similar output (see Appendix, chapter 7) and may easily be confused. Further tests when more devices are added to the small device set may then interfere with the Electric Heat and suggest that Kitchen1 is being active more. To properly look into this however more work will be needed. Further tests with more devices and more levels to split the devices on would be required as part of this work.

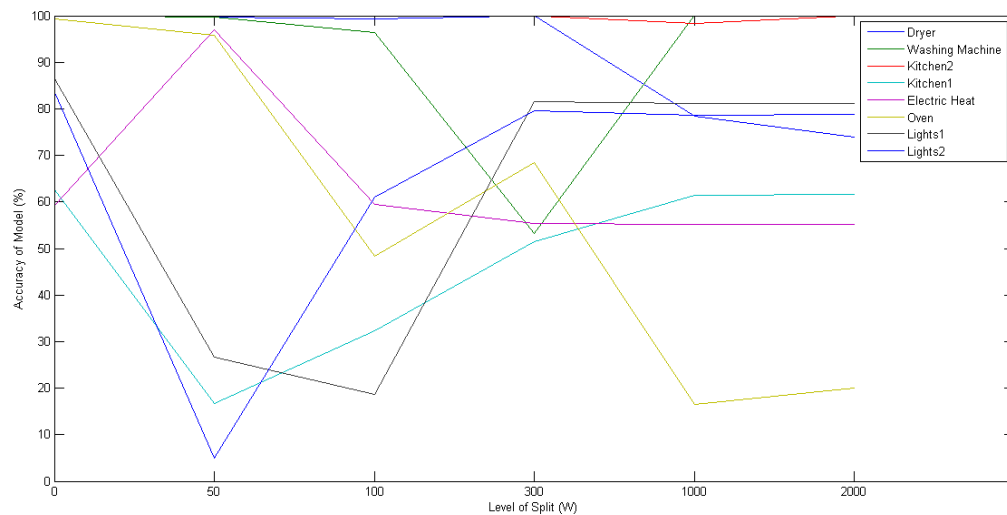


Figure 4.14 Plot of the accuracy of each device across each level of split that was used.

Chapter 5: Conclusion

5.1. Summary and contribution

Non-intrusive load monitoring and energy disaggregation have shown to be quite beneficial in aiding our understanding how we use electricity and gives insights on how we can save energy. With obvious benefits being able to save money on power bills, alternate uses also include monitoring for faults in devices or even monitoring for criminal activity. While there are many methods for performing energy disaggregation the focus of this research has been through the use of Hidden Markov Models.

This research project provides a framework for the use of HMMs with the expected use with energy disaggregation. This study has found many issues regarding the complexity of inference methods used for HMMs and provides methods with aiding this, the highlight being the idea to split devices based on their energy signals to help identify small usages devices when noise from large devices exists. Other methods introduced to increase efficiency have been to simplify the state space of devices, and to assume only one device is able to change state at a time.

The results from the experiments showed generally that the more complex the model is the less accuracy of predicting states, and the longer computation time is required. It is also shown through the benchmarking results that methods of disaggregation can be too simple require more complexity. The use of the splitting method showed it was easier to detect smaller devices when ignoring times when a big device is active, along with this the splitting method increased the efficiency of the model by simply splitting the input.

5.2. Future work

This Research project has found many issues surrounding current approaches to Energy Disaggregation and as a result has inspired many future areas of research in this area.

While the complexity issues seemed to cause an explosion in the state space of the FHMM, future studies could look into how the number of devices affects this and the runtime of inference techniques with HMMS.

Future research in the complexity issues can also be undertaken while looking at how much information is needed to describe a device. The research would explore how to find an optimal number of states to encapsulate the behaviour of the device while not risking the accuracy of the inference methods for the model.

We could also further look into the idea of only one device changing state at a time and abstract it to saying that 'k' devices can change state at once, exploring how the amount of devices changing state may affect the accuracy of the model, and should almost definitely affect the runtime.

The use of filtering the data as a pre-processing step introduces the idea of whether the random spikes in the energy signals was actually random. Research in this may be to see if these random spikes are part of some cycle within the device usage. This research however would require a lot more data which would greatly increase the complexity of any inference performed.

