

Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE A Framework for Estimating Energy Consumed by Electric

Loads Through Minimally Intrusive Approaches

PRESENTED BY Suman Giri

ACCEPTED BY THE DEPARTMENTS OF

Civil and Environmental Engineering

Mario Berges

ADVISOR, MAJOR PROFESSOR

April 27, 2015

DATE

David A. Dzombak

DEPARTMENT HEAD

April 30, 2015

DATE

APPROVED BY THE COLLEGE COUNCIL

Vijayakumar Bhagavatula

DEAN

May 4, 2015

DATE

**A framework for estimating energy consumed by electric
loads through minimally intrusive approaches.**

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Advanced Infrastructure Systems

Suman Giri

B.A., Physics, Mathematics, Oberlin College

M.S., Civil Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA
May, 2015

©2015 Suman Giri. *Some rights reserved.* Except where indicated, this work is licensed under a Creative Commons Attribution 3.0 United States License. Please see <http://creativecommons.org/licenses/by/3.0/us/> for details.

The views and conclusions contained in this document are those of the author, and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

Keywords: non-intrusive load monitoring, NILM, energy estimation, virtual sensing, finite state machines, error correction.

Abstract

This dissertation explores the problem of energy estimation in supervised Non-Intrusive Load Monitoring (NILM). NILM refers to a set of techniques used to estimate the electricity consumed by individual loads in a building from measurements of the total electrical consumption. Most commonly, NILM works by first attributing any significant change in the total power consumption (also known as an *event*) to a specific load and subsequently using these attributions (i.e. the labels for the events) to estimate energy for each load. For this last step, most proposed solutions in the field impart simplifying assumptions to make the problem more tractable. This has severely limited the practicality of the proposed solutions. To address this knowledge gap, we present a framework for creating appliance models based on classification labels and aggregate power measurements that can help relax many of these assumptions. Within the framework, we model the problem of utilizing a sequence of event labels to generate energy estimates as a broader class of problems that has two major components (i) With the understanding that the labels arise from a process with distinct states and state transitions, we estimate the underlying Finite State Machine (FSM) model that most likely generated the observed sequence (ii) We allow for the observed sequence to have errors, and present an error correction algorithm to detect and correct them. We test the framework on data from 43 appliances collected from 19 houses and find that it improves errors in energy estimates when compared to the case with no correction in 19 appliances by a factor of 50, leaves 17 appliances unchanged, and nega-

tively impacts 6 appliances by a factor of 1.4. This approach of utilizing event sequences to estimate energy has implications in virtual metering of appliances as well. In a case study, we utilize this framework in order to substitute the need of plug-level sensors with cheap and easily deployable contactless sensors, and find that on the 6 appliances virtually metered using magnetic field sensors, the inferred energy values have an average error of 10.9%

आमाको लागि ...

Acknowledgments

Although, and I presume for practical reasons, this thesis bears my name alone, it is a byproduct of hundreds of helpful discussions, and thousands of words of encouragement and prayers from those who willed me on. So, I would be remiss if I did not mention, at the very start, that this contribution- however insignificant it maybe- is as much theirs as it is mine.

I owe much gratitude to my advisor, Dr. Mario Bergés, for taking a huge risk by accepting me as his PhD student. Although I sometimes still question if that was right decision on his part, what I am absolutely sure of is that I could not have made a better choice in selecting an advisor. Your patience and approach to life (both personal and professional) have been contagious and inspiring, and have shaped my evolution as a human being and researcher through these years. So, thank you for being there, and for being awesome.

I am also indebted to my committe members (Dr. Lucio Soibelman, Dr. Chris Hendrickson, Dr. Artur Dubrawski, and Dr. Mario Bergés) who are all extremely busy people in their own rights, but who always made the time to give me valuable feedback when I needed guidance. Without their experience, and helpful suggestions during my proposal, this thesis would not have materialized. I would also like to acknowledge the role Dr. Soibelman, Anu dai, and Adrian played in me selecting Carnegie Mellon as my grad school of choice- guidance for which I am forever thankful. People from industry including Sean and Mark from Samsung Electronics, and Sri and Marco from SmartB have also helped

me at various parts of my tenure with invaluable feedback, and I would like to thank them. A shoutout also to Samrachana for her statistics input, Patrick and Niranjini (and Anthony's lab) for hardware support, Dr. Zico Kolter and Farrokh for generously sharing their data, and Jingkun and Emre for help with miscellaneous (but important) bits. I am also indebted to Kyle, Farrokh, and Lucas for various helpful discussions on NILM related topics, and colleagues at Inferlab for listening to my presentations and giving feedback.

My work was supported at various points by Samsung Telecommunications America, and the NSF. In addition, the Bertucci fellowship and Dean's fellowship also sponsored part of the research work. I would like to thank them all. Grad school, now I can safely say, is a long and taxing process, both emotionally and intellectually, and -as the Beatles so eloquently put it- *I got by with a little help from my friends*; and more than just a little at times. Thanks to Darshana for being with me through the ups and downs, you have shaped my life in ways you'll never know. Thanks to Amsul, Regeant, Prabhat, Sariph, Samrachana, Ashish, Samikshya, Ankit, Sona, Ichhuk, Isha, etc. for ensuring I had some semblance of a social life through these years. Thanks also to Prativa and Maya for being more excited about my impending graduation than I could ever be. Thanks to Shimon, Amy, and Steve and family, for being my mentors/ adopted-family-members. Thanks to my mom who has always been my biggest fan and my source of inspiration, and to both my parents for having complete faith in all the decisions I made. Finally, thanks to my younger brother, because why not.

Contents

1	Introduction	1
1.1	Non-Intrusive Load Monitoring	1
1.2	The need for appliance models in NILM	7
1.3	The need for error correction	9
1.4	The case for virtual metering	13
2	Motivating case studies	16
2.1	Case studies on appliance models	16
2.1.1	Formalizing the energy estimation process	17
2.1.2	A more general view of the problem	19
2.1.3	Energy estimation in supervised settings without appliance models	21
2.1.4	Energy estimation in unsupervised settings without accurate appliance models	22
2.1.5	Energy estimation with basic appliance models	25
2.2	Case studies on Virtual Metering	27
2.2.1	System Hardware	28
2.2.2	System Software	31

2.2.3	System Evaluation	34
3	An Energy Estimation Framework	40
3.1	State of the art in appliance behavior modeling	41
3.1.1	Finite State Machine formulations	42
3.1.2	Probabilistic prior models for appliances	43
3.1.3	Other ways of modeling appliances	44
3.2	Introduction to the framework	45
3.3	Classifying distinct state transitions	49
3.4	Perturbance	57
3.5	Creating transition probability matrices	61
3.5.1	Finding feasible cycles	61
3.6	Creating state transition models	62
3.7	Correcting errors	64
3.8	Computing energy	65
3.8.1	Further evaluation with simulated data	67
3.9	Discussion	69
3.9.1	Discussion of results	69
3.9.2	Discussion of framework	70
3.9.3	Evaluation criteria	74
3.10	Chapter Conclusion	74
4	An Error Correction Framework	77
4.1	Introduction to the framework	78
4.2	Shortest-path formulation	82
4.3	Evaluation	84

4.3.1	Evaluation metrics	85
4.3.2	Evaluation on Simulated Data	87
4.3.3	Evaluation on real data	92
4.4	Discussion	102
4.5	Chapter Conclusion	105
5	Virtual metering of electrical appliances	107
5.1	Literature Survey	108
5.1.1	Direct metering techniques	108
5.1.2	Indirect metering techniques	109
5.2	Framework for analyzing sensor data	111
5.3	Experiment and Results	113
5.4	Discussion	115
5.5	Chapter Conclusion	123
6	Summary	124
6.1	Summary and Broader Impact	124
6.2	Future Work	125

List of Figures

1.1	Break-down of annual energy savings by type of feedback. This graph, adapted from [1], is based on the results of 36 studies that looked into potential energy savings based on different kinds of feedback provided to consumers.	3
1.2	Overall power consumption of a residential building during a twenty minutes interval. Some appliances change states quickly during operation (like the stove burner). By extracting appropriate features from these power signatures, appliance types can be classified.	5
1.3	Steps involved in supervised NILM. In this thesis, we explore the step of energy estimation by assuming results of the previous steps are already available	6
1.4	FSM diagram for a refrigerator from BLUED-1 (left), tv from BLUED-1 (center) and laptop from BLUED-2 (right).	8
1.5	A generic communication model for a sequential data stream as it gets corrupted on passing through a channel.	12

1.6	Ground truth power trace for Refrigerator extracted from a public dataset (BLUED [2]) plotted against a power trace reconstructed using event labels. As can be seen, missing a couple of events results in a reconstructed trace that is very different from ground truth.	15
2.1	Section of a power trace reconstructed using event labels at mains for refrigerator (above) and TV (below)	21
2.2	Section of a power trace reconstructed using event labels and appliance models at mains for refrigerator, TV and computer	26
2.3	Data extracted for training HMM models for a dishwasher in House 4 for REDD from the aggregate using different priors (above), and the ground truth for dishwasher as measured at circuit level (below). The newer model is capable of extracting the right segment for learning, contributing to better energy estimation results	27
2.4	Network Architecture	30
2.5	EMF Event Detector Waveforms. Top: Ceiling fan with light switch activated at point (a), manually turned on at point (b), and turned off at point (c). Bottom: Desktop computer is an example of a noisy signal due to switching.	31
2.6	EMF event detector stacked on FireFly3 sensor node.	32
2.7	Experimental Setup	33
2.8	Piecewise appliance energy estimator running on a refrigerator. Thick vertical lines at bottom show EMF event inputs onto the mains power waveform. The bars across the top represent regions where power is estimated to be constant.	34

2.9	EMF sensor event detection confusion matrices. Total event count in parentheses after appliance name.	36
2.10	Energy estimation performance	38
2.11	Error vs Mains sampling window	38
3.1	The input and output associated with the framework proposed in this Chapter.	41
3.2	Summary of the major steps involved in the energy estimation framework presented in the chapter. Steps B and C, and steps E and F can be carried out in parallel, as their inputs are independent of each other.	48
3.3	Weighted sum of errors in predicting the number of clusters, and the values for exemplars for state transitions for different algorithms. . . .	53
3.4	RMS error evaluated on simulated data for different clustering algorithms.	54
3.5	Clustering results on P-Q features using affinity propagation for the Refrigerator and AV system on BLUED. The labels show the mean ΔP values of each cluster. The histograms show the frequency of ΔP 's associated with the appliances.	55
3.6	State-transition diagrams for a Refrigerator (left) and TV (right) learnt from the time series of events.	64
3.7	FSM diagram for a Refrigerator from BLUED-1 (left), TV from BLUED-1 (center) and Laptop from BLUED-2 (right) created using adjacency matrices	65

3.8	Flow of data through the framework pipeline for the refrigerator in BLUED-1. For illustration purposes, we pick a segment between 18:00 and 20:00, and note the actual values of C_{seq} , C'_{seq} and X for that portion only. The cluster number highlighted in red shows the error that was corrected (from transition 3 to 4) in step F.	68
4.1	The input and output associated with the framework proposed in this Chapter.	78
4.2	Network flow diagram for the sequence $C_{seq} = \{c_1 \dots c_m\}$. Three different kinds of error corrections are possible: Substitution, Insertion and Deletion.	82
4.3	DTW cost matrix for two sequences (original and corrupted) for a washer (label: 3) in dataset REDD-1. The appliance only has two states, and hence the sequence fluctuates between two values.	86
4.4	Path (shown in white) with minimal cost of alignment for the original sequence and corrupted sequence as shown in Figure 4.3. The DTW distance was 5808.	86
4.5	Median error in energy estimation (e) for varying values of μ and ν , with $\gamma = 16$. The medians were calculated after 100 simulations of each cost combination.	91
4.6	Standard Deviations for the median errors as reported in Figure 4.5 for 100 iterations of different cost combinations.	92

4.7	Box plots of logarithmic values of the energy estimation error: \mathbf{e} (top), DTW distance of corrected sequences (middle), and RMS error of reconstructed power traces (bottom), with box margins representing the 25th and 75th percentile values, and median was calculated after 1000 simulation trials. EC stands for Error Correction.	93
4.8	The inputs and outputs involved in the evaluation process when the error correction is applied on BLUED and REDD datasets. Energy estimation framework refers to the framework from Chapter 3, and Error correction framework is the one proposed in this chapter. . . .	95
4.9	A log-log plot of the change in energy estimation error upon using the framework (y-axis) and the energy estimation error without the framework (x-axis) for 43 appliances.	97
4.10	A log-log plot of the energy estimation error versus cluster quality as measured by Dunn Index. Each dot represents appliances in the same order as they appear in Table 4.3.	99
4.11	Change in energy estimation error (logarithmic) as the number of clusters were varied for a sample set of appliances from the dataset. The red circle denotes the number of clusters and error value the algorithm converges to if it is not forced to assume a certain number of clusters.	99
5.1	The vision for a contactless appliance-level energy metering setup. The sensors log state changes in appliances, and send that info to a central computational platform. The platform also receives aggregate electricity data sampled using some energy meter. Combining the two, it calculates energy for each appliance.	112

5.2	HOBO motor on/off loggers (sensors) placed next to 6 appliances for virtual metering. The red rectangle shows the location in the appliance where the sensors were placed.	114
5.3	Ground truth power traces for two cycles each of washer (top), dryer (middle), and AC (bottom) as measured by Firefly plug meters on a specific day (24 th February 2015)	117
5.4	Left (a): P and Q power trace of a refrigerator mapped in the PQ plane to extract steady states. Right(b): Simultaneous events occuring within a window (above) and transient spikes that can affect ΔP computation (below) for two event instances of a dryer.	118

List of Tables

2.1	Energy estimation errors based on a framework with and without appliance models. The basic model that is used for comparison assumes that the observed changes in power lie within the set of ΔP 's that are assigned.	23
2.2	Errors in energy estimation with (AT) and without (NT) training using generic prior models of appliance types in REDD data. The old prior models are models used by the author in [3] and the updated models are empirical models chosen by us.	24
3.1	Summary of the datasets used for evaluation in this chapter	46
3.2	Results of perturbation on state transition values extracted after clustering in different datasets	60
3.3	Actual and estimated error for estimated energy for a select group of appliances in 3 houses from different datasets. The energy was estimated using the framework presented above.	75
3.4	Models for generating simulated data for analysis. 100, week-long, power traces were generated for each appliance	76
3.5	Average error in energy estimation for simulated data for a 100 houses.	76
4.1	Summary of the parameters used in the simulation model	89

4.2	Summary of the datasets used for evaluation in the chapter. Dataset index B stands for BLUED and R stands for REDD. Appliances are indexed as follows: A-Refrigerator; B-Lights; C-TV; D-Computer; E-Laptop; F-AV System; G-Washer; H-Oven; I-Dishwasher; J-Microwave; K-Dryer; L-Furnace; M-PoolPump; N-HairDryer.	96
4.3	Results of energy estimation on 43 appliances from 19 houses with and without the error correction framework. The sequence for error correction, and subsequent energy estimation was done using the framework discussed in Chapter 3. Dataset index B stands for BLUED and R stands for REDD.	100
5.1	A summary of the appliances used in the experiment and the events associated with each appliance along with the results from energy estimation using the virtual metering framework.	115

Chapter 1

Introduction

1.1 Non-Intrusive Load Monitoring

Electricity use in residential buildings is a big contributor to the total electricity and total energy consumption, and, consequently, total fossil fuel exploitation. In the U.S., 38% of the total electricity consumption is attributed to the residential sector, which makes them contributors of around 17% to the total greenhouse gas emissions [4]. Over the years, this realization has motivated academic and industrial research in finding ways to reduce electricity consumption in residential buildings. One way of reducing electricity consumption is by giving feedback to users on their electricity consumption habits and patterns. From as far back as 1976, studies had established that such feedback would amount to sizable savings in energy consumption [5]. Multiple researchers have looked at different kinds of feedback and the extent of savings each can potentially produce [1, 6]. Figure 1.1, adapted from a meta-review of previous studies reported in [1], lays out the potential savings from different kinds of feedback mechanisms. Darby, in her review of the effectiveness of feedback in reducing consumption, has noted that direct feedback on aggregate

electricity consumption (immediate and easily visualized) can produce savings ranging from 5-15% [6]. One form of meaningful feedback about electricity consumption could be the appliance-level breakdown of electrical energy usage in a building. Numerous studies have looked into savings generated in electricity consumption based on feedback mechanisms, and real time disaggregated (appliance-level) feedback has been found to generate the maximum savings [1]. Dennis et al. have claimed that the reason feedback on aggregate electricity consumption is not as effective as it should be is because consumers do not know what each component of their electricity consumption is costing them [7]. Dobson and Griffin later studied the efficacy of real time appliance-level consumption feedback to users [8]. Based on a two-month long study on 25 households, they attributed savings of up to 12.9% to such feedback. Since then, four other studies have looked at the potential for savings from giving appliance-level energy consumption information to users and have reported savings from 9-15% [9, 10, 11, 12]. Based on these studies, Martinez et al. have estimated the expected average savings from such feedback to be around 12% [1]. The sample size and duration of these studies demand caution before the results are generalized, but the savings-potential of appliance-level feedback is promising. Such information can also be valuable for other stake-holders in the energy efficiency chain. For instance, other stake-holders like facility managers, energy auditors, and even suppliers of electricity (e.g., Electric Utilities, Independent System Operators (ISOs), energy retailers, etc.) can make informed decisions about issues like energy management, planning, generation, and feedback. In addition to feedback about usage patterns, such information also enables consumers and/or auditors to identify appliances that are not operating optimally and need to be replaced. The potential use cases can go as far as alerting users for any anomalous behavior (e.g., oven left

on), and pre-emptive fault/failure detection for appliances.

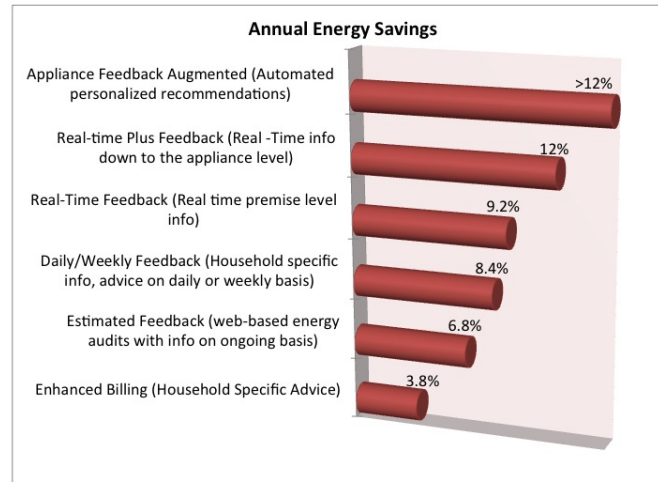


Figure 1.1: Break-down of annual energy savings by type of feedback. This graph, adapted from [1], is based on the results of 36 studies that looked into potential energy savings based on different kinds of feedback provided to consumers.