Another research area we can use to expand on this project, which was the original plan of this paper before the complexity issues were found, is use of prior knowledge of devices in the model and how it may or may not affect the accuracy of the inference performed. Prior knowledge could be time of day/year, climate data or even the correlations between devices, i.e. when a washing machine is used the dryer is used shortly after. This requires the use of an extension to the

FHMM known as Conditional Factorial Hidden Markov Model (CFHMM) or Factorial Hidden Semi Markov Models (FHSMM) if the use of dependencies between devices is present, or even both as a CFHSMM.

The biggest contribution of this study was the introduction of splitting devices into the two groups for big and small devices we may be able to look into how to choose an optimal diving line for the split between this groups, as a split too high may not be necessary if it splits two very distinct devices and a split too low may ignore the use of this divide. Alternatively we could look at if creating a third group of medium usage devices affects the complexity and accuracy of the inference techniques. At the end of chapter 4 a few tests were performed to see how the level used to split big and small devices can affect the accuracy. This didn't show any direct relation but I suspect more work will be needed to find any possible relation and possible the most optimal split level. Further work on these tests will require more devices to allow for more splits and to more closely research the relationship between devices on the same network as this could be the cause of the large variation of accuracy found in the final tests.

Chapter 6: References

- [1] G. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870-1891, 1992.
- [2] S. Darby. The effectiveness of feedback on energy consumption. Technical report, environmental Change Institute, University of Oxford, 2006.
- [3] J. Z. Kolter, M. J. Johnson. REDD: A Public Data Set for Energy Disaggregation Research. *SustKDD*, August 2011.
- [4] A. Zoha, A. Gluhak, M.A. Imran, S. Rajasegarar. Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey. In *Sensors* Vol.12(12), pp.16838-66, 2012.
- [5] G. Hart. Residential Energy Monitoring and Computerized Surveillance via Utility Power Flows. *IEEE Technology and Society Magazine*, 8(2), 12-16, 1989.
- [6] M.M. Eissa, S.M. Wasfy and M.M. Sallam. Load Management System Using Intelligent Monitoring and Control System for Commercial and Industrial Sectors. Helwan University at Helwan-Department of Elect. Eng., Cairo, Egypt, 2012.
- [7] M. Ziefman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics*, 57(1):76–84, 2011.
- [8] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. In *Proceedings of the SIAM Conference on Data Mining*, 2011.
- [9] H. Najmeddine, K. El Khamlichi Drissi, C. Pasquier, C. Faure, K. Kerroum, A. Diop, T. Jouannet, M. Michou, State of Art on Load Monitoring Methods. In *Proceedings of the 2nd IEEE International Conference on Power and Energy Conference*, 2008 pp. 1256–1258.
- [10] S. Gupta, M.S. Reynolds, S.N. Patel. ElectriSense: Single-Point Sensing Using EMI for Electrical Event Detection and Classification in the Home. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, 2010, pp. 139–148.
- [11] A.I. Cole, A. Albicki. Data Extraction for Effective Non-Intrusive Identification of Residential Power Loads. In *Proceedings of Instrumentation and Measurement Technology Conference*, 1998, Vol(2), pp. 812–815.
- [12] R. Cos, S.B. Leeb, S.R. Shaw, L.K. Norford. Transient Event Detection for Nonintrusive Load Monitoring and Demand Side Management Using Voltage Distortion. *Twenty-First Annual IEEE Applied Power Electronics Conference and Exposition*, 1751-1757, 2006.
- [13] M. Berges, L. Soibelman, H.S. Matthews. Leveraging data from environmental sensors to enhance electrical load disaggregation sensors. In *proceedings of the International Conference on Computing in Civil and Building Engineering*, June 2010.
- [14] L. R. Rabiner, B. H. Juang. An introduction to hidden markov models. *Proceedings of the IEEE*, 3(1): 4-16, 1986.
- [15] D. Srinivasan, W. Ng, A. Liew. Neural-network-based signature recognition for harmonic source identification. *IEEE Trans. Power Del.* 2006, 398–405.
- [16] A.G. Ruzzelli, C. Nicolas, A. Schoofs, G.M.P. O’Hare. Real-Time Recognition and Profiling of Appliances through a Single Electricity Sensor. In *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, June 2010 pp. 1–9.

- [17]Z. Ghahramani. An Introduction to Hidden Markov Models and Bayesian Networks. In International Journal of Pattern Recognition and Artificial Intelligence, 15(1), 9-42, 2001.
- [18]Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. Machine Learning, 29(2-3):245-273, 1997.
- [19]J. Z. Kolter, T. Jaakkola. Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation. In proceedings of the 15th international conference on Artificial Intelligence and Statistics, 2012.
- [20]L. Rabiner. A tutorial on hmm and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257-286, February 1989.
- [21]J. Besemer, M. Borodovsky. GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses, in Nucleic Acids Research, 2005, Vol.33, 451-454
- [22]H. Zhang, W. Zhang, A. Palazoglu, W. Sun, Prediction of ozone levels using a Hidden Markov Model (HMM) with Gamma distribution, in Atmospheric Environment, 2012, Vol.62, pp.64-73.
- [23]A. Zoha, A. Gluhak, M. Nati, M.A. Imran. Low-Power Appliance Monitoring using Factorial Hidden Markov Models, In IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, April 2013, pp.527-532
- [24]L. Wang, X. Luo, W. Zhang. Unsupervised Energy Disaggregation with Factorial Hidden Markov Models Based on Generalized Backfitting Algorithm, in IEEE International Conference of IEEE Region 10, Oct. 2013, pp.1-4.
- [25]J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281-297, 1967.
- [26]K. E. Barner, G. R. Arce. Order-Statistic Filtering and Smoothing of Time-Series: Part II. In C. R. Rao and N. Balakrishnan, Handbook of Statistics, vol 16. Elsevier Science B.V. 1998.

Chapter 7: Appendices

The following is a list of the devices used for tests in this study.

It contains the device label the number of state and the mean and standard device of the power signal from each state.

Washing Machine

State	1	2
Mean Power (W)	0.06	441.98
sd of Power	1.95	46.15

Dryer

State	1	2
Mean Power (W)	0.01	2687.28
Sd of Power	2.04	38.89

Kitchen2

State	1	2
Mean Power (W)	1.06	1070.07
Sd of Power	0.47	17.51

Oven

State	1	2	3
Mean Power (W)	0.05	12.54	4003.29
Sd of Power	0.45	4.67	242.71

BathGFI

State	1	2	3
Mean Power (W)	0.96	12.74	1603.28
Sd of Power	0.26	10.24	20.52

Kitchen1

State	1	2	3
Mean Power (W)	13.82	20.69	23.46
Sd of Power	1.54	0.80	1.13

Kitchen3

State	1	2	3
Mean Power (W)	0.02	4.82	1519.29
Sd of Power	0.19	1.39	31.62

Electric Heating

State	1	2	3
Mean Power (W)	0.00	7.87	19.96
Sd of Power	0.06	1.60	0.36

Microwave

State	1	2	3	4
Mean Power (W)	4.04	44.27	265.28	1518.51
Sd of Power	0.31	13.67	94.84	46.43

Stove

State	1	2	3	4
Mean Power (W)	0.01	6.67	20.18	1420.86
Sd of Power	0.13	1.29	10.47	50.42

Dishwasher

State	1	2	3	4	5
Mean Power (W)	0.04	38.02	217.14	423.80	1089.62
Sd of Power	1.38	15.57	11.65	74.18	76.40

Fridge

State	1	2	3	4	5
Mean Power (W)	6.49	46.16	191.43	214.89	423.31
Sd of Power	0.54	3.84	5.29	11.37	5.06

Lights1

State	1	2	3	4	5
Mean Power (W)	1.43	28.28	81.16	168.79	282.71
Sd of Power	0.81	2.31	1.95	10.65	21.93

Lights2

State	1	2	3	4	5
Mean Power (W)	0.00	7.84	31.88	64.88	97.57
Sd of Power	0.08	1.14	2.59	2.53	2.60

Lights3

State	1	2	3	4	5
Mean Power (W)	1.01	9.48	44.67	56.57	68.62
Sd of Power	0.14	1.70	1.90	1.13	1.25