The straightforward way to obtain appliance level power consumption information is through the use of plug-through power meters, also referred to as plug-meters. Although several alternatives and easily deployable products are available in the market, as discussed in Section 5.1, they suffer from two major limitations. First one being the cost associated with these solutions. Berges et al. note that although direct metering using plug-meters is a useful way to perform energy audits, their payback periods (based on current market prices and estimated savings due to installation of such technologies) are fairly long [13]; typically, it is more than what most customers are willing to engage in. The second major limitation is that not all appliances can be monitored using plug-meters. Appliances whose plugs are typically difficult to access or those that do not have a standard two or three-pronged plugs (e.g, dishwashers, light fixtures, garbage disposals, dryers, HVAC, etc.) are

all examples where a plug-meter cannot perform the task of energy metering.

A less invasive and perhaps more cost-effective approach would be one pioneered in the 1980s: Non-Intrusive Load Monitoring (NILM) [14]. NILM refers to a set of techniques that algorithmically estimate the power consumed by individual devices in a building from measurements of voltage and/or current taken at a limited number of locations in the electrical distribution system of the premise. The *event*-based approach to solving this problem attempts to attribute statistically significant changes in the total power consumption (*events*) to changes in the operational mode of specific devices, by classifying features of the signal changes (e.g., a change in magnitude, frequency content). Figure 1.2 illustrates the initial idea proposed in [14], where step changes of power in the main power line are used to determine which appliance caused such change. Every time an appliance turns on within the house, there is a change on the overall power consumed by the building, which manifests as a step change - also known as power deltas (ΔP). Features based on these step changes, including the step change magnitude, the spike at the moment of change (also known as transient) [15], and other quantities describing higher frequency content on the voltage [16, 17] and/or current signals [18], have been found to be characteristic to specific appliance types and may be used to distinguish one appliance from another. A good review of the field including the features used and popular methods can be found in [19]. Based on the presence or absence of the need for training labels for the associated algorithms to identify and disaggregate appliances, the common approaches in the NILM literature can be classified as supervised and unsupervised approaches. Approaches that exploit meta-information about an appliance’s operation (like sound, magnetic field, etc.) to facilitate NILM

algorithms, known as sensor-aided NILM approaches, have also been gaining traction lately [20, 21].

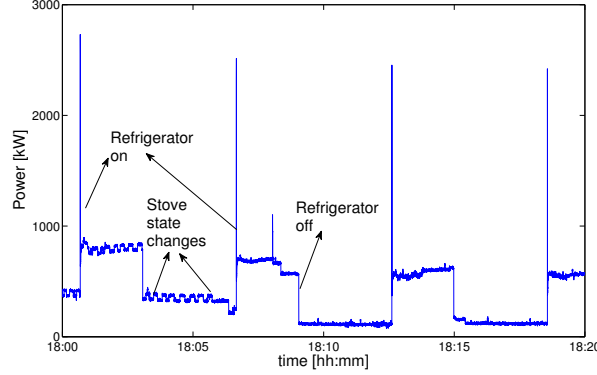


Figure 1.2: Overall power consumption of a residential building during a twenty minutes interval. Some appliances change states quickly during operation (like the stove burner). By extracting appropriate features from these power signatures, appliance types can be classified.

In this thesis, we explore the sub-problem of energy estimation for appliances through automated learning of appliance models, as it applies to supervised NILM. Typically, supervised NILM consists of following steps: training, event detection, feature extraction, classification, and energy estimation as summarized in Figure 1.3. Briefly, the process runs as follows: first, a signal $S[t]$ (typically active power) is monitored at the aggregate level. Then an event detection algorithm (ED) searches for change-points in $S[t]$, which are assumed to indicate when appliances change their state of operation (this also assumes that each operational state has a relatively stable power consumption during steady state). Following this, a feature-extraction algorithm (FE) extracts features associated with these events. A model is created in advance – through a training phase – to learn a function (ϕ) that maps features (X) of the events to labels (Y) corresponding to the appliances (and, optionally, the

state transitions) that were responsible for these events. In the classification step, the function ϕ – learnt during training – classifies the extracted features into one of the appliance labels. Finally, in the energy estimation step, the classified event labels are used to estimate the energy consumed by each appliance. By definition, if $P_k[t]$ is the power consumed by appliance k at timestamp t , then the total energy as consumed by the appliance over all time t is given by $\sum_t P_k[t]$. The total aggregate energy, by definition (and principles of conservation of energy) is given by Equation 1.1. Here, N represents the total number of appliances being supplied by the mains, and ϵ captures contributions of noise, cross-talk, etc. In Section 2.1.1, we formalize the process of energy computation for an appliance as given by ΔP values at event timestamps as detected by ED .

$$\sum_t P[t] = \sum_{k=1}^N \left(\sum_t P_k[t] \right) + \epsilon \quad (1.1)$$

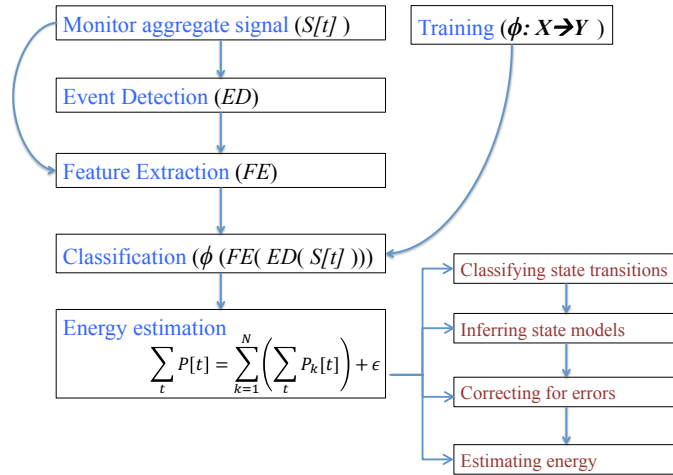


Figure 1.3: Steps involved in supervised NILM. In this thesis, we explore the step of energy estimation by assuming results of the previous steps are already available

Although an intriguing concept in theory, the problem of identification and disaggregation can become very complex as the number of appliances in the house increases. As of date, no complete NILM solution suitable for all types of household appliances has been developed. Roth and Zeifmann in [19] note that the available solutions are either unsuitable for some appliances or still at an early developmental stage and that no complete set of robust and widely accepted appliance features has been identified. Other authors who reviewed the field [22, 23] have reached similar conclusions.

1.2 The need for appliance models in NILM

Part of the reason no generalizable solution for NILM exists yet is that the solutions proposed to tackle energy estimation have typically assumed appliances to be two-state machines exhibiting only *on* and *off* states¹. However, many appliances exhibit more complex behavior and thus, in many cases, it is necessary to learn from data, the kinds of states an appliance can exist in, and the different state transitions that are possible. Zeiffman and Roth note that appliances can be grouped into the following categories based on their operational behavior: (1) On-Off appliances, (2) Continuously variable loads, (3) Permanent devices, and (4) Finite state machines (FSMs) [19]. Efforts to model FSMs, power consumption patterns of variable loads, or power consumption between distinct state changes of appliances (for both On-Off and FSMs) have been distinctly lacking so far. The FSM model is a directed graphical representation with the nodes representing the different states that can occur within an appliance and the edges representing the possible transitions be-

¹For instance, of the 40 studies compared by Armel et al. in Appendix A in [23], none perform energy estimation in multi-state appliances.

tween them. Figure 1.4, taken from [24] shows the FSM models for a refrigerator, tv, and laptop from a publicly available dataset called BLUED [2]. Using the FSM model, it is possible to constrain the power trace and energy estimates resulting from power delta sequences to mitigate some of the errors resulting from NILM.

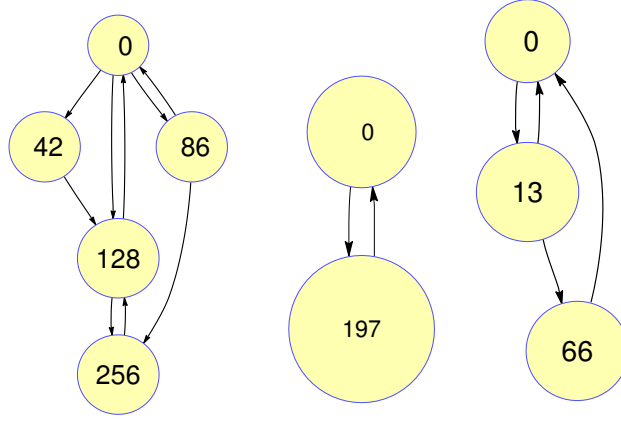


Figure 1.4: FSM diagram for a refrigerator from BLUED-1 (left), tv from BLUED-1 (center) and laptop from BLUED-2 (right).

The majority of the proposed NILM solutions are supervised and event-based. Energy estimation is traditionally an overlooked problem within supervised approaches, and most of the focus is on event-detection and appliance identification (classification) [25]. Again, for context, we look at the 40 approaches summarized in the meta-review by Armel et al. in [23], and find that only 6 perform energy estimation. This trend is reflective of most proposed solutions in this realm. In other words, most studies to date have stopped short of providing a solution that estimates the energy consumed by each appliance and instead provide simply a time-indexed list of operational state changes for each appliance. As we show in Chapter 2, energy estimation fares poorly in supervised solutions without a proper appliance model. Unsupervised approaches typically require prior knowledge of appliance be-

havior, which is assumed to be input by an expert [26, 3]. In some cases, they assume a simplistic two state model for appliances, e.g., [27]. Learning such models from data, and using them to supplement the algorithm has not been studied in detail². We argue that such a process could increase the appeal of both supervised and unsupervised methods by increasing their accuracy. Likewise, having appliance models could help a sensor-aided NILM system estimate energy accurately, even when the input from the additional sensor is not directly related or translatable to power consumption of the appliance it is monitoring. Details on the motivation behind such methods can be found in Section 1.4 and, in Chapter 5, we present a case study that achieves the same.

In Chapter 2, we build the case for the need for appliance models to perform accurate energy estimation for both supervised and unsupervised NILM approaches by evaluating some methods with and without the models on some standard datasets.

1.3 The need for error correction

As shown in Figure 1.3, the output of the classification step in supervised NILM is a time series of state-transition labels assigned to the observed events. Formally, we represent labels as tuples of the form (y_i, t_j) where y_i is the appliance ID for the appliance that caused the event, and t_j is the timestamp at which the event occurred. It is trivial to extract ordered sequences of such labels pertaining to each appliance from this time series. Each such sequence will be denoted as C_{seq} . Assume that an operating model for the appliance that represents each such sequence, as described in Section 1.2 is also available in the form of an FSM. The sequence C_{seq} is a direct

²As will be discussed in later, some authors like Kolter et al have proposed a semi-supervised way to learn appliance models from data for unsupervised approaches, but the generalizability and feasibility of such models have not been validated [28]

product of the event detection and classification steps and errors occurring in these steps (e.g., missed events, misclassifications, etc.). This results in a sequence that violates the FSM that generates it, and causes erroneous energy estimation results. Hence, an additional step of error identification and correction is required before energy estimation is performed.

To get a better understanding of this, let's assume an appliance with three states of operation labelled 1, 2, and 3 corresponding to ΔP values of 40, -40 and -20 respectively. Say, the FSM model is available for the appliance and is represented by a transition probability matrix as given by $A = \begin{bmatrix} 0 & .5 & 0 \\ .1 & 0 & .6 \\ 1 & 0 & 0 \end{bmatrix}$. Also, assume a sequence of events are observed to occur as $C_{seq} = (1, 2, 1, 2, 1, 2)$. Now, the net power consumption of the appliance at each of the events are $(40, 0, 40, 0, 40, 0)$. For simplicity, if we assume the units of the power deltas to be in kilowatt-hour (kWh), and if each event lasts for 1 hour, the total energy consumption for the appliance, over the span of 6 hours, is 120 kWh. Now, assume there was an error in C_{seq} resulting from misclassification, and the third event was classified as a 2 instead of 1, resulting in $C_{seq} = (1, 2, 2, 2, 1, 2)$. As we can see, this observed sequence of events violates the FSM model that generates it, as the transition from state 2 to itself is not possible as given by A . The new C_{seq} indicates that the net power consumption would now be $(40, 0, -40, -80, -40, -80)$, and the energy consumption under previous assumptions would be 40 kWh³. As can be seen, due of one error in the sequence, the energy estimates were off by 3 times. An ideal error correction algorithm should be able to recognize this and replace the erroneous state with its proper substitute.

³negative energy consumption were treated as zeros.

The problem of correcting errors in sequences generated by FSMs has been studied extensively for problems in communication theory, DNA sequencing, pattern recognition, etc [29, 30]. Figure 1.5, adapted from [31] shows a typical channel data stream as it goes through corruption from a noisy channel, and subsequent correction via some error correction mechanism (decoder). Hart drew similarities between the NILM problem and the problem of decoding additive signals on a Multiple-Access Channel (MAC) [32]. He cited the low signal to noise ratio, and low message rate to channel capacity ratio as some of the desirable features of this channel. But compared to other decoding problems where error correction is required, the NILM channel presents a unique set of challenges. Typically, solutions for correcting errors assume certain properties about the noise in the channel, which makes the correction process more systematic. In addition, as shown in Figure 1.5 most error correcting algorithms have a coder for the input sequence that allows it to be encoded in a specific way to facilitate decoding. The NILM problem does not allow for either of these steps, as it has no control over the input sequence. Essentially, the only flexibility that exists is the capacity to exploit information about the constraints imparted by the FSM to correct for errors. It is also worth noting that the definition of the channel in this problem not only includes the NILM hardware, but also the process by which the algorithm arrives at appliance specific time-series. Finally, since the goal of correcting errors in NILM is to perform energy estimation, errors are highly sensitive to propagation, and hence the error-correction algorithm for such errors needs to minimize such possibilities.

Hart also proposed an algorithm to correct errors in sequences generated by

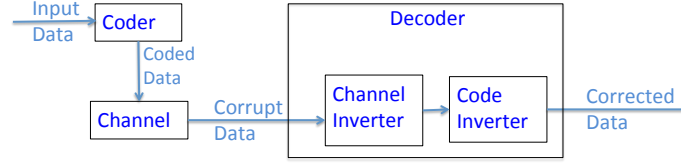


Figure 1.5: A generic communication model for a sequential data stream as it gets corrupted on passing through a channel.

FSMs in [31] where he mentioned NILM as one of the potential use cases for his algorithm. His framework introduces the idea of channel rules, where the user knows, a priori, the kinds of modifications the channel-inverter can impart on the observed sequence. In Chapter 4, we modify his concept to allow for the same channel rules (namely single bit insertion, deletion or substitution) for all sequences, which makes the method more generalizable. In addition, we introduce constraints specific to energy estimation in the algorithm, which caters to the ultimate goal of the error correction. Although the algorithm utilizes prior beliefs specific to appliance behavior, we believe the general framework of incorporating domain knowledge to correct for errors in sequential data streams can be beneficial in other fields like character recognition, automated spelling correction, DNA sequencing, etc. The appliances, in our case, act as *coders* (Figure 1.5) constraining the input data to assume a finite number of states. Utilizing this information about appliance behavior, and some assumptions about how the *channel* is likely to corrupt the coded data, we create an appropriate decoder that inverts the effect of the channel. As mentioned before, this approach has the potential to be transferred to other fields once the effects of the coder and channel have been established. To summarize, our major contribution in this particular chapter (Chapter 4) are: (a) an extended algorithm for correcting errors in supervised NILM with novel evaluation metrics (b) results from real data from 43 appliances collected from 19 different houses.

1.4 The case for virtual metering

Until NILM becomes a practical and viable solution, one temporary way to obtain appliance level power consumption information and overcome the aforementioned limitations of direct plug-through meters, is through the use of indirect sensing methods to infer the electricity consumption of appliances. The idea is to use sensors that do not necessarily need to be in contact with the power source that feeds the appliance⁴ to measure effects in the appliance that correlate with its power consumption. This kind of arrangement when information from the physical implementation of a system (also known as side-channel information) is exploited in order to infer its intrinsic characteristics is also common in cryptography, where there have been demonstration of possible attacks to an encrypted communication by making use of information leaked through side-channels such as the power consumed by the computers (e.g., [33, 34]). Sensors like sound sensors, light sensors, magnetic field sensors, vibration sensors, etc. are possible alternatives [35]. Such sensors typically are cheaper than plug-sensors by orders of magnitude, sometimes ranging as low as 2-3 dollars [21]. They are also easy to deploy, and can sense effects from appliances that do not have a dedicated or traditional plug power supply. Appliance metering using such a setup typically also requires the monitoring of aggregate power in the building using an energy meter.

But use of such sensors to meter appliance-level energy comes with challenges of its own. The first challenge is that of calibration. Typically methods that rely on converting the effect sensed by the sensor directly into power values require proper calibration. Factors like sensor placement, orientation, ambient noise, etc. can

⁴Hence, called contactless sensors

greatly affect readings in such situations. To avoid this, some methods rely on utilizing these sensors only as state or event estimators for the appliance. These methods typically also require the aggregate power to be measured using an electric meter. They then use the aggregate power measurements, during events as reported by such sensors, to calculate the change in power observed during events. Once all such power changes are logged, energy is then inferred by assuming power values to be constants between events. Such methods are prone to errors resulting from simultaneous events occurring at the aggregate level, and false readings from the sensors. This is especially likely for complex appliances with multiple states. Figure 1.6 shows a case where missing an event can result in accumulation of errors, which can significantly impact the estimation of energy consumed by the appliance. Knowing the exact states that the appliance can exist in, and the respective power consumptions of those states are also crucial in estimating the energy consumed. In Chapter 5, we utilize our energy estimation framework for virtual metering of appliances. Although NILM in itself provides an exciting alternative to the appliance-metering problem by only requiring single point sensing to perform appliance-level metering, the technology is not yet at a stage where it can be implemented in a realistic setting [19, 22]. But the framework we proposed for energy estimation can be used to virtually meter appliances in the case where several contactless sensors provide state change information to the aggregate power monitor. This method is attractive because it requires minimal calibration⁵, and has the ability to correct for errors imparted by noise, false events, etc. This provides a practical method for energy audits of appliances using commercially available and easily deployable sensors and using a framework that addresses some of the major issues in the energy metering process.

⁵The power meter at the electric panel needs to be calibrated to read accurate values.

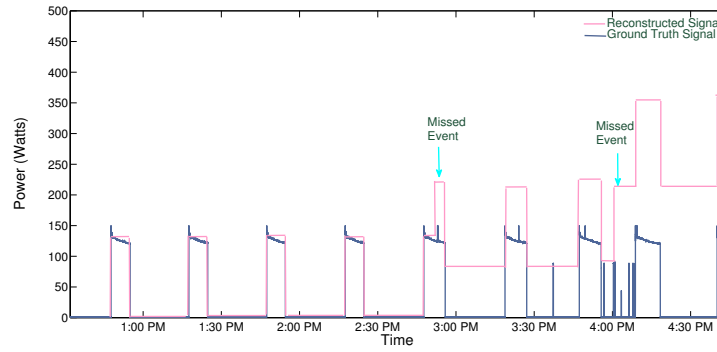


Figure 1.6: Ground truth power trace for Refrigerator extracted from a public dataset (BLUED [2]) plotted against a power trace reconstructed using event labels. As can be seen, missing a couple of events results in a reconstructed trace that is very different from ground truth.

We test this using 6 different appliances with multiple states as monitored using such contactless sensors and show that results are promising. Until NILM becomes viable, or plug-level meters become cost-effective, we believe this is a practical way to conduct appliance metering.

Chapter 2

Motivating case studies

In this chapter, we build an empirical case for the need and purpose of an energy estimation framework. In Section 2.1, we empirically evaluate some standard datasets to understand how lack of a robust framework affects energy estimates. In Section 2.2, we explore the feasibility of a cheap and easily deployable Electro-magnetic Field (EMF) sensor to provide event labels for virtual metering.

2.1 Case studies on appliance models

This section explores the advantages of having appliance models for energy estimation. Throughout this section, and the thesis in general, we use publicly available, and standard datasets: namely, REDD [37] and BLUED [2] for the purposes of our analysis. BLUED contains high frequency current and voltage data sampled at 12 kHz for a week from one house. The dataset also contains real and reactive power sampled at mains (at 60 Hz) with each event timestamp labelled with the ID of the appliance that caused it. To our knowledge, it is the only dataset available publicly that has appliance-level event labels. We add two more houses in BLUED, which

This chapter is partly based on the manuscript in [36]

are not public yet, but are collected using the same methodology later in the thesis. These will be referred to as houses BLUED-2 and BLUED-3. REDD, on the other hand, has real power sampled at 1 Hz, with sub-circuit level ground truth sampled at ~ 0.25 Hz. It contains data from 6 houses collected for varying periods of time (almost a month). Although a couple of houses in REDD have high frequency data, the files were found to be corrupt with usable data missing for the most part. The appliances we choose to analyze are selectively picked based on their complexity, and availability of ground truth. In the case of REDD where only sub-circuit level ground truth is available, and event labels are not, we label the events at the sub-circuit level manually to the best of our knowledge. Inevitably, only appliances that have a dedicated sub-circuit get chosen because of this limitation on the availability of ground truth labels. Through this section, we aim to make the case that learning better appliance models can improve energy estimates of an appliance, irrespective of the technique used for energy estimation.

2.1.1 Formalizing the energy estimation process

Terms like *energy estimation* and *appliance models* will be used throughout the thesis; in this section we formalize the definition for these terms. The process of appliance modeling refers to the estimation of the following aspects of a given appliance y_i :

1. The number of distinct state transitions possible for y_i , denoted by the set $\Delta P = \{\Delta P_1, \dots, \Delta P_n\}$
2. The number of distinct states y_i can exist in, denoted by the set $P = \{P_1, \dots, P_n\}$
3. The transition probabilities between the state transitions, as denoted by the matrix A , with $a_{ij} \in A$ indicating the probability of transition between ΔP_i

and ΔP_j .

As illustrated in Figure 1.4, FSM diagrams provide a good visual way to illustrate these properties.

To perform energy estimation the event timestamps associated with y_i are required, and we assume they are denoted by the set $T = \{t_1, \dots, t_m\}$. At each timestamp t_i , the state transition that the appliance went through (from the set ΔP) needs to be computed. Once ΔP values are calculated, a simplistic approach to obtain the power trace for an appliance given a ΔP sequence, would be to assume piece-wise constant power between each element of this sequence, along with an initial power value P_0 , and then perform the following sum to obtain the power value at timestamp t : $\hat{P}[t] = P_0 + \sum_{j=1}^{|R|} \Delta P_j$ where $|R|$ denotes the cardinality of set $R = \{t_j : t_j < t; \forall t_j \in T\}$. Energy (\hat{E}_i) for y_i between timestamps t_1 and t_m can then be computed with the following relation:

$$\hat{E}_i = \hat{P}[t_1](t_2 - t_1) + \hat{P}[t_2](t_3 - t_2) + \dots + \hat{P}[t_{m-1}](t_m - t_{m-1}) \quad (2.1)$$

It is worth reiterating that this method of energy estimation assumes that power consumption for appliances remains constant between consecutive events, and is referred to as the piece-wise constant assumption. In Sections 2.1.3 and 2.1.4, when we compute energy without appliance models, we assume that sets ΔP and P are not available, and hence, power differentials, and consequently $\hat{P}[t]$ values are computed as they are observed during events. As a result, $\hat{P}[t]$ will assume a wide range of values, which effects the energy estimation results. In Section 2.1.5, we assume that this information is available and force $\hat{P}[t]$ to take on a finite set of values.

2.1.2 A more general view of the problem

In a sense, the whole problem of energy estimation in NILM is the problem of inferring the appliance model (ΔP , P , and A) from event labels, and using the model to, first, correct for any erroneous labels, and, second, estimate energy as given by Equation 2.1. This problem statement can be made more general by drawing comparisons with a communication model as depicted in Figure 1.5. The appliance is the *coder* which forces the input data to assume certain properties as dictated by their FSMs. The electric circuit along with the algorithms for ED , FE , and ϕ constitute the *channel*. The process of inferring the appliance model is the *code inverter* which takes in the times series of event labels as the input, and outputs the appliance model (ΔP , P , and A). The process of error correction is akin to the *channel inverter*, which takes in the corrupt sequence as input, and outputs the corrected sequence. The process of inferring energy from the corrected sequence is through a straightforward implementation of Equation 2.1.

Notationally, we denote the *code inverter* as a function $g : \Xi \rightarrow \Psi$, which maps a time-series ξ in a vector-space of time-series data denoted by Ξ , where $\xi_i \in \mathbb{R}^n, \forall \xi_i \in \xi$, to the model-space Ψ , where each $\psi \in \Psi$ is a set of the form $\{\Delta P_\psi, P_\psi, \text{ and } A_\psi\}$. Similarly, the *channel inverter* is denoted by a function $h : (\Upsilon \times \Psi) \rightarrow \Upsilon'$, where v is a time-series in a vector-space denoted by Υ , with $v_i \in \mathbb{Z}^n, \forall v_i \in v$. All the time-series in Υ have a specific number of unique elements (given by the cardinality of their respective ΔP_ψ sets, i.e. $|\Delta P_\psi|$, in (v, ψ)). Υ' is a subspace of the vector-space Υ (denoted mathematically as $\Upsilon' \leq \Upsilon$), where all elements v'_i , of time-series $v' \in \Upsilon'$, follow the transitions according to A_ψ . Basically, function h maps a time-series and its associated model, denoted by (v, ψ) , to another

time-series, denoted by v' , that follows certain Markovian restrictions imparted by the transition probability matrix (A_ψ) ¹.

We spend most of Chapter 3 in presenting a method to infer the function g . The inference of a function with such kind of mapping has been studied in literature for various applications including FSM estimation [32], topology estimation for latent variable models [38, 39], etc. The available solutions typically work by iteratively creating an FSM that maximizes the likelihood of the observed data. As will be shown, the case of energy estimation carries a unique set of properties that allow for the use of certain techniques that is not possible in a typical topology estimation problem. We utilize these properties in our framework presented in Chapter 3. Similarly, Chapter 4 presents a method for learning function h . This is a problem that has been studied under fields like error correction techniques in noisy channels in communication theory [32], inference of hidden states in latent variable models [40], etc. We present a solution that incorporates certain appliance behavior related properties, and prior belief about noise in the channel, in the Chapter 4. The solutions in this framework (specifically, inference of functions g and h) can be utilized in similar problems where an observed stream of data assumes a specific number of states, and is supposed to follow defined rules of transition. Automated spelling correction, character recognition, DNA sequencing, etc. are some possible examples.

¹In fact, and as we will show later, the time series v' that h maps to is the one in Υ' that has the least cost of alteration from Υ .

2.1.3 Energy estimation in supervised settings without appliance models

We use the first house in BLUED data for this analysis. Event detection and classification labels were assumed to be perfect, and available to perform energy estimation. Without any models for appliance behavior, the algorithm looks for power differentials observed at the aggregate level around each event, and uses that as the state change in power for the associated appliance. The power consumption is assumed to be a piece-wise constant between consecutive events. A similar method was applied in [36] to perform energy estimation in the sensor-aided case. Throughout this thesis, we use percentage error in energy estimation as the metric for evaluating our frameworks ($\frac{\hat{E}-E}{E} \times 100\%$). Here \hat{E} is the estimated energy and E is the ground truth energy as measured in kilowatt hours (kWh).

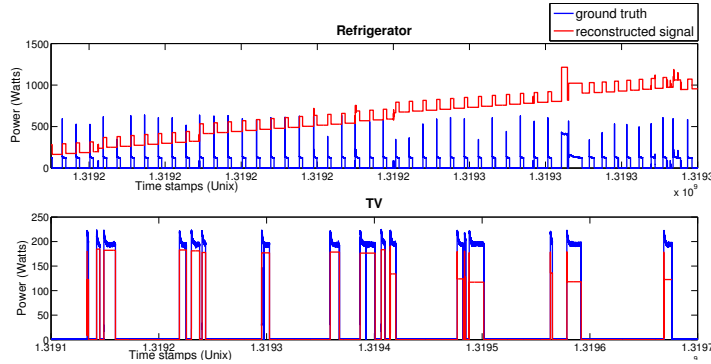


Figure 2.1: Section of a power trace reconstructed using event labels at mains for refrigerator (above) and TV (below)

The first row for each appliance in Table 2.1 shows the accuracy of such energy estimation techniques based on perfect classification labels, but with no models used for energy estimation (labelled "*without a model*"). As is evident from the errors,

the energy estimation fares poorly. Even with almost-perfect ground truth, the observed discrepancy exists because of the following reasons:

1. Sensitivity to outliers (caused either through mislabeling or simultaneous events) which results in accumulation of errors.
2. Step changes in power are different when appliances turn on and off. So, the model will accumulate errors because of this.

In addition, figuring out the transient window to calculate power differentials is not trivial, as some transients are longer than others. This impacts the sensitivity of the energy estimation framework significantly as well. Figure 2.1 serves to elucidate the shortcomings of this method. The traces in red are the reconstructed power traces based on observed power differentials at the mains, and the ones in blue show the respective ground truth traces; the trace of the refrigerator shows the accumulation of errors because of points (1) and (2) mentioned above.

2.1.4 Energy estimation in unsupervised settings without accurate appliance models


















In this section, we highlight some of the shortcomings of expert based prior models for appliances in unsupervised solutions. As a representative case, we choose the HMM models proposed by Parson et al. in [3] and point to where models based on general expert input might fail. The method relies on a training process where prior knowledge (which includes the states, emission probability, and transition probability) of generic appliance types are tuned to specific appliance instances using the aggregate power consumption data. Each appliance is represented using a probabilistic graphical model, and the training process corresponds to learning the parameters

Table 2.1: Energy estimation errors based on a framework with and without appliance models. The basic model that is used for comparison assumes that the observed changes in power lie within the set of ΔP 's that are assigned.

House	Appliances (label)	Models for ΔP (Watts)	Estimated Energy (kWh)	Actual Energy (kWh)	Error (%)
BLUED	Refrigerator (111)	Without a model	213.7	6.7	3089.6
		[70, 135, -135, -70, 350, -410]	7.1	6.7	6.0
	Lamp (101)	Without a model	1.1	1.1	0
		[12, -12]	1.1	1.1	0
	TV (129)	Without a model	4.9	6.4	-23.4
		[20, 180, -180, -200]	6.2	6.4	-3.1
	Computer (118)	Without a model	13.2	2.0	560.0
		[49, -49]	2.1	2.0	5.0
	Laptop (120)	Without a model	7.4	0.8	825
		[25, -25]	1.4	0.8	75.0
	AV System (112)	Without a model	7.3	4.9	49
		[41, -16, 8, -41, 48]	4.8	4.9	-2.0

of this model. The training process deploys an Expectation Maximization (EM) algorithm using prior models on small overlapping windows of aggregate data to find clean signatures of individual appliances. These signatures are then used to further tune the prior models. Once the models are fine-tuned, a modified version of Viterbi algorithm is used to infer appliance level energy consumption. We pick two implementations of the paper in particular: one without training (NT) and one with aggregate training (AT). The former relies purely on the expert input of priors

Table 2.2: Errors in energy estimation with (AT) and without (NT) training using generic prior models of appliance types in REDD data. The old prior models are models used by the author in [3] and the updated models are empirical models chosen by us.

Appliances	House (REDD)	Expert models				Improved models				Color Scale (0-1)
		Old State means (Watts)	Old Transition probabilities	Error (NT) (%)	Error (AT) (%)	Updated State means (Watts)	Updated Transition probabilities	Error (NT) (%)	Error (AT) (%)	
Refrigerator	2	$\begin{bmatrix} 2 \\ 180 \\ 160 \end{bmatrix}$		91	33	$\begin{bmatrix} 6 \\ 186 \\ 160 \end{bmatrix}$		86	26	
	3	$\begin{bmatrix} 2 \\ 180 \\ 160 \end{bmatrix}$		121	109	$\begin{bmatrix} 1 \\ 132 \\ 118 \end{bmatrix}$		71	92	
Microwave	1	$\begin{bmatrix} 1 \\ 1700 \end{bmatrix}$		81	74	$\begin{bmatrix} 4 \\ 1575 \end{bmatrix}$		63	70	
	2	$\begin{bmatrix} 1 \\ 1700 \end{bmatrix}$		68	55	$\begin{bmatrix} 6 \\ 1800 \end{bmatrix}$		54	62	
	3	$\begin{bmatrix} 1 \\ 1700 \end{bmatrix}$		95	95	$\begin{bmatrix} 2 \\ 1745 \end{bmatrix}$		92	94	
Dishwasher	4	$\begin{bmatrix} 1 \\ 1400 \end{bmatrix}$		105	101	$\begin{bmatrix} 0 \\ 1300 \end{bmatrix}$		96	76	
Washer/ Dryer	1	$\begin{bmatrix} 0 \\ 5000 \end{bmatrix}$		447	70	$\begin{bmatrix} 1 \\ 5420 \\ 650 \end{bmatrix}$		220	64	
	3	$\begin{bmatrix} 0 \\ 5000 \end{bmatrix}$		4930	48	$\begin{bmatrix} 1 \\ 4408 \\ 4564 \end{bmatrix}$		78	43	

and transition probabilities to estimate appliance states, the latter picks segments from aggregate data where only the appliance of interest is operating (by performing EM calculations on the observed data with expert based appliance models as priors) in order to tune the model parameters. Putting aside, for a moment, the possibility that such segments where only one appliance operates at a time might not be found for training in a given aggregate power time series, we focus on how the procedure is limited by appliance models; namely, the training process itself is dependent heavily on the type of prior chosen. Table 2.2 (columns labelled "*expert models*") shows the performance of both methods on a few appliances in the REDD dataset using the same parameters as used by the author in [3]. In section 2.1.5, we choose better appliance models and show improvements in energy estimation (Table 2.2). The error values reported are the percentage error in energy estimation over the duration of the datasets. All pre-processing and un-listed parameters are kept the same as in the original implementation to provide a fair comparison.

2.1.5 Energy estimation with basic appliance models

To illustrate how simple appliance models can significantly improve energy estimation, we manually assign possible state changes for an appliance based on its labels at the aggregate power data. Tables 2.1 and 2.2 serve to elucidate this point. The third column in Table 2.1 shows all the empirically built state transitions that different appliances in BLUED could partake in. A heuristic is added such that all observed power differentials at the mains are classified as the closest power differential (in terms of Euclidean distance) that the appliance can have. The results in terms of energy estimation are found to be very close to ground truth, and significantly better than the case with no models. Figure 2.2 illustrates the difference between the estimated power traces and ground truth. Similarly, Table 2.2 (columns

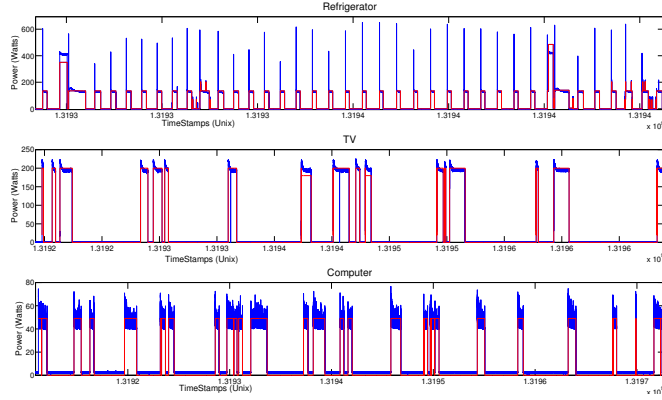


Figure 2.2: Section of a power trace reconstructed using event labels and appliance models at mains for refrigerator, TV and computer

labelled "*improved models*") demonstrates improved performance on appliances using better prior models. Figure 2.3 provides some insight into how better appliance models provide improved energy estimation results in an HMM based approach; the training data selected for learning a dishwasher model from aggregate data using EM for a house in REDD extracts the right training data with better models.

For the supervised case (Table 2.1), apart from the models being fairly primitive, the discrepancies probably also result from modeling power values as step-wise constants and not taking into account different behaviors appliances exhibit in between state transitions. In the case of unsupervised modeling (Table 2.2), the values are empirically extracted without any sophisticated modeling, or training from ground truth. The conclusion from this analysis is that appliance models can improve the energy estimation techniques drastically. In Chapter 3, we propose some methods for learning appliance models automatically from data, and provide subsequent validation.

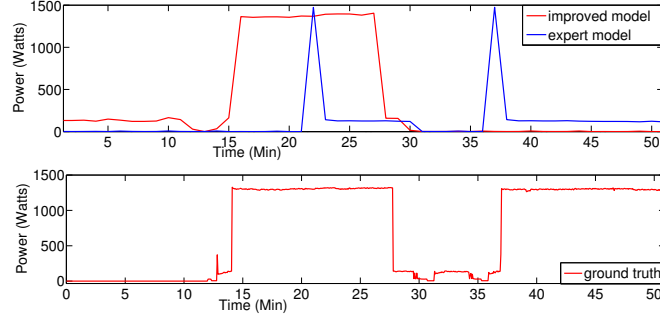


Figure 2.3: Data extracted for training HMM models for a dishwasher in House 4 for REDD from the aggregate using different priors (above), and the ground truth for dishwasher as measured at circuit level (below). The newer model is capable of extracting the right segment for learning, contributing to better energy estimation results

2.2 Case studies on Virtual Metering

The case for utilization of the energy estimation framework to perform virtual metering hinges on the feasibility of using a cheap and easily deployable sensor that can provide (side-channel) information in the form of event labels. To achieve this, we worked on the design an implementation of an Electro-Magnetic Field (EMF) sensor that amplifies the magnetic field power values that it receives from appliances and conveys this to a central computing platform via a wireless interface. We completed the system with a three-phase meter that can sample current and voltage readings from the aggregate level at high frequencies, and a plug-level meter than can collect ground truth data. The goal of this case study was to establish the feasibility of using external sensors to label events. Hence, the values were sampled at higher frequencies than what commercially (and easily) available sensors might be able to achieve. This work was done in collaboration with Dr. Anthony Rowe, and Niranjini Rajagopal, and is based on the manuscript in [36]

2.2.1 System Hardware

The system consisted of a three-phase meter and EMF sensors that, in conjunction, collected and correlated appliance on/off events to meter them. Figure 2.4 shows an overview of the general system architecture which consists of the three-phase meter used for overall mains power metering, plug-meter devices used for ground-truth data collection and our EMF detectors. These components are connected to a networked backend and displayed on the dashboard described in [41]. The circuit panel meter was a custom three-phase power meter that employs ADE7878 energy metering chip from Analog Devices, specifically to collect high resolution data which can be correlated with events from our EMF detectors. Off-the-shelf energy meters often make it difficult to capture high-speed raw waveforms. In contrast, the meter samples both the current and the voltage on each phase at 1KHz, and uses an on-chip Digital Signal Processor (DSP) to compute true, apparent and reactive power, as well as several other energy metrics.

Both for ground-truth validation and for devices that can benefit from remote actuation, we used a FireFly plug meter [42]. Each plug meter contains the ability to monitor and control two electrical outlets using wireless communication. The meter uses an efficient switching power supply that draws less than 0.1 watts ensuring that it does not unnecessarily increase building power consumption. The meter measures true power, apparent power, power factor, frequency, rms current and rms voltage with a sampling rate of 1KHz. Two solid-state relays are used to independently control each outlet.

To get events associated with appliances through contactless methods, as de-

scribed in Chapter 1, we designed a custom EMF sensor (Figure 2.6). The core principle behind the EMF event detector is the ability to sense when an appliance changes state by monitoring changes in nearby electromagnetic fields. From the laws of physics, we know that alternating current flowing through a conductor will generate a corresponding magnetic field (H). Typically AC wires run as parallel pairs and hence most of the magnetic fields cancel out. However, imbalances in wires and stray currents flowing on ground lines as well as through appliances produce a significant magnetic field. The amplitude of this field is generally small (millivolts), but if sufficiently amplified, one can reconstruct the original source to a reasonable degree of approximation.

Each time an appliance changes how much power it is consuming (e.g. for example transitioning between on and off) there is a corresponding change in the nearby magnetic field. In contrast, differences in voltages are responsible for creating electric fields. This means that an appliance that is not drawing current may still generate a strong electric field (E). The distinction between the electric and magnetic field is useful for two reasons. First, the electric field can be used to detect if a device is "live" or not. For example, overhead lights often switch the hot AC lines which can easily be detected by inspecting the electric field. Second, if a device is powered, but not active, the electric field strength can be used as a guide to find placement areas where there will be a strong magnetic field once current begins to flow. Since the electric field is not dependent on current flowing, abnormal fluctuations in the electric field tend to indicate potential noisy situations. For example, if people are nearby or touching the sensor, both the electric and magnetic field will be disturbed.

Figure 2.5 shows two example waveforms received by the circuit when placed

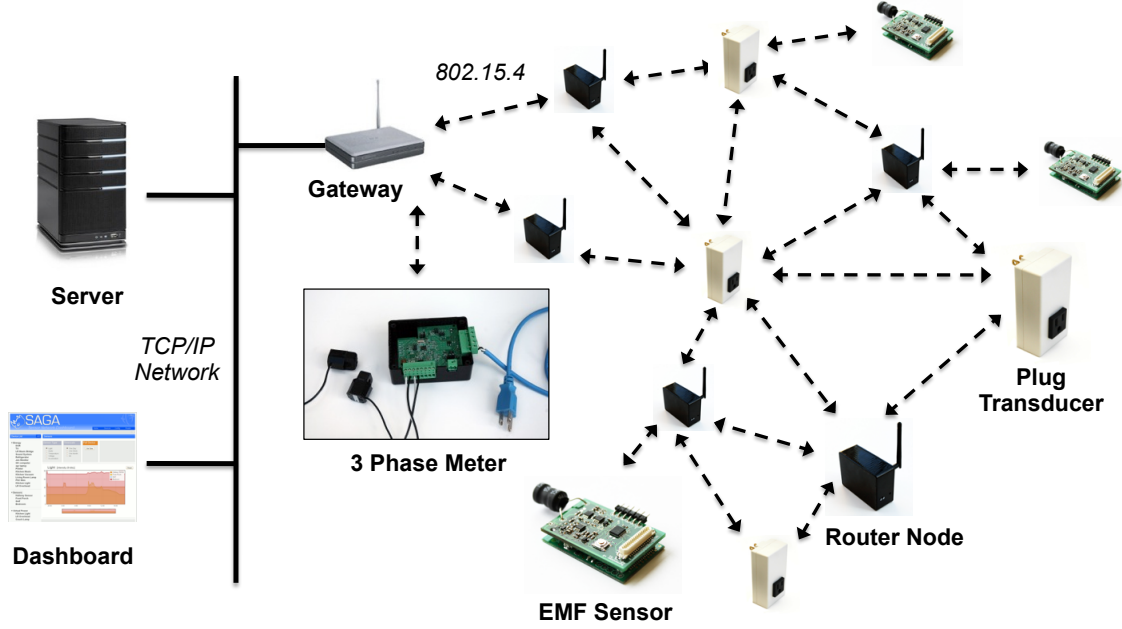


Figure 2.4: Network Architecture

near a ceiling fan and a desktop computer. Point (a) in the ceiling fan waveform denotes when the wall switch is turned on which generates a corresponding electric field. At point (b), the ceiling fan is manually switched on (by pulling the hanging cord) causing current to flow and hence generating a magnetic field. The bottom line in the upper graph shows the root mean square (RMS) value of the magnetic field signal averaged over a window of 16ms ($1/60\text{Hz}$). The bottom graph shows the magnetic field and the same sliding RMS value for a desktop computer. In both of these cases the edges in the RMS signal are quite pronounced.

Figure 2.6 shows a picture of the EMF detector hardware connected to a FireFly wireless sensor node. The FireFly node is responsible for periodically sampling the magnetic field in order to report appliance activation events. Since the signal from the EMF detector has a steady-state value associated with the current of the appliance, the FireFly node can duty-cycle its sampling to save energy. We explore

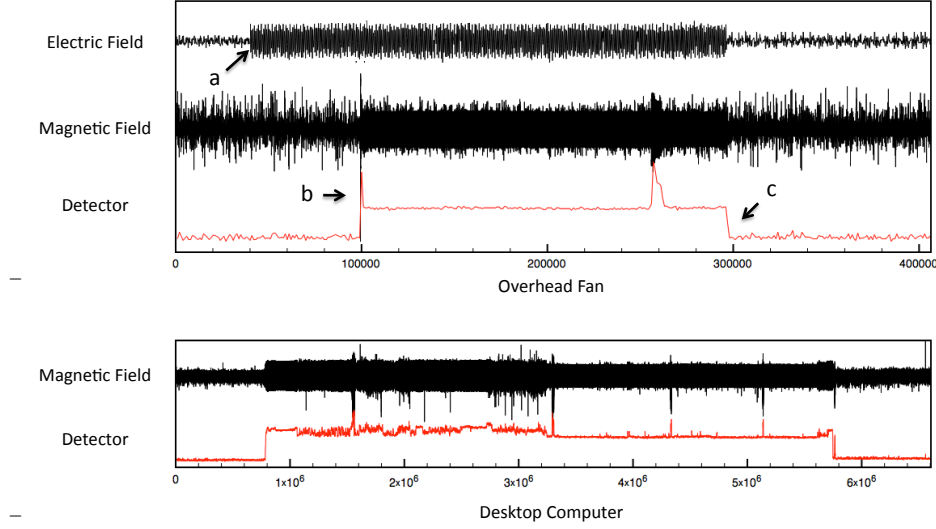


Figure 2.5: EMF Event Detector Waveforms. Top: Ceiling fan with light switch activated at point (a), manually turned on at point (b), and turned off at point (c). Bottom: Desktop computer is an example of a noisy signal due to switching.

this and provide additional evaluation results related to the sensor’s sensitivity and detection range in [21]. We measured that the EMF detection front-end consumes approximately $45\mu W$ but this value can vary depending on the strength of the measured magnetic field.

2.2.2 System Software

In this section we first describe and evaluate the event detection algorithm running on each EMF detector. We then describe the approach used to label and record per-appliance energy usage by communicating with the three-phase meter.

Event Detection Firmware

The EMF sensor locally performs two main tasks: (1) it adjusts a hardware gain setting on the magnetic field sensing front-end to maintain a fixed peak-to-peak value for the sensed signal and (2) it is responsible for detecting significant changes

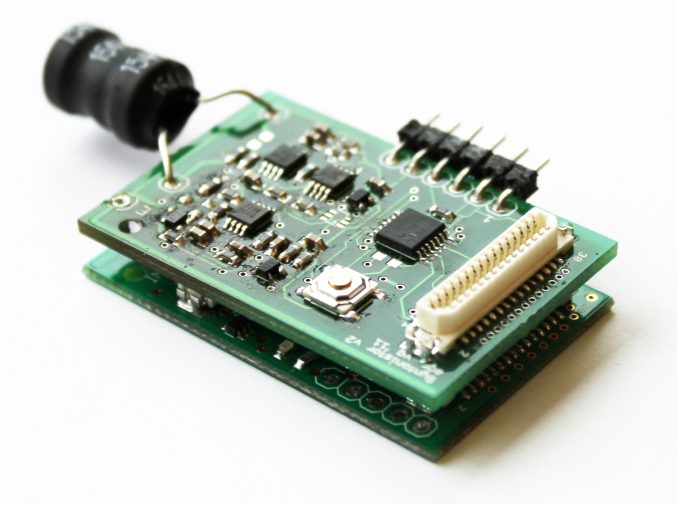


Figure 2.6: EMF event detector stacked on FireFly3 sensor node.

in field strength and reporting those to the wireless sensor node.

Our first challenge was choosing an adequate ADC sampling rate such that the EMF sensor could accurately reconstruct the magnetic field signal. Sampling too slowly would lead to poor performance, while sampling too quickly would waste energy. In order to determine a sufficient sampling rate, we validated the detection accuracy as compared to sampling rate using the experimental setup shown in Figure 2.7. The EMF sensor was placed at a distance of 5 *cm* from the wire while two different appliances were transitioned (switched on/off) 40 times each. The accuracy metric is computed as the number of correct transitions divided by the total transitions detected in order to penalize false positives. For this experiment we used a 60 Watt fan and a 60 Watt incandescent light bulb since one is composed of a largely inductive load and the other almost entirely resistive. Based on the experiment, we decided on 1 kHz as the sampling frequency. Further details can be found in [36]

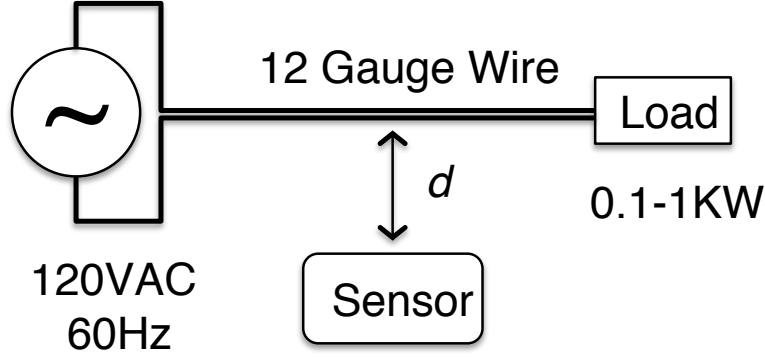


Figure 2.7: Experimental Setup

Appliance Metering

Each time an event is detected by an EMF sensor, a time-stamped message is sent back to the gateway. Our algorithm then determines the change in power across a time window before and after the event (the size of this window is evaluated in Section 2.2.3). This power change can be positive or negative based on whether the appliance went from a low-power to high-power state (eg. going from OFF to ON) or vice versa. We then make an assumption that the power for this appliance remains constant until the next observed event. At the next event, we either add or subtract the new power delta and integrate energy over the previous period. This process is continuously repeated while the system runs. Figure 2.8 illustrates this power computation process for a couple of hours while metering a refrigerator.

A benefit of estimating power based on events, apart from its simplicity and effectiveness, is that it can handle finite state transitions of an appliance to estimate varying power consumption over a cycle, which has traditionally been a hard problem to tackle. The main limitation is that this approach fails to correctly estimate energy usage for devices whose power consumption varies slowly over time (and be-

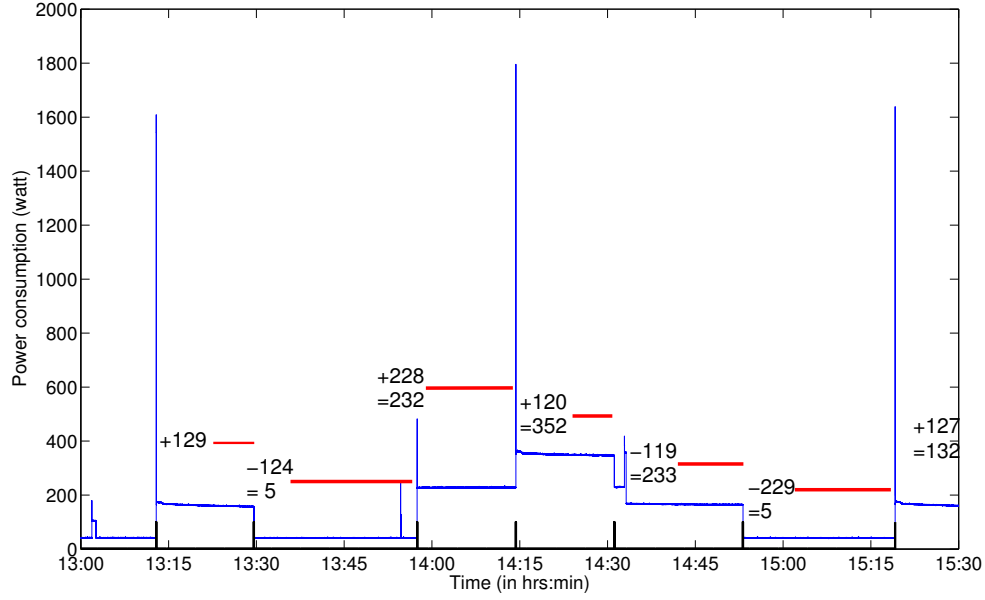


Figure 2.8: Piecewise appliance energy estimator running on a refrigerator. Thick vertical lines at bottom show EMF event inputs onto the mains power waveform. The bars across the top represent regions where power is estimated to be constant.

tween events) or devices that have significant phantom loads. This is admittedly a very rudimentary way to estimate energy based on events, as the focus here is on the ability of contactless sensors to provide events. The focus of this thesis, however, is on creating a software framework to make this process robust and accurate. In Chapter 5, we actually incorporate our energy estimation framework for the purposes of virtual metering.

2.2.3 System Evaluation

In order to evaluate overall energy metering performance, we deployed our system in a single-family residential building, and collected data for 1 week. We installed our three-phase energy meter on the two 150amp mains lines that feed the house. We installed plug-level meters (28 total) on any accessible plug-load appliances. Fi-

nally, we installed 7 EMF sensors on the following appliances: LCD TV, Washing Machine, Toaster Oven, Window AC, Laser Printer, Refrigerator and Iron. Each appliance with an EMF sensor also had a plug-meter which could be used to measure ground-truth readings.

Event Detection Performance

First, we evaluate how accurately each EMF sensor was able to detect appliance transitions. Over the period of 1 week, we compared the EMF sensor's threshold-based event detector with a hand-tuned threshold selected for each plug-meter. When looking at the plug-meter data, we selected thresholds that represented each appliance in either an *on* or an *off* state. Figure 2.9 shows the confusion matrix for each appliance as well as the overall average across all appliances. The confusion matrix was generated by comparing the amount of time that the EMF sensor categorized an appliance in a state that agreed or disagreed with the ground-truth. This is a more telling metric than doing an event-by-event comparison since one poor event transition could potentially set an appliance in the wrong state for an extended period of time.

From the confusion matrix, we can observe that the system in general performed well in categorizing the appliance-state in agreement with the ground-truth. We also observe that the Iron, TV and Laser Printer show above average misclassification. In the case of the Iron, the EMF-node was accidentally moved away from the appliance around the fifth day in the week of testing, hence it failed to detect some of the *on* events. The EMF-detector assumed that every appliance takes only two states, but the TV takes three states - *on*, *off* and *standby*. When the TV was in the *standby*

EMF Sensor	Ground Truth		EMF Sensor	Ground Truth		EMF Sensor	Ground Truth		EMF Sensor	Ground Truth	
	On	Off		On	Off		On	Off		On	Off
	Off	On		Off	On		Off	On		Off	On
Refrigerator (195)			Toaster Oven (5)			TV (20)			Window AC(2)		
On	99.8	0.1	On	99.7	0.08	On	92.6	0.08	On	99.1	0.00
Off	0.2	99.9	Off	0.03	99.2	Off	7.4	99.2	Off	0.09	100
Washer (47)			Iron (29)			Printer (24)			Total (322)		
On	99.5	0.00	On	83.1	0.50	On	96.5	2.7	On	95.8	0.7
Off	0.05	100	Off	16.9	99.5	Off	3.5	97.3	Off	4.2	99.3

Figure 2.9: EMF sensor event detection confusion matrices. Total event count in parentheses after appliance name.

mode, the signal strength was not sufficient to be classified as an On-state by the EMF-detector, resulting in a misclassification. Notice that all appliances except the Printer perform well while the appliance is *off*. The EMF-sensor corresponding to the Printer was physically close to a monitor and was detecting the signal due to both appliances. Hence the EMF-sensor detected an *on* state even though the Printer was *off*.

Appliance Metering Performance

We then evaluated how well the system worked at estimating the energy consumption of each appliance. We analyzed how the system performed in two steps. First, we computed the ground-truth energy consumption of each device as recorded by the plug-meters. Next, we ran our energy estimation algorithm based on main measurements and using hand-selected thresholds derived from the plug-meters. The idea was that this would show appliance-metering performance assuming perfect event detection. Finally, we let the system compute the energy for each appliance using the actual EMF sensor data. Figure 2.10 shows a stacked bar-graph plot of each scheme. In this example, we choose a before-event and post-event window size of 2 seconds for determining the change in mains power for the appliance. Figure 2.11 shows the error for the EMF sensor as we vary this window size. In

certain cases, having too small a window leads to poor performance because the appliance has a slow *on* or *off* transient. For example, an appliance like a refrigerator draws an abnormally high amount of current for a second or two while starting up. If the window is too small, the system miscalculates the energy consumption by assuming the high transient-power for the entire *on* cycle of the appliance. On the other hand, even if we select a large window size, the system runs the risk of miscalculating the energy due to events that occur nearby in time. A window size of 2 seconds performed the best, with an overall error of 0.83%, which can also be observed in Figure 2.11.

Consistent with the confusion matrix, we observe that the EMF-based energy estimate of the TV and the Iron are lower than the ground truth, due to the misclassification of few *on* states as *off* states. The estimate of the energy consumption of the Laser Printer is much higher than ground truth. As mentioned earlier, the corresponding EMF-sensor was also detecting a monitor placed nearby. Though the power-consumption of the monitor is much lower than the printer, the monitor was on for a much longer duration, resulting in a several-fold increase in the energy-estimation.

Limitations

There are three main limitations to this approach. First, a local event detector still has the challenge associated with determining which internal state transitions are significant. In our system, we were focused on signaling large state changes, but often appliances could have a sequence of small internal states or continuously variable consumption. In these cases, a different type of detection algorithm may need to be investigated; perhaps one that analyzes the signals in the frequency domain.

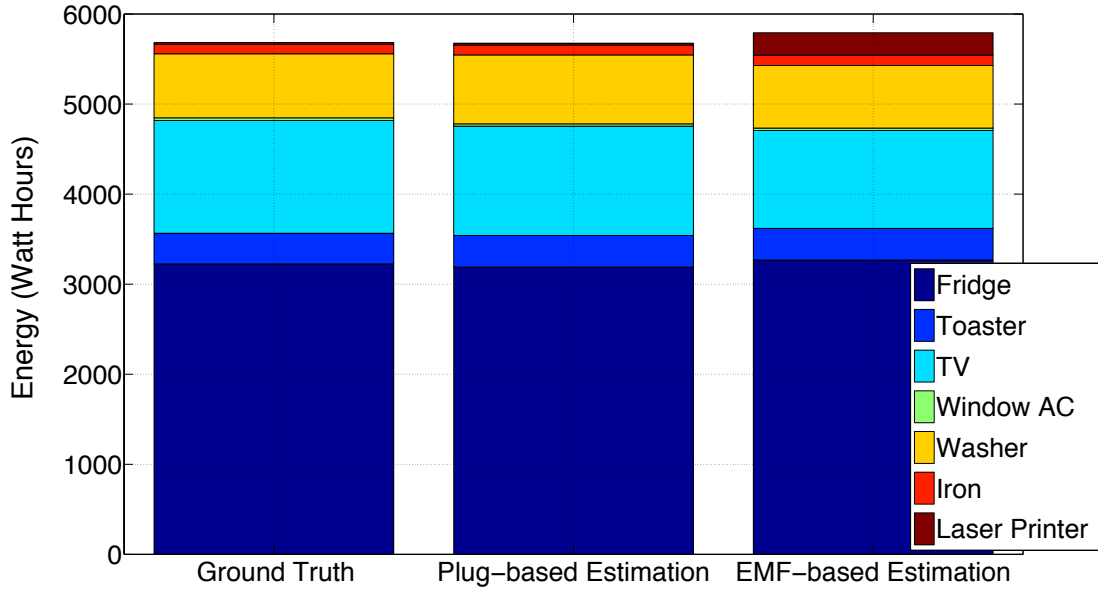


Figure 2.10: Energy estimation performance

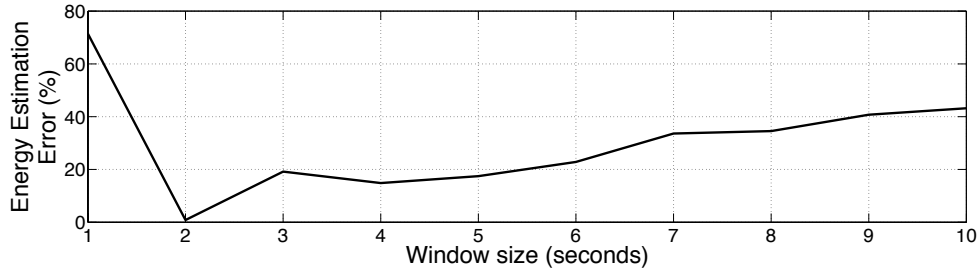


Figure 2.11: Error vs Mains sampling window

The second limitation is that these devices can suffer from cross-talk with different appliances if the devices or cabling are in close proximity of each other. We see this in our evaluation of certain devices present in our experiments. For example, the laser printer suffers from false positives. Part of optimizing the design is to build a device where the range is large enough to detect hard to reach wires, but small enough to minimize overhearing other signals. The third limitation is that the system cannot attribute static base-load values to appliances. In certain cases, the energy consumed while the appliance is supposedly *off* may be larger than its active energy. Despite these limitations, we believe that the EMF sensor and three-phase

meter provide a simple and low-cost alternative to appliance-level energy metering.

These studies done on custom-designed hardware lead us to believe that obtaining accurate event labels from appliances is possible through the use of external sensors. In Chapter 5, we utilize our energy estimation framework and study this problem in detail by utilizing off-the-shelf solutions for energy metering and external sensing.

Chapter 3

An Energy Estimation Framework

This chapter presents a framework for creating appliance models (as previously defined in Section 2.1.1) based on classification labels and aggregate power measurements that can help to relax many of simplistic assumptions that are currently used. The framework automatically builds models for appliances to perform energy estimation. From the communication theory analogy in Figure 1.5, this is similar to creating a *code inverter* for the received message. This inference is possible because of prior belief about how the appliance typically *codes* the message. Using the notation we used in Section 2.1.2, this Chapter presents a set of algorithmic techniques to estimate the function $g : \Xi \rightarrow \Psi$, with the time-series $\xi \in \Xi$ here denoting the time-series of event timestamps observed at the aggregate power data for a specific appliance. The output $\psi \in \Psi$ is the set of state transitions, states, and transition probabilities, i.e. $\{\Delta P_\psi, P_\psi, \text{ and } A_\psi\}$, for the appliance. This process is illustrated in Figure 3.1.

This chapter is based on the manuscript in [24]

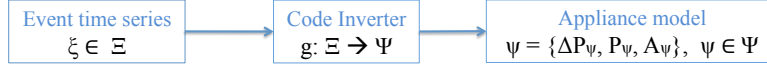


Figure 3.1: The input and output associated with the framework proposed in this Chapter.

The framework relies on feature extraction, clustering via affinity propagation, perturbation of extracted states to ensure that they mimic appliance behavior, creation of finite state models, correction of any errors in classification that might violate the model, and estimation of energy based on corrected labels. In this chapter, we evaluate our framework on 3 houses from standard datasets in the field and show that the framework can learn data-driven models based on event labels and use that to estimate energy with lower error margins (e.g., 1.1-42.3%) than when using the heuristic models used by others. A major component of the framework is the error correction algorithm, which we discuss in detail in Chapter 4. For simplicity and contrast, we use a primitive version of the error correction algorithm that relies on Maximum Likelihood Estimates (MLE) in this chapter.

3.1 State of the art in appliance behavior modeling

This section summarizes the efforts that have hitherto attempted to model appliance behavior with the goal of tackling different problems within NILM; it briefly discusses the contributions and potential drawbacks as well. Researchers have typically explained appliance behavior either through deterministic models learned from data (for e.g., FSMs) or using probabilistic models; some have used certain heuristic based methods as well.

3.1.1 Finite State Machine formulations

Hart briefly touched on the problem of learning appliance states and transitions from power differentials observed at the aggregate level using certain heuristics like the zero loop sum constraint (ZLSC), and the uniqueness constraint (UC) [14]. These will be expanded on later in the chapter. This method, however, is simplistic in that it assumes that power differences observed at mains do not include simultaneous events, and that the power differentials observed for each state transition are identical. His framework was a simple way of computing ΔP values at event-sites and matching them in chronological order, similar to the energy estimation framework without appliance models in Section 2.1.3. Although he focused on only validating two-state appliances, he conjectured that possible FSMs for all appliances could possibly be learnt directly from the sequence of ΔP events, the same way it is done in certain coding theory applications. However, the prevalence of errors and the need for perturbation (as discussed later) makes this possibility highly unlikely. Baranski and Voss [43, 44] presented a completely unsupervised method of estimating appliance behavior based on observed power differentials, and optimization of a quality function using genetic algorithms (GA). They calculate ΔP values based on a fuzzy clustering algorithm, and rely on the optimization step to automatically build the FSMs for appliances. The quality function in the optimization step assigns penalties according to the total sum of state transitions assigned to an appliance (forcing them to be close to zero), the relative frequency of each state transition, and the number of states for a particular appliance. The results from GA are filtered further using a variant of the Viterbi algorithm. Although this method is notable for acknowledging the need to model appliance behaviors, it has not been quantitatively validated so far, and the optimization algorithm involved does not guarantee con-

vergence to an optimal solution [19]. More recently, Streubel and Yang proposed the modeling of appliance behavior using FSM formulations by separating power traces of a single appliance into transients and steady-state modes [45]. They propose heuristics to identify segments that are transients (based on some prior modeling of transient distribution), and segments that are steady states from the power time series of a particular appliance. Then, distance metrics are used to calculate which segments represent the same operational state. This is analogous to computing the set ΔP for all appliances. This information then feeds into the final FSM of that particular appliance. Their approach remains to be validated, and the requirement that only one appliance has to be operating at a given time to model it is one of the drawbacks. In their more recent work, the authors in [15] have continued to refine their approach (e.g., [46, 47]). However, their main objective is to use these algorithms for direct disaggregation and thus is different from the objectives of this thesis.

3.1.2 Probabilistic prior models for appliances

Kim et al. modeled appliance behavior for their HMM models by learning the parameters for prior distributions of appliance states, on and off durations, and an appliance's state at a certain time of day using data [27]. But they assume appliances to be composed of two states only, which is not how many appliances behave. Johnson and Willksy manually set Gaussian priors for modeling power consumption levels, and a Negative Binomial prior for modeling the duration for which the state lasts, for the appliances that they are interested in [26]. These are set using some expert knowledge of how appliances are likely to behave and how they might impact the Hierarchical Dirichlet Process Hidden Semi-Markov Model (HDP-HSMM) formulation. Parson et al. follow a similar philosophy while modeling appliances in

their HMM framework; they expect an expert to provide generic models (including Gaussian priors for states, transitions, and respective probabilities) for appliances [3]. They also note that such parameters could be learnt from segments in aggregate data where only the appliance of interest is operating. Learning these models for appliances from data as opposed to relying on manual input (which by itself might not be possible or feasible in many cases) could potentially improve the performance of the respective algorithms.

3.1.3 Other ways of modeling appliances

Kolter et al. modeled appliance behavior based on real power transitions observed at the main circuit [28]. They employed an unsupervised method where "snippets" of the power signal were extracted when power consumption increases over some threshold and returns back to its original level. They then calculated pairwise transition probabilities between these snippets (probability of one snippet generating another) and used spectral clustering to group them together. Finally, the snippets that belonged to the same cluster were assumed to represent the behavior of one appliance, and were combined to create "motifs". An expert then manually labeled the motifs with labels of appliances they most likely represent. The authors acknowledge that this approach is limited to learning models for devices with relatively short cycles, and propose using an iterative implementation where the same technique is applied on the remaining "unexplained" part of the power signal. However, this version remains to be validated and could have limitations of its own.

As is evident, several researchers have indirectly incorporated the modeling of appliance behavior into their framework, but generalizable and completely data-

driven methods are lacking. The method proposed in [28] is perhaps the closest candidate as it looks to extract a general set of features from the aggregate data to build appliance-specific models automatically, yet it needs further validation. Among other reasons, data-driven appliance models are important because:

1. Energy estimation in the supervised setting fails without having proper appliance models. Even in cases where sub-state level classification is available, a few errors can drastically change energy estimates.
2. In supervised cases where only appliance labels are available, or equivalently, in the sensor-aided case, these models are the only cue the system has for estimating energy reasonably. Otherwise errors caused by simultaneous events, spikes, etc. can affect the results drastically.
3. Unsupervised models that utilize prior models of appliances (beyond two state) can benefit from knowing what states an appliance can exist in; in fact, as will be shown later, their accuracies can improve significantly.

In the following sections, we propose a method for learning appliance models automatically from data, and test it on data from three houses. We use publicly available standard datasets, namely REDD [37] and BLUED [2] for the purposes of our analysis. Table 3.1 summarizes the major properties of these datasets.

3.2 Introduction to the framework

Notationally, we formalize the process of appliance modeling in supervised NILM as follows. We denote the signal being monitored at the aggregate level at time t by $S[t]$. Typically, $S[t]$ is real power, but other signals like reactive power, apparent power, features of voltage, etc. are also possible candidates. If $T = \{t_1, t_2, \dots, t_m\}$

Table 3.1: Summary of the datasets used for evaluation in this chapter

Dataset	Sampling Frequency	Duration (days)	Appliances used	# of events
BLUED-1	12 kHz	7	Refrigerator	616
			Lamp	26
			TV	54
			Computer	45
			Laptop	14
			AV System	8
BLUED-2	12 kHz	7	Refrigerator	287
			Laptop	53
			TV	289
			Washer	1897
REDD-1	1 Hz	26	Oven	800
			Refrigerator	56
			Dishwasher	452
			Lighting	154
			Washer	65
			Microwave	443
			Dryer	236

represents the set of timestamps where events occur for a particular appliance, and $X = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \dots, \vec{x}_m\}$ is the set of corresponding features that are extracted from the events, then, appliance modeling is the process of learning, from those features and the power signal, the set of states of operation $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_k\}$ and the set of state changes $\Delta\Gamma = \{\Delta\Gamma_1, \Delta\Gamma_2, \Delta\Gamma_3, \dots, \Delta\Gamma_n\}$ possible for the appliance. In addition, the process of appliance modeling also involves using the time series sequence of events to learn the probabilities of transition between different state changes.

Typically, the set Γ represents real power consumed by the appliance during its different states of operation, and the set $\Delta\Gamma$ represents the step changes in real

power observed during state transitions. In order to estimate $\Delta\Gamma$, the step changes in real power observed at all events points $S[t_i]$ need to be computed. Such step changes (denoted by $\Delta\hat{\Gamma}_i$ for the event in timestamp t_i) are typically computed over some predefined time window around the detected event, and will be referred to as power differentials throughout this thesis. Various methods can be used to estimate the step change from each event. A very simplistic approach is to compute the difference between the first and last points of the window. So, if the time window is μ samples wide, then the power differential for the i^{th} event at timestamp t_i is given by:

$$\Delta\hat{\Gamma}_i = S[t_i + \frac{\mu}{2}] - S[t_i - \frac{\mu}{2}] \quad (3.1)$$

Even though this method can be significantly affected by noise, we use it throughout the remainder of the chapter and relegate further improvements as future work. Also throughout this chapter we perform analysis on real power signals only, and use a time window of 1 second. For real power computed at 60 Hz, this means $\mu = 60$. Accordingly, the states Γ are represented by P , power differentials $\Delta\Gamma$ are represented by ΔP , and the signal $S[t]$ is represented by $P[t]$.

Once ΔP values are calculated, a simplistic approach to obtain the power trace for an appliance given a ΔP sequence, would be to assume piece-wise constant power between each element of this sequence, along with an initial power value P_0 , and then perform the following sum to obtain the power value at timestamp t : $\hat{P}[t] = P_0 + \sum_{j=1}^{|R|} \Delta P_j$ where $|R|$ denotes the cardinality of set $R = \{t_j : t_j < t; \forall t_j \in T\}$. However, errors in the observed ΔP values (due to mislabeling, measurement noise or other reasons) would accumulate in the \hat{P} estimation. Furthermore, the piece-wise constant power consumption assumption does not always hold, which further

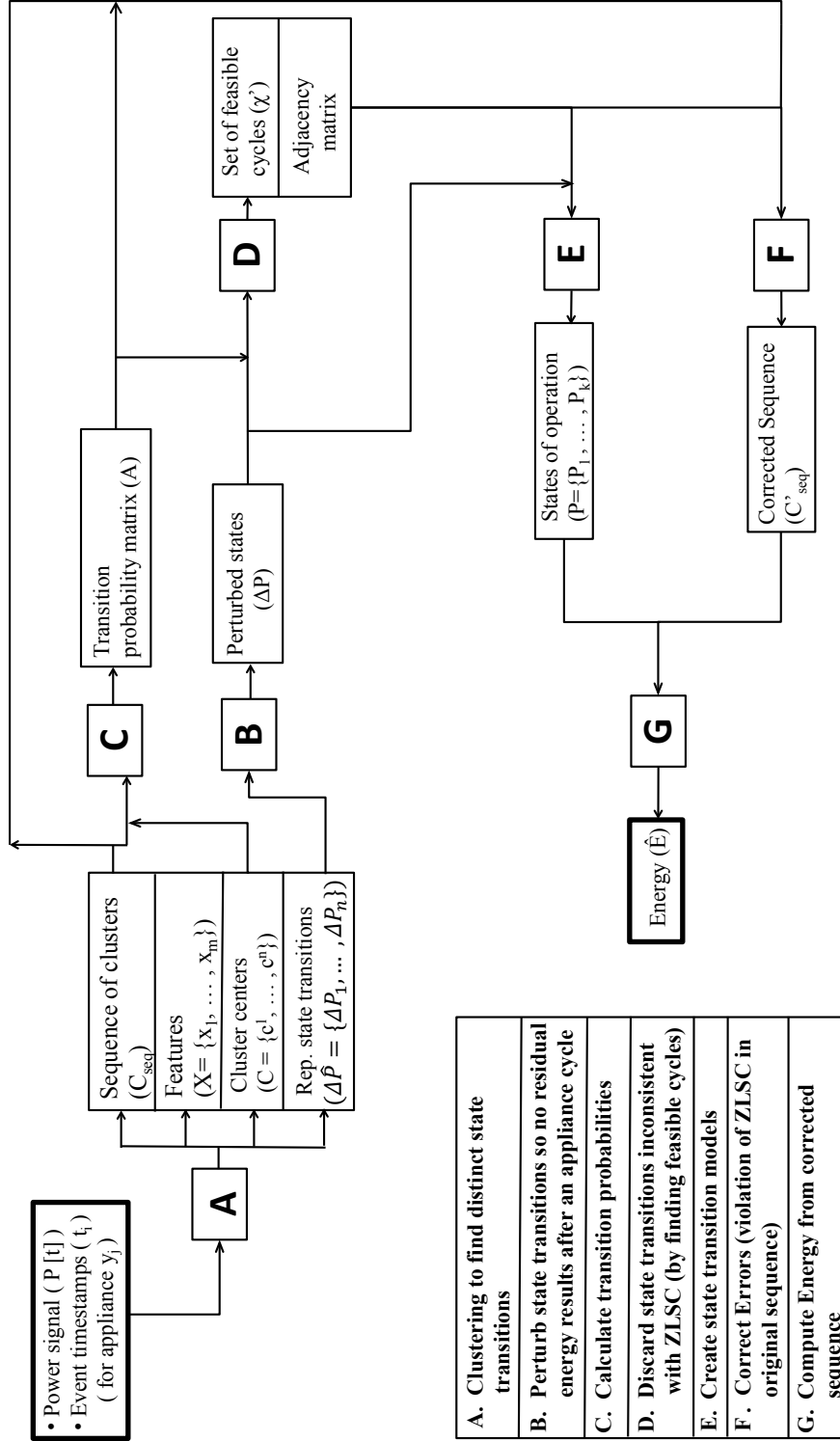


Figure 3.2: Summary of the major steps involved in the energy estimation framework presented in the chapter. Steps B and C, and steps E and F can be carried out in parallel, as their inputs are independent of each other.

increases the error accumulation. In essence, our framework redefines the values of the elements of ΔP (or, more generally $\Delta \Gamma$) so that error accumulation is avoided.

The following sections will elaborate on the major steps involved in the framework, and relate it to the problem of energy estimation within NILM. Figure 3.2 provides a graphical summary of major steps in the framework. All the computation mentioned in the framework was carried out in MATLAB 2013a, with a computer of 2.8 GHz i7 processor, and 4 GB RAM. The run times vary depending on the optimization process and the number of events associated with the appliance in question; on average the framework takes 10-15 seconds per appliance on the dataset we tested. We have made code and dataset available online in <https://github.com/sumangiri/EE-framework> under the Apache (2.0) license.

3.3 Classifying distinct state transitions

This first step, depicted as step A in Figure 3.2, attempts to infer the actual number and type of state transitions for an appliance from observed features and labels. In other words, what is the number of elements of ΔP ? Thus, the challenge here is to identify the true number of such step changes in real power that exist for an appliance, and assign each event with an appropriate ΔP value in a way that avoids error accumulation. Although this information could be extracted purely using the observed power differentials only, other features that are observed during events can also be used to enhance the analysis.

This is a problem that lends itself naturally to the idea of clustering. Several features that are associated with events themselves can be exploited to estimate the

true state transitions. Ideally, such features should be extractable from the aggregate power data itself. In this chapter, we explore the P-Q plane (step change in real and reactive power) as done by Hart [14], the real power transients observed at mains as analyzed by Norford et al. [15], and the projections of the transients onto the first two principal component axes of the set of all transients. Each resulting cluster represents a state transition that the appliance can make. All the events associated with the appliance are assigned the labels of the cluster that they are grouped into. Depending on the sampling rate at which the data is being captured, this assignment may not always be reflective of reality. This is because for lower sampling rates, there is a higher possibility of simultaneous events (from multiple appliances). However, it is reasonable to assume that such events are still sparse, and steps such as de-noising and error correction that are explained later in the framework can counter the adverse effect such faulty assignments may have on the energy estimation process. Moreover, for notational convenience, in the remainder of the chapter, we assume that appliances are being evaluated one at a time. Hence, unless specifically noted, all representations are for events collected from one particular appliance.

Notationally, features extracted from an event occurring at timestamp t_i will be denoted as \vec{x}_i , where $\vec{x}_i \in \mathbb{R}^b$ is an b dimensional vector. Let the set $X = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \dots, \vec{x}_m\}$ denote the set of features extracted from events occurring at timestamps $\{t_1, t_2, t_3, \dots, t_m\}$, with $1 \leq i \leq m, \forall i \in \mathbb{N}$. The clustering step uses a clustering algorithm g to map the set of features X to a set of cluster labels C , where $C = \{c^{(1)}, c^{(2)}, \dots, c^{(n)}\}$, with $n \in \mathbb{N}$. The result of the mapping $g : X \mapsto C$ is an ordered sequence of cluster transitions $(g(\vec{x}_1), g(\vec{x}_2), g(\vec{x}_3), \dots, g(\vec{x}_m))$, denoted by

$C_{seq} = (c_1, c_2, c_3, \dots, c_m)$ where $c_i \in C$. For clustering, various options are available, some of which we explore and evaluate in this section. Although the algorithms will be briefly mentioned in text, we invite the interested reader to explore Xu et al.'s review for a more detailed description [48]. The most popular technique, perhaps, is k-means, which is based on centroid calculations for clusters and divides data based on distances to clusters. The drawback of this method is that the number of clusters has to be known in advance for it to work properly, the end value depends upon initialization, and it is easily affected by noisy data. These shortcomings can be overcome by techniques such as gap statistics, which find the optimal value for the number of cluster [49]. Methods like k-means ++ are available, which provide the optimal set of starting points [50]; another modification to the standard algorithm, namely weighted k-means, assigns low weights to data that have low-confidence associated with them, thus tackling the noise issue [51]. We augment the standard k-means with these modifications in our use case, and denote it by k-means*. Other methods like DBSCAN and OPTICS that rely on relative spatial density to form clustering, conveniently termed as density based clustering methods, were also explored [52, 53]. These are desirable because they are more robust to noise and do not require the number of clusters beforehand; in turn, they need the minimum number of points needed to form a cluster, and minimum distance between points as the input. To ensure that all clustering forms were explored, we also looked at spectral clustering (based on similarity measures between points), and affinity propagation (relies on message passing between data points)[54, 55]. Typically, spectral clustering requires the number of clusters as an input, but in this case, we used a self-tuning version of the algorithm that exploits the eigenvalues of the affinity matrix to find the number of clusters [56].

To evaluate the suitability of these clustering methods in finding the right number of clusters and appropriate exemplars for appliance models, we tested them with simulated data. 1000 appliances were simulated with each appliance having anywhere from 2-10 states (as is typically the case with appliances). The number was assigned randomly according to a uniform distribution between 2 and 10. After this, a mean and variance was assigned as the representative value to each state. This, in turn, was sampled from a uniform distribution between 50 and 2500, with a variance between 1 and 400, as state transitions in most appliances typically lie within this range. Finally, 100 samples were drawn for each state transition for each appliance from a Gaussian distribution with the mean and variance assigned to it. This comprised as the ground truth data. For evaluating the suitability of different algorithms in clustering this data, we devised certain metrics that are more suited to the end goal of energy estimation. First, we evaluate the model created, and the factors that matter here are the strength of the algorithm in predicting the number of clusters correctly, and in picking the exemplars correctly. We evaluate this with weighted model error, which, in turn, is a function of value prediction error and cluster number prediction error. They are described as follows :

$$\mathbf{e}_n = \frac{|\hat{n} - n_{true}|}{n_{true}}$$

Here, \mathbf{e}_n is the error in predicting the number of clusters. \hat{n} is the number of clusters estimated by the algorithm, and n_{true} is the true number of clusters.

$$\mathbf{e}_{\Delta P} = \sum_i \frac{|\Delta \hat{P}_i - \Delta P_i|}{\Delta P_i}$$

Here, $\mathbf{e}_{\Delta P}$ is the error made by the algorithm in predicting the means of the state transitions correctly. ΔP_i is the mean value of the state transition for an appliance and $\Delta \hat{P}_i$ is the value in the predicted exemplar set that is closest to the mean.

Finally, the model error is calculated as the weighted sum of value prediction error and cluster number prediction error.

$$\mathbf{e}_{model} = w_1 \mathbf{e}_n + w_2 \mathbf{e}_{\Delta P}$$

We used $w_1 = 0.5$ and $w_2 = 0.5$ for our evaluation. Figure 3.3 depicts the values of \mathbf{e}_{model} for different algorithms. The average value was taken for appliances with the same number of states. Affinity propagation outperformed other clustering algorithms, both in terms of predicting the true number of state transitions, and the mean values for state transitions.

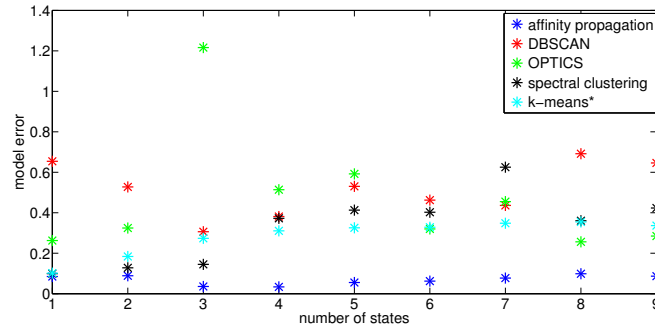


Figure 3.3: Weighted sum of errors in predicting the number of clusters, and the values for exemplars for state transitions for different algorithms.

To evaluate the effectiveness of the clustering algorithms in correctly assigning true state transition values to observed state transitions, we also evaluated another error metric that measures the total RMS error between predicted states and true states. This metric is especially useful as it gives insight into how a clustering al-

gorithm will affect the energy estimation process. The metric was defined as follows:

$$\mathbf{e}_{RMS} = \frac{\|\Delta\vec{P} - \Delta\hat{P}\|}{\|\Delta\vec{P}\|}$$

Here, $\Delta\vec{P}$ is the vector form of the ordered sequence with mean values for state transitions associated with an appliance. $\Delta\hat{P}$ is the estimated value assigned by clustering algorithms. Figure 3.4 shows the results of this evaluation on our simulation data. As before, the average value was taken for appliances with the same number of states to be depicted in the graph. Affinity propagation outperforms other algorithms here as well. As a result, we chose it as our choice of clustering algorithm for further analysis.

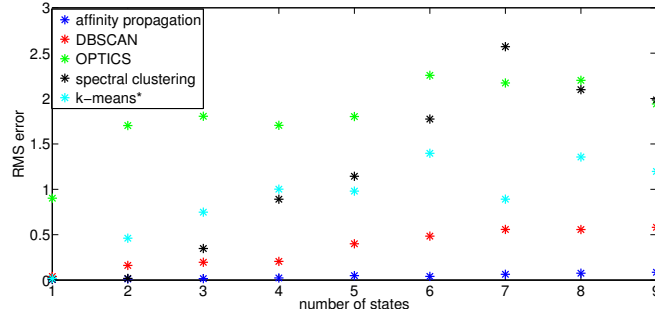


Figure 3.4: RMS error evaluated on simulated data for different clustering algorithms.

Affinity propagation relies on the concept of message passing between data points. For a good introduction to the AP clustering algorithm and its applications, we recommend reading references [57] and [55]. The inputs to this clustering algorithm are similarities between datapoints $s(i, k)$ defined as the suitability of datapoint k to act as exemplar for datapoint i . Self-similarities, i.e. $s(k, k)$, also known as preference, control the suitability of a data-point to be an exemplar. We

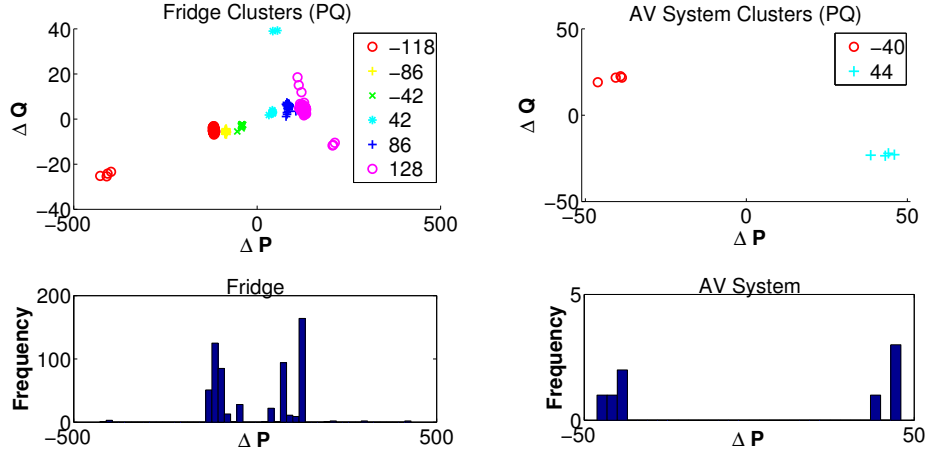


Figure 3.5: Clustering results on P-Q features using affinity propagation for the Refrigerator and AV system on BLUED. The labels show the mean ΔP values of each cluster. The histograms show the frequency of ΔP 's associated with the appliances.

use Euclidean distance as the similarity metric, and all preferences are set to a common value (equally likely to be exemplars). Figure 3.5 shows the results of using affinity propagation on P-Q features for the Refrigerator and AV system in BLUED.

Once clustering is done, the effect of noisy data and misclassifications is negated to some extent by throwing away small clusters. Such noisy features could have resulted from simultaneous events, incorrect labeling, or the use of inappropriate window sizes during feature extraction. A simple heuristic is used here, where any cluster with less than 3 elements is assumed to be noise. This is based on empirical observation of noisy features in the clustering step, though a more thorough analysis of other de-noising strategies can be performed in the future. The new cluster labels for such noisy events are assigned according to the closest valid cluster in terms of Euclidean distance. For notational convenience, we denote the de-noised clusters and the resulting sequence with the same notation as before.

After de-noising the clusters, we then calculate the power differential associated with each feature. We represent the power differential observed during the event corresponding to feature \vec{x}_j with ΔP_j . The mean power differential value (step change) of each cluster is first assumed to be the representative value of that state transition. So,

$$\Delta P_{c^{(i)}} = \frac{1}{k} \sum_{j=1}^k (\Delta P_j); \forall j : g(\vec{x}_j) = c^{(i)} \quad (3.2)$$

is the representative value of power state differential for all events in cluster $c^{(i)}$, with k being the total number of elements in the cluster.

The representative state-transition values calculated in this way will typically not conform to the way appliances draw power. For instance, say we have a refrigerator that has six state transitions: +86 W (compressor on), +43 W (door-light on), +129 W (both on), -86 W (compressor off), -43 W (door-light off), -129 W (both off). Now, owing to both the electromechanical operation of the appliance, and potentially erroneous extraction of power step changes, or misclassification, as discussed earlier, the cluster centers may be calculated as (+85.9 W (compressor on), +42.3 W (door-light on), +128.3 W (both on), -85.8 W (compressor off), -42.2 W (door-light off), -118.3 W (both off)). A cycle is assumed to have happened when an appliance operating at any state, undergoes changes in operational modes, and returns to the same state. So, 86+43-129 is one possible cycle as the power consumption returns from zero to zero. If the values given by clustering algorithm are taken without modification, for each cycle like the one just mentioned (86+43-129), a value of (85.9+42.2-118.3) will be recorded, and an error of 9 W will accumulate throughout the remainder of the power trace. This will negatively impact the energy estimation process. To avoid such accumulation, we introduce the concept of perturbation in

the next section.

3.4 Perturbance

A central idea to the perturbation framework is that of the Zero Loop Sum Constraint (ZLSC), introduced by Hart [14]. It stipulates that the sum of all power transitions over a valid cycle of an appliance should be zero. We utilize this concept to define a cycle as a set of state transitions that satisfy this constraint with a slack of ϵ . The cluster centers are perturbed in such a way that the number of cycles is maximized. This captures typical appliance behavior during state transitions for multi-state appliances. Step B in Figure 3.2 shows the inputs and outputs of the perturbation step.

The perturbation is done so that the total sum of all positive and negative ΔP values sum up to zero. We model this new assumption as a constraint in an optimization formulation, explained below, to find the values with which the representative state transitions from the previous steps need to be perturbed by so that they follow the ZLSC. We begin with definitions for the terms used in the optimization.

Let $\Delta\hat{P} = \{\Delta P_{c(1)}, \Delta P_{c(2)}, \Delta P_{c(3)}, \dots, \Delta P_{c(n)}\}$ be a set of representative state transitions. For notational convenience, it will be denoted simply as: $\Delta\hat{P} = \{\Delta P_1, \Delta P_2, \Delta P_3, \dots, \Delta P_n\}$. Also, let $\Delta\hat{P}_{pos} = \{\Delta P_i \mid \Delta P_i \in \Delta\hat{P}, \Delta P_i > 0\}$ be the set of all positive state transitions within $\Delta\hat{P}$, and a similar definition for the negative state transitions, $\Delta\hat{P}_{neg}$.

U is the power set of all subsets of $\Delta\hat{P}$ such that $\forall U_i \in U$, there is at least one

element $u_i \in U_i \mid u_i \in \Delta\hat{P}_{pos}$ and at least one element $u_j \in U_i \mid u_j \in \Delta\hat{P}_{neg}$. This is meant to represent all the possible cycles that can occur within the appliance. Using the basic principles of cardinality of subsets, it can be seen that the size of the set U is $(2|\Delta\hat{P}_{pos}| - 1)(2|\Delta\hat{P}_{neg}| - 1)$, where $|\Delta\hat{P}_{pos}|$ represents the cardinality of the set $\Delta\hat{P}_{pos}$ and likewise for $|\Delta\hat{P}_{neg}|$.

$f : \mathbb{R} \mapsto \{0, 1\}$ is a function that takes in all elements of a given set W , where $W \subset \mathbb{R}$, as an input and maps it to either 0 or 1 according to the following conditional:

$$f(W) = \begin{cases} 1 & \text{if } \sum_j w_j < \epsilon \quad \forall w_j \in W; \quad \epsilon \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

Logically, this function amounts to a counter that counts whether or not a possible permutation is a cycle that can theoretically exist in the appliance, and it does this by enforcing the ZLSC with a maximum error of ϵ .

The set $E = \{e_1, e_2, \dots, e_n \mid e_i \in \mathbb{R}\}$ is a set of perturbations that can be applied to elements in $\Delta\hat{P}$ so that they follow the ZLSC.

Let E_i denote a set that is the same size as U_i , and has direct correspondence with elements of U_i . So, if $U_i = \{u_a, u_{a+1}, \dots, u_k \mid u_i \in U\}$, E_i is the set of corrections that correspond to those elements, i.e., $E_i = \{e_a, e_{a+1}, \dots, e_k\}$

Now the optimization framework is:

$$\begin{aligned}
& \arg \max_E \quad \sum_i f(U_i + E_i) \quad \forall U_i \in U; E_i \subset E \\
& \text{s.t. } -c < e_i < c; \forall e_i \in E
\end{aligned} \tag{3.4}$$

This amounts to altering the representative state transitions in such a way that the number of resulting cycles is maximized. Here c is the perturbation threshold that controls how flexible the alteration of representative state transitions is. Clearly, this is a combinatorial optimization problem and is NP-hard. We solve it by creating the following convex approximation of the objective function instead.

$$\begin{aligned}
& \arg \min_{e_i} \quad -|f(\vec{\mathbf{u}}_j + \vec{\mathbf{e}}_j)|_1 + \lambda |\vec{\mathbf{e}}|_1; \text{ where } \vec{\mathbf{e}} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}; e_i \in E_i \\
& \text{s.t. } -c < e_i < c; \quad \vec{\mathbf{u}}_j + \vec{\mathbf{e}}_j = 0 \quad \forall \vec{\mathbf{u}}_j \mid f(\vec{\mathbf{u}}_j) = 1
\end{aligned} \tag{3.5}$$

Here $\vec{\mathbf{u}}_j$ represents the vector form of the set U_j and likewise for $\vec{\mathbf{e}}_j$. λ is a regularizing parameter that forces the error terms to be sparse and hence controls unnecessary perturbations. Table 3.2 shows the result of this analysis on representative state transitions extracted from appliances in three houses in three different datasets. Resulting state transitions that are below 5 Watts are removed from analysis.

Table 3.2: Results of perturbation on state transition values extracted after clustering in different datasets

House	Appliances (label)	Unperturbed state transitions (after clustering) [Watts]	Perturbed state transitions [Watts]
BLUED-1	Refrigerator (111)	{85.9, 128.3, 42.3, -42.2, -85.8, -118.3}	{86, 129, 43, -43, -86, -129}
	Lamp (101)	{27.1, -34.6}	{31, -31}
	TV (129)	{170.3, -224.4}	{197, -197}
	Computer (118)	{27.2, -26.9}	{27, -27}
	Laptop (120)	{57.2, -47.9}	{53, -53}
	AV Syst.(112)	{43.6, -39.8}	{42, -42}
BLUED-2	Refrigerator (111)	{122.9, 173.8, -106.1, -274.0}	{106, 171, -106, -277}
	Laptop (120)	{2, 18.9, 44.1, -23.9, -50.9}	{21, 48, -21, -48}
	TV (129)	{36.1, -24.3, -47.2}	{42, -42}
	Washer (183)	{15.5, 110.6, 149.9, 336.5, -13.1, -96.2, -191.7, -361.5}	{103, 164, 350, -103, -164, -350}
REDD-1	Oven (3)	{2471.3, -2443.5}	{2457, -2457}
	Refrigerator (5)	{34.6, 222.9, -37.2, -173.8}	{36, 196, -36, -196}
	Dishwasher (6)	{207.7, 930.4, 1145.7, -40.5, -200.9, -380.2, -871.5, -1117.1 }	{205, 1037, 1152, -40, -205, -271, -765, -1113, -197}
	Lighting (9)	{29.6, 79.3, -30.1, -77.1}	{31, 78, -31, -78}
	Washer (10)	{692.3, -650.6}	{671, -671}
	Microwave (11)	{1512.0, -1511.2}	{1512, -1512}
	Dryer (120)	{2832.4, -2676.3}	{2754, -2754}

3.5 Creating transition probability matrices

This step deals with the calculation of probability of transition between all of the state changes extracted from the previous step. As will be shown in subsequent sections, this matrix is required for calculating the set of possible cycles that can exist in the appliance, and also for correcting errors in the sequence C_{seq} . Figure 3.2 (referred to as step C) illustrates the steps in which the transition probability matrix is used, and the inputs and outputs of the step. It is of size $n \times n$, where n is the number of possible state transitions for the appliance. To calculate it, a first order Markovian transition probability matrix is populated using the time series information about transitions between clusters. This is done by counting the number of state transitions from one state to another and subsequently normalizing the counts. This is denoted by the matrix A of dimensions $n \times n$, where an element in row i and column j , i.e. a_{ij} , denotes the probability of transition from cluster $c^{(i)}$ to $c^{(j)}$.

3.5.1 Finding feasible cycles

The transition probability matrix populated directly from data can allow state-transitions that are not possible (and violate the ZLSC). The goal of this step is to find and discard all such violations and preserve only the feasible cycles for the appliance so that an appropriate adjacency matrix can be created. This step is denoted as step D in Figure 3.2. As will be shown in Section 3.7, the adjacency matrix created this way is central to identifying errors in the time series of cluster assignments. We use the perturbed states to represent distinct edges in a state transition diagram. Using the edges, and transition probabilities learned from the event time series, a list of feasible cycles are extracted. To do this, the states are

modeled as the nodes (vertices) of a graph, and the transition probability matrix as an adjacency matrix. The goal is equivalent to finding the strongly connected components of a directed graph, and extracting all the possible sub-graphs. Several solutions have been proposed to solve this problem in graph theory, and we use a modification of Tarjan's strongly connected algorithm to extract all the existing cycles [58]. Once the cycles are extracted, they are checked for feasibility of ZLSC. The ones that violate ZLSC are discarded. Additionally, a check is placed which enforces the total sum of all state transitions for each feasible cycle to be more than or equal to zero at any given point. The diagonal elements of the adjacency matrix are set to zero since we are not interested in modeling steady state operation.

Notationally, the set of perturbed state transitions, i.e., $\Delta P = \{\Delta P_1, \dots, \Delta P_n\}$, denote the nodes of a graph. Let the set of all possible cycles be denoted by $\chi = \{\chi_1, \dots, \chi_k\}$, where each $\chi_i \subset \Delta P$ is an ordered set of state transitions (ΔP_i) such that the transition probability $a_{i,i+1} \neq 0$ for all $i \mid \Delta P_i \in \chi_i$. The set of feasible cycles $\chi' = \{\chi'_1, \dots, \chi'_k\}$ is a subset of the set of all cycles that satisfy the ZLSC. Notationally, $\chi' \subset \chi \mid \forall \chi'_i \in \chi', f(\chi'_i) = 0$. Here, the function f is the same function defined in Equation 3.3 in Section 3.4.

3.6 Creating state transition models

The operations in Section 3.5.1 are trivial for two state appliances, but for appliances with multiple states, this process can reduce the number of cycles considerably. For instance, for the Refrigerator in BLUED 1, the number of feasible cycles reduces from 168 to 24. The adjacency matrix for the Refrigerator in BLUED 1 as calculated from data and transition probabilities is:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The values for state transitions are presented in Table 3.2. Using the strongly connected algorithm, the number of cycles in this graph amounts to 168, but only 24 of them follow the ZLSC. Using the feasible cycles, a new adjacency matrix is created that accurately reflects the possible state transitions. A state transition diagram is then created for an appliance based on this information. Figure 3.6 displays state transition diagrams for the Refrigerator and TV. These diagrams represent the possible transitions between state changes as observed from the time series of events. This is different from the FSM diagram of the device, which basically shows the different states (as opposed to state transitions) that an appliance can operate in. Figure 3.7 details the FSMs extracted from the state transition diagrams. Although the FSMs do not have a direct role in energy estimation framework presented here¹, they provide information on how an appliance operates. For models that are not based on differential HMM and rely on the current state of the observed data to estimate the states of the appliances, such FSMs are needed. For instance, standard unsupervised NILM techniques like Additive Factorial Hidden Markov Models (AFHMM), need information about the states that an appliance can exist in, and the transition probabilities between such states [28]. For such cases, an FSM model can improve the accuracy of the inference process.

We extract FSMs using the set of feasible cycles. For each feasible cycle $\chi'_i \in \chi'$, the sum of state transitions are noted to get the possible states that an appliance can exist in. For instance, if $\chi'_i = \{\Delta P_a, \Delta P_{a+1}, \dots, \Delta P_{a+k}\}$ denotes one such ordered

¹They provide the upper limit for power consumption of an appliance, which, as is explained later, can be used as an additional heuristic during energy estimation.

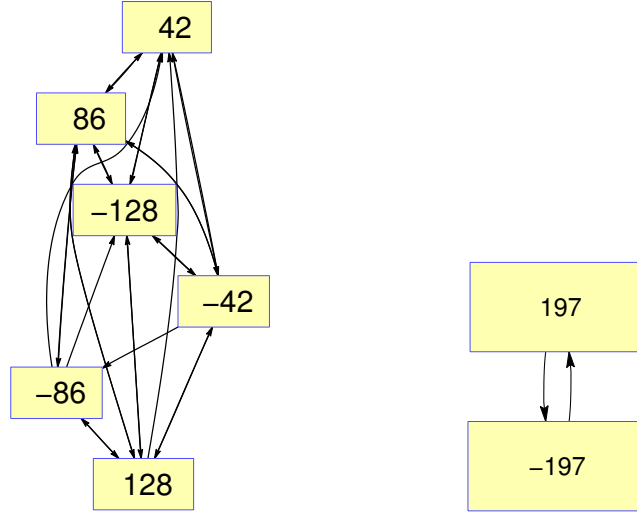


Figure 3.6: State-transition diagrams for a Refrigerator (left) and TV (right) learnt from the time series of events.

set of feasible cycles (with $a, k \in \mathbb{N}$), then the set of states extracted from χ'_i are given by:

$$\chi'_i = \left\{ \sum_{i=a}^{a+1} \Delta P_i, \sum_{i=a}^{a+2} \Delta P_i, \dots, \sum_{i=a}^{a+k} \Delta P_i \right\} \quad (3.6)$$

This process is repeated for all such feasible sets χ'_i . The redundant resulting states are removed to get the final set of states $P = \{P_1, \dots, P_k\}$. States that are too close to each other (i.e., differing by less than the minimum state transition possible), are averaged to represent one single state. Figure 3.2 shows how this step (labelled step E) fits into the whole energy estimation framework.

3.7 Correcting errors

A simple error correction heuristic is applied to correct state transitions in the original events time-series that violated the models learnt about the appliance (step F, in Figure 3.2). Any state transition that had a zero probability of transition from its previous state is replaced by the most likely state transition (learnt in Section

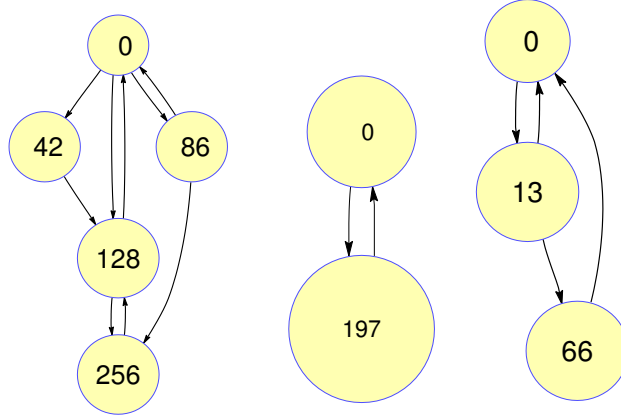


Figure 3.7: FSM diagram for a Refrigerator from BLUED-1 (left), TV from BLUED-1 (center) and Laptop from BLUED-2 (right) created using adjacency matrices

3.5). Correcting errors in sequences generated by probabilistic FSMs is a complex research problem, and we discuss it in detail in Chapter 4. Here, however, a simple algorithm is deployed to automatically correct for such errors, as our primary goal in this chapter to make a case for the need of a framework for energy estimation. Mathematically, the heuristic used amounts to the following:

If $C_{seq} = (c_1, c_2, c_3, \dots, c_m)$ represents the ordered sequence of cluster assignments as defined previously such that c_i denotes the state transition that occurs at event associated with timestamp t_i . If $c_{i-1} = c^{(i)}$; $c_i = c^{(j)}$; $\forall c^{(i)} \in C$, an error occurs whenever $a_{ij} = 0$. The correction applied is as follows:

$$c_{(i)} = \{c^x : x = \arg \max_x (a_{ix}); 1 \leq x \leq n; a_{ij} \in A\}. \quad (3.7)$$

3.8 Computing energy

Once the appliance models are created, and errors in transitions are corrected for, energy is estimated assuming power consumption between events to be piece-

wise constants. This is the final step of the framework (step G in Figure 3.2). The total power consumption at any event point is the net sum of all transitions leading up to that point. Mathematically, power at any time point for an appliance is given by

$$\hat{P}(t) = \sum_{i=1}^t [h(c_i)] \quad (3.8)$$

where $c_i \in C$ denotes the state transition that occurs at time t_i . The function h maps the cluster label c_i to the representative power transition $(\Delta P_{ij} + e_{ij})$. By definition, energy is calculated as

$$\hat{E} = \sum_t \hat{P}(t) \quad (3.9)$$

As an additional heuristic, a check is put in place at each event timestamp t_i in order to ensure that the value of power consumption does not exceed the power level of the maximum state in the FSM derived for the appliance. If it happens, the power value is simply set to the maximum value for that device to avoid accumulation of errors. Similarly the power value has a lower bound of zero, and any value that amounts to less than zero is set to zero, i.e., if $P = \{P_1, \dots, P_k\}$ denote the finite states that an appliance can exist in, then the following checks are put in place:

$$\hat{P}(t) = \begin{cases} \max(P) & \text{if } \hat{P}(t) > \max(P) \\ 0 & \text{if } \hat{P}(t) < 0 \end{cases} \quad (3.10)$$

Table 3.3 shows the results of energy estimation using the aforementioned framework on a few datasets. We use the percentage error in energy estimation as the metric for evaluating the framework $(\frac{\hat{E}-E}{E} \times 100\%)$. Here, E refers to the ground-truth energy. The features used for calculation of clusters are different for each.

For the house in BLUED-1, both real and reactive power differentials are available and are labeled reasonably accurately. So we use a 2-D feature vector $(\Delta P, \Delta Q)$ for this house. For BLUED-2, we found the labeled reactive power to be unreliable, and hence the projection of the real transient on the first two PC axes are used as features. It is worth mentioning here that both these houses have high frequency data sampled at mains (real power was sampled at 60 Hz). For the third house, we use house 1 of REDD. This is different from the previous dataset as the power is sampled at 1 Hz only, and only the real power information is available. So, we use ΔP as the feature vector for this house.

To better illustrate how the framework operates on actual data, we take the example of the refrigerator in BLUED. Figure 3.8 shows how different steps of the energy estimation framework would work on real data. For illustration purposes, we pick a segment between 18:00 and 20:00 of a particular day, and show the features extracted from events during that time. One of the events during this time is mislabeled, and subsequently corrected for during error correction (highlighted in red).

3.8.1 Further evaluation with simulated data

In this section, we evaluate the framework further on simulated data. Week-long power traces for five appliances, sampled at 1 Hz, are generated assuming a Markovian model; the appliances, assumptions of transition probabilities, and states are shown in Table 3.4. The **hmmgenerate** function in MATLAB is used to create the power traces; emission probabilities (probability of a hidden state generating the observed state) were set to identity matrices as the hidden states and observed states

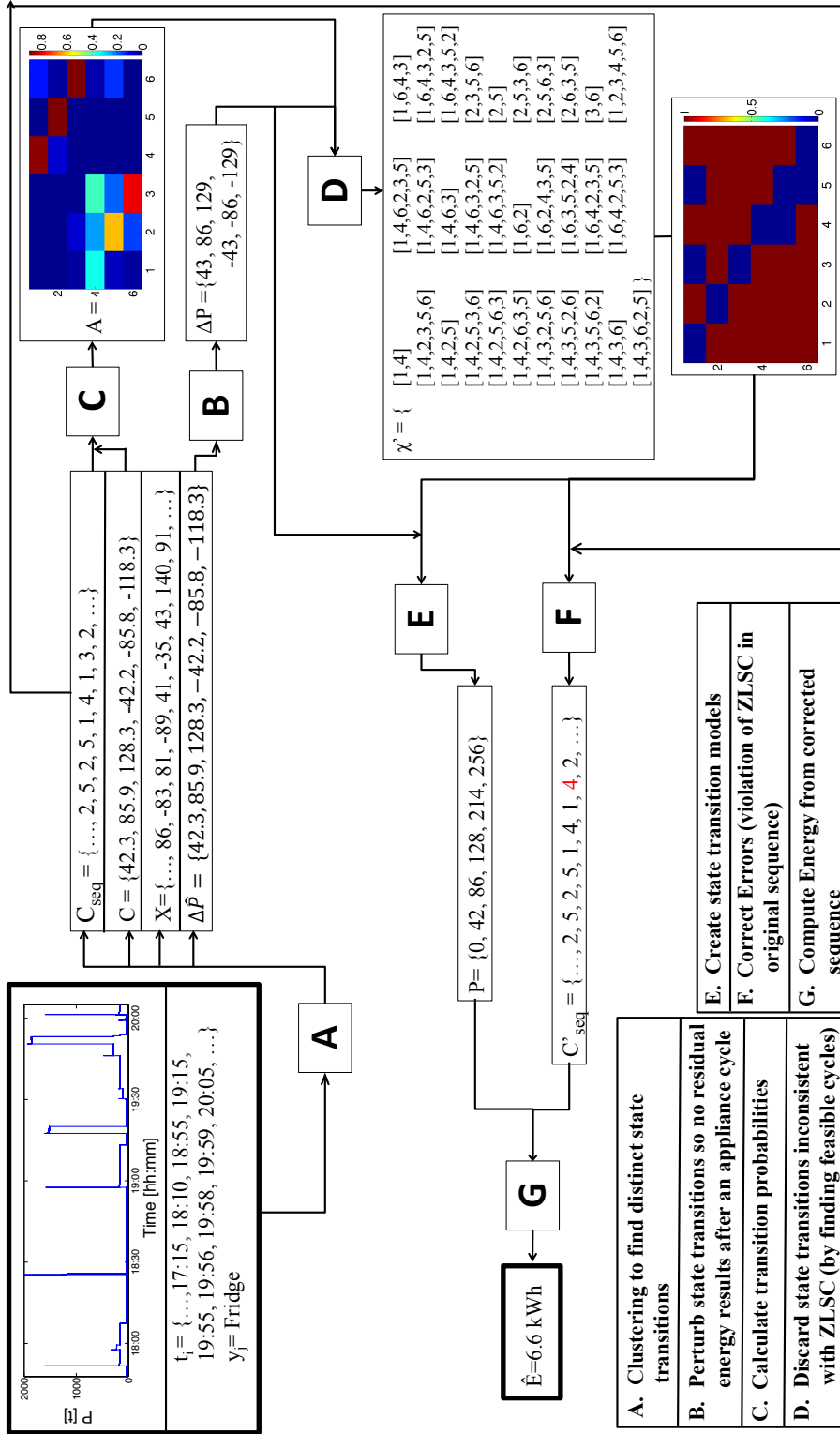


Figure 3.8: Flow of data through the framework pipeline for the refrigerator in BLUED-1. For illustration purposes, we pick a segment between 18:00 and 20:00, and note the actual values of C_{seq} , C'_{seq} and X for that portion only. The cluster number highlighted in red shows the error that was corrected (from transition 3 to 4) in step F.

are the same for this particular use case. For each iteration, and for each appliance, the values for the states are sampled from a Gaussian with the mentioned mean and variance. The models are taken from the generic prior models described for these appliances in [3]². Next, we add Gaussian white noise with a signal to noise ratio (SNR) of 1 dB; this decision is based on the kind of noise we observed at plug-level data for BLUED. Before adding the noise, we ran an event-detector on these individual power traces to find the exact event timestamps for ground truth. Given that the minimum state-transition value is known and that there are no simultaneous events, event detection at plug-level is a fairly trivial process. Finally, for each appliance trace, we randomly sample a week-long power segment (collected at 1 Hz) from the available houses in REDD, and added the two together. This gave us a realistic power trace with perfect ground truth for one appliance. We ran our framework (Section 3.3 -Section 3.8) on this power trace to estimate the energy consumed by that appliance. We repeated this process a 100 times for all 5 appliances, hence simulating a case with 100 houses where ground truth data was available for 5 major appliance types. The results, in terms of average percentage error in energy estimated for the week for 100 houses, are summarized in Table 3.5.

3.9 Discussion

3.9.1 Discussion of results

The results in Table 3.3 show that the energy estimation framework works reasonably well for the three houses analyzed. Appliances that are traditionally considered difficult for event-based approaches, like laptop and TV, were selected for evaluation. The superior accuracies for BLUED-1 are probably because of more ac-

²The transition probabilities are altered to properly resemble the case for 1 Hz sampling

curate labels. The labels in BLUED-2 are not as accurate as BLUED-1 because of manual errors in labeling. Out of the three houses, REDD-1 performs poorly. This is expected because (1) the events were manually labeled by us for the purposes of this work and given that REDD-1 has measurements for almost a month, there is a higher probability of mislabeling; (2) only ΔP was available as the feature for estimating states. Also, given the low frequency of the dataset (as compared to the other two datasets), there is a higher probability that the time window selected for ΔP has multiple events occurring within it. To test how much of the error was due to poor labeling versus simultaneous events, we performed this analysis on simulated data mimicking REDD data as closely as possible in the simulations (Table 3.4). The fact that the errors decrease significantly indicate that most of the contributions in the error for REDD is likely due to manual errors in labeling. Appliances like the microwave still continue to have large errors due to the fact that they are used during times when simultaneous events are most likely in low frequency data. Moreover, supervised and event-based NILM is typically done on higher frequency data, so results BLUED-1 is indicative of how this framework would perform in such scenarios. A better error correcting algorithm (Section 3.7) could probably improve the performance of BLUED-2 given the presence of incorrect or missing labels.

3.9.2 Discussion of framework

The proposed framework has room for improvement, which we discuss here and leave its implementation as future work. Foremost, we believe the need for automated appliance modeling in NILM, which is the primary goal of this chapter, has been made clear through our analysis in previous sections; hence, in this section we provide our thoughts on how the framework could be improved.

One of the more important parameters that controls how representative state transition values, energy estimates, and even cluster centers are calculated is the ΔP parameter. The extraction of ΔP is dependent on the time window that is considered. If the window is larger than necessary, then it can include events from appliances that are not of interest, hence skewing the results. On the flip side, if it is too small, it might not capture the actual ΔP because the transients might not have stabilized into steady-state within that window span. Although we used a generic time window of 1 second for our analysis, perhaps a more robust method can exist that extracts this parameter directly from data by maximizing the window size while constraining the number of events seen in each window to one. An ideal implementation would use different window sizes for different appliances in a house based on values learnt from data.

Another major assumption in the framework is that the convex approximation closely approximates the optimization framework in the perturbation step. Although the results seem to make sense empirically (Table 3.2), evaluating this step exhaustively and identifying the cases where it is likely to fail is left as future work. Similarly, there might be other formulations of the optimization framework, for instance, mixed integer programs, which can be solved by dynamic programming tools. Evaluating those options and comparing them with the convex formulation is left as future work as well. One of the drawbacks of the current optimization formulation is that a unique solution is not guaranteed. This is evident in the case of the Washer for BLUED 2 as shown in Table 3.2. The state transitions 15.5 and -13.1 are perturbed to result in new state transitions: 0.1 and -0.1 (and hence are removed as they are below 5 W, as explained in the section). A more reasonable solution perhaps is to

perturb those to 15 and -15, and still preserve the total number of cycles (which is 4). Incorporating this extra constraint for more *reasonable* solutions is also part of our future improvements.

Moreover, the framework has also assumed power values to be piece-wise constants between events. Incorporation of frameworks such as those proposed by Barker et al. that model appliance behavior between events based on the dominant electric component of the operating load could be beneficial as well [59].

Model selection and estimation of parameters for latent variable models like Hidden Markov Models (HMM) are well studied problems in various applications [60, 38, 39]. Our problem of discovery of appliance models fits well in this paradigm of topology learning in HMMs. Typically, such methods work by assuming that the model has a small number of states, and continuously splitting states until the likelihood of the observed data is maximized [60]. New variants of the algorithm have proven to be computationally efficient as well [60, 61]. Modeling the problem of energy estimation in this manner would perhaps be excessive, both in terms of computation and complexity. Because of the fact that hidden states and observed states are relaying the same information, clustering allows for a more efficient and arguably accurate way to estimate the number of states in energy estimation. Estimating emission probabilities would be problematic and prone to errors, as we are not assuming that training data (ground truth) is available from the appliances. Our estimation of transition probabilities is based on a maximum likelihood estimator, as opposed to a Baum-Welch algorithm which is used for parameter estimation in HMMs. Regardless of how the states and transitions were discovered, steps like

perturbation would be required to ensure errors do not get accrued upon reconstruction of power traces. In addition, checking for feasible cycles and getting rid of ones that violate appliance behavior is a definitive way to enforce zero probabilities on certain transitions. So, these checks would still be necessary even if the models were discovered using alternative techniques. In Chapter 3, we talk about how the error correction algorithm is analogous to inference in HMMs, and provide our case for the algorithm we use (as opposed to Viterbi). We believe comparison of computational performance and model accuracy of existing model discovery methods like STACS [60], ML-SSS [38] against our method would be intriguing. We delegate it as future work.

Although the results from the limited set of houses and limited set of appliances cannot be taken as validation of the proposed method itself, they do reflect its utility. Collecting data with extensive ground truth and event labels is a very cumbersome undertaking, and as a result, the state of the art has been limited to testing on controlled laboratory testbeds [62, 19, 22]. In addition, because we are introducing the first formalized framework dedicated for energy estimation, comparison with other methods or even benchmarking against a standard is not meaningful. In this thesis, we have extended the standards for “demonstration of utility” of proposed methods by working with publicly available datasets, and by labeling them manually when labels were not available. To overcome the potential for errors due to manual labeling of events, a clear strategy is required in the long term. For instance, labeling by multiple experts with a posterior cross-validation step would be one possible solution.

3.9.3 Evaluation criteria

In our analysis, we use the percentage error in energy estimation as the metric for evaluating the framework ($\frac{\hat{E}-E}{E} \times 100\%$). Although separate steps of the framework could be evaluated individually, the ultimate reflection of the robustness of the method is in its ability to predict energy consumption. Measures of model complexity such as Bayesian Information Criteria (BIC) could be evaluated if multiple energy estimation frameworks are to be compared against each other.

3.10 Chapter Conclusion

In this chapter, we presented the case for the need of automated appliance modeling, and its importance in energy estimation for NILM. We surveyed relevant literature for propose ways to model appliances, and their suitability for energy estimation. We proposed a framework for energy estimation that incorporates the following steps (1) clustering of features observed at events using affinity propagation, (2) perturbation of extracted state transitions so that they follow ZLSC, (3) creation of FSMs based on observed transitions, (4) correction of any errors in labeling that might violate ZLSC, and (5) estimation of energy using the new states. We evaluated the framework for energy estimation on public datasets from 3 houses and 17 appliances, and presented the results as errors in energy estimation as compared to ground truth. Finally, we discussed ways in which the framework could be improved. In the next chapter, we explore the step of error correction in detail and study the framework further.

Table 3.3: Actual and estimated error for estimated energy for a select group of appliances in 3 houses from different datasets. The energy was estimated using the framework presented above.

House	Appliances (label)	Estimated Energy (kWh)	Actual Energy (kWh)	Error (%)
BLUED-1	Refrigerator (111)	6.6	6.7	1.3
	Lamp (101)	0.9	1.1	18.2
	TV (129)	6.2	6.4	2.2
	Computer (118)	2.1	2.0	-5.3
	Laptop (120)	0.9	0.8	-7.8
	AV System (112)	4.8	4.9	1.1
Average Error				5.9
BLUED-2	Refrigerator (111)	24.3	25.7	5.4
	Washer (183)	0.2	0.2	15.6
	TV (129)	0.5	0.6	15.9
	Computer (120)	6.8	5.3	-29.2
Average Error				16.5
REDD-1	Oven (3)	3.9	3.4	-16.7
	Refrigerator (5)	29.2	24.0	-21.4
	Dishwasher (6)	12.6	11.1	-12.8
	Lighting (9)	21.4	18.6	-15.2
	Washer (10)	2.5	1.7	-42.3
	Microwave (11)	6.4	9.7	33.9
	Dryer (20)	16.1	14.0	-14.6
Average Error				22.4

Table 3.4: Models for generating simulated data for analysis. 100, week-long, power traces were generated for each appliance

Appliances	Mean States (Watt)	Variance (Watt ²)	Transition probabilities
Refrigerator	{2 160 180}	{5 100 100}	$\begin{bmatrix} 0.99 & 0.005 & 0.005 \\ 0 & 0 & 1 \\ 0.005 & 0 & 0.995 \end{bmatrix}$
Microwave	{4 1700}	{100 1000}	$\begin{bmatrix} 0.9995 & 0.0005 \\ 0.4 & 0.6 \end{bmatrix}$
Washer	{0 5000}	{100 5000}	$\begin{bmatrix} 0.99999 & 0.00001 \\ 0.0099 & 0.9901 \end{bmatrix}$
Dishwasher	{0 1400}	{100 1300}	$\begin{bmatrix} 0.99999 & 0.00001 \\ 0.0006 & 0.9994 \end{bmatrix}$
AC	{4 2300}	{100 2300}	$\begin{bmatrix} 0.99 & 0.01 \\ 0.0001 & 0.9999 \end{bmatrix}$

Table 3.5: Average error in energy estimation for simulated data for a 100 houses.

Appliances	Average # of events per week	Average energy consumption per week (kWh)	Average error in energy estimation (%)
Refrigerator	501	20.3	3.8
Microwave	13	0.7	15.2
Washer	12	9.2	0.2
Dishwasher	12	4.3	0.4
AC	122	382.2	2.2

Chapter 4

An Error Correction Framework

As discussed in earlier chapters, the energy estimates can be affected by errors resulting from different algorithmic steps in NILM or in the energy estimation framework. A robust framework for energy estimation should use the labels from classification to (1) model the different state transitions that can occur in an appliance (2) account for any misclassifications by correcting event labels that violate the extracted model, and (3) accurately estimate the energy consumed by that appliance over a period of time. In this chapter, we extend the error-correction step in the framework presented in Chapter 3 by introducing an improved algorithm which looks at sequences generated by FSMs and corrects for errors in the sequence. Here, errors are defined as state transitions that violate the FSM.

Continuing with the communication model analogy in Figure 1.5, this step of error correction is analogous to creating a channel inverter for corrupt data that has passed through a noisy channel. The inverter is designed using certain assumptions

This chapter is based on the manuscript in [63]

about how the channel likely corrupts the data¹. Notationally, this is the step where we learn the function $h : (\Upsilon \times \Psi) \rightarrow \Upsilon'$ defined in Section 2.1.2. The input to the function h is a tuple (v, ψ) consisting of a time-series data v for an appliance, and its model ψ . v is a series of state transition assignments for all the events associated with the appliance, and ψ is the model learnt from the event time-series of the appliance (ξ) using function g . The output of this step is a time-series ($v' \in \Upsilon'$) for the appliance which follows the state transitions dictated by the transition probability matrix (A_ψ) as given by its model ψ . This process is illustrated in Figure 4.1.

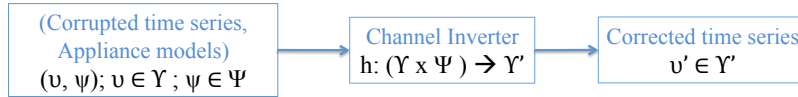


Figure 4.1: The input and output associated with the framework proposed in this Chapter.

4.1 Introduction to the framework

Our framework for error correction is based on the concept of minimizing the global cost of correction for errors in a sequence. Ultimately, this reduces to solving the shortest path problem in a directed acyclic graph. Following are the notations that will be used:

For rest of the chapter, the notations will refer to events occurring for a particular appliance y_i . As summarized in Figure 3.2, once all the features (X) are extracted from events occurring at timestamps T , they are clustered using affinity propagation algorithm. $C = \{c^1 \dots c^n\}$ denotes the set of labels for the n clusters that result (with $n \in \mathbb{N}$). Let $C_{seq} = \{c_1, c_2 \dots c_m\}$ represent the ordered sequence of cluster

¹As will be discussed later, the assumptions about the channel are that it imparts single bit insertion, deletion, or substitution errors on the data stream

assignments for all the events such that c_k denotes the cluster label associated with the state transition that occurs at event timestamp t_k . Let A denote the transition probability matrix learnt from the FSM, where $a_{ij} \in \mathbb{R}$ denotes the probability of transition from state c^i to state c^j . If $c_{k-1} = c^i; c_k = c^j; \forall c^i, c^j \in C$, an error occurs whenever $a_{ij} = 0$.

The first step is to model the cost functions. Costs can be of different types based on the kind of error correction being done. For error correction using substitution, where a faulty state is replaced by another state which has a valid transition, we define errors as the negative logarithm of transition probabilities, i.e.

$$\chi_{i,j}^S = \log(-\gamma(a_{ij})) \quad (4.1)$$

Here the superscript S denotes the cost for substitution between states $c_{k-1} = c^i; c_k = c^j; \forall c^i, c^j \in C$, and γ is a constant that alters the probability of substitution based corrections. This ensures that transitions with low probabilities will have a higher associated cost when replacing an erroneous term, and vice versa. Transitions that are not possible, will have infinite associated cost. Costs for insertion of a new state, and deletion are also calculated.

For insertion, we calculate the joint probability of the new state being inserted, again, c_k , coming after its preceeding state, c_{k-1} and before the succeeding state, c_{k+1} . As before, assume $c_{k-1} = c^h; c_k = c^i$, and $c_{k+1} = c^j; \forall c^h, c^i, c^j \in C$. The probability is $a_{hi} a_{ij}$. Depending on the prior knowledge we have about the process, i.e. belief on the kinds of errors that can occur, the cost function for inserting a

state c^i before erroneous transition from c^h to c^j is calculated as

$$\chi_{i,j}^I = \log(-\mu(a_{hi} a_{ij})) \quad (4.2)$$

where μ is a constant (typically a small number) that alters the probability making insertions less likely based on prior belief. The superscript I denotes that the cost is for insertion.

Similarly, for deletion, we calculate the cost using the transition probability from the state before the one being deleted to the state after it. Extra cost is added to make deletions more or less likely depending on prior knowledge. Using the same notation as before, if c^k is to be deleted, mathematically, this amounts to calculating,

$$\chi_{i,j}^D = \log(-\nu(a_{hj})) \quad (4.3)$$

The variable ν is again a constant similar to μ , and the superscript D denotes the cost for deletion, with c^i being the state that is deleted for correcting the error.

The goal is to correct errors, as defined above, by making changes that minimize the cost of correction. For instance, a local greedy algorithm, as proposed in Chapter 3 can result in error propagation. An ideal algorithm will be able to correct for local errors while ensuring that the effect of the correction does not result in propagation of error. In the example presented in Section 1.3, the error occurring at the second event in $C_{seq} = (1, 2, 2, 2, 1, 2)$, could be solved by checking the transition probability for the most likely state as done in Chapter 3. This would involve replacing the erroneous state (2) in the third event with the state with the highest probability (3),

resulting in $C_{seq} = (1, 2, 3, 2, 1, 2)$. This would, however, introduce another error, as the transition from 3 to 2 is not possible. So, the algorithm would iterate again resulting in $C_{seq} = (1, 2, 3, 1, 1, 2)$. After several iterations, the final sequence will converge to $C_{seq} = (1, 2, 3, 1, 2, 3)$, with energy consumption of 60 kWh. So, even though the resulting sequence does not have "errors", the error correction at a single point has resulted in a propagation of errors which has affected the energy estimate. An ideal error correction algorithm will be able to account for such propagation of errors resulting from correction steps in the sequence, and choose the optimal correction step (which, in this case, would be to replace 2 with a 1 in the third event in C_{seq} .)

To tackle this problem, we represent C_{seq} as a fully connected network diagram as shown in Figure 4.2. For an error point c_i in C_{seq} there are multiple error correcting steps possible, each with specific costs denoted by the matrix χ of dimension $m \times (2n + 1)$. Throughout this chapter, we use χ_{ij} to denote the cost element in row i and column j of χ . The goal will be to find a feasible path from c_1 to c_m with the least cost possible. We achieve this by modeling the problem as a special case of the min-cost flow problem in Network theory, known as the shortest path problem [64]. The basic idea behind the problem is to find the shortest path between two nodes in a directed graph, when the distance between all connected nodes (also known as weights or cost) is known. Several algorithms have been proposed to solve the problem, with Dijkstra's algorithm being perhaps the most well known and simplest in terms of implementation [65]. In the next section, we formalize this optimization problem.

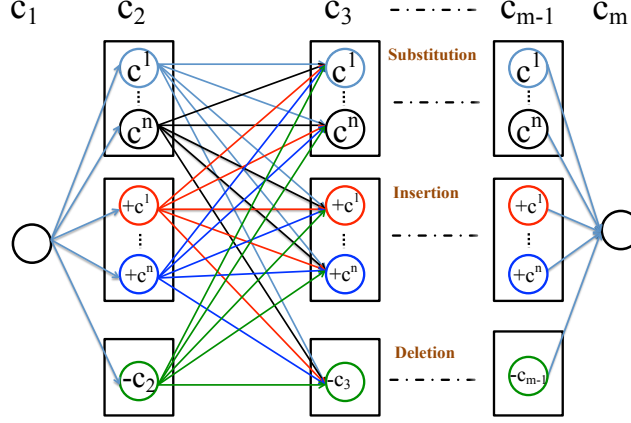


Figure 4.2: Network flow diagram for the sequence $C_{seq} = \{c_1 \dots c_m\}$. Three different kinds of error corrections are possible: Substitution, Insertion and Deletion.

4.2 Shortest-path formulation

Notationally, we have the following optimization framework for the shortest-path formulation of the error correcting algorithm. As depicted in Figure 4.2, C_{seq} is the erroneous sequence. The source node is c_1 and the destination node is c_m . The goal is to find the shortest path from the source to the destination. Say d_{uv} is a variable that denotes whether an edge lies in the shortest path or not. So, $d_{uv} = 1$ if the edge between node u , denoted by c_u and node v denoted by c_v lies in the shortest path. The way the network flow is arranged in this particular formulation, $v = u + 1$. The source node is denoted by c_1 and the destination node denoted by c_m . As mentioned before, $c_u, c_v \in C$. The shortest path algorithm optimizes:

$$\min \sum_{u,v} (d_{uv} \chi_{uv}) \quad (4.4)$$

subject to: $d > 0 \forall u, v$

$$\sum_u d_{uv} - \sum_v d_{uv} = \begin{cases} 1 & \text{if } u = 1 \\ -1 & \text{if } u = m \\ 0 & \text{otherwise} \end{cases}$$

The unimodularity of the constraint co-efficient matrix allows the relaxation of the inherent integer constraints in the setup, while still ensuring that all solutions are either 0 or 1. Proof and further details can be found in [66]. There are several algorithmic approaches to solving this optimization problem. We use, what is perhaps, the most popular option, known as the Dijkstra's algorithm. The computational complexity of Dijkstra's algorithm is $\mathcal{O}(V^2)$, where V is the total number of vertices in the graph. For sequence C_{seq} , $V = m(2n + 1)$, and hence the computational complexity in using Dijkstra's algorithm is $(m(2n + 1))^2$, where $n = |C|$ is the total number of possible states. The number of vertices can be pruned using some prior knowledge about state transitions, and by removing certain nodes in the sequence that are clearly not possible (for instance, edges with $\chi_{uv} = \infty$). Worth noting is that there are variations of Dijkstra's methods (d-way heap, Fibonacci heap, etc.), that can solve the shortest path problem faster, depending on the setup of the network and the kinds of data structures used [66, 67]. Since computational complexity of the algorithm is not a major limiting factor for the length of sequences typically encountered in NILM, we only explore the standard version of the algorithm.

The shortest path calculated using Dijkstra's algorithm incorporates information about state transitions, but does not yet utilize the constraints resulting from the possible states in the FSM. So, for instance, the net energy consumption at any given point for an appliance can still be more than the maximum allowed state for

that appliance. To incorporate this information, we employ an iterative algorithm that changes the cost function at specific points in the shortest path. The idea, as depicted in Algorithm 1, is to iteratively identify nodes that allow the aggregate net power consumption to be in violation of the maximum or minimum possible state, and locally set the cost for the immediate edge that leads to such node to infinity. Then the shortest path is recalculated, and the violation of states is checked for again. The algorithm terminates after a shortest path with no violation is found.

Algorithm 1 Iterative correction of violations in FSM states.

```

EndFlag = 0
while EndFlag  $\neq$  m do
  for i in 1 to m do
     $C_{seq}^{sum}[i] = \sum_1^i (\Delta Ps \text{ mapped by } C_{seq})$ 
    if  $C_{seq}^{sum}[i] > \max(\text{FSMstates})$  OR  $C_{seq}^{sum}[i] < 0$  then
       $\chi_{(i-1,i)} = Inf$ 
      EndFlag = i
      Break
    end if
  end for
   $C_{seq} = \text{CalcShortestPath}(\chi)$ 
end while

```

4.3 Evaluation

In this section, we evaluate the effect of using this error correcting formulation on C_{seq} -like sequences originating in supervised NILM. Given that the ultimate goal of the corrected sequence is to perform energy estimation (i.e. calculate E_i for appliance y_i), we evaluate the capacity of the framework to both recreate a sequence that is as close as possible to the original sequence, and to estimate energy as closely as possible.

4.3.1 Evaluation metrics

We denote the erroneous sequence as C_{seq}^E ², the corrected sequence as C_{seq}^C and the uncorrupted (original) sequence as C_{seq}^O . To evaluate how well the error correction framework works, we need some metric to compute the similarity between C_{seq}^C and C_{seq}^O . Traditional metrics to evaluate the proximity of time series sequences like Euclidean distance, or RMS error fail in this case because insertions and deletions may lead to C_{seq}^C and C_{seq}^O which have different lengths. To get around this limitation, we leverage the Dynamic Time Warping (DTW) distance measure, which is basically a metric to compute the similarity between two temporal sequences of different lengths. DTW has been used to compute the similarity of sequences that might be warped non-linearly in time in applications ranging from shape matching to speaker recognition [68, 69].

Say $X = (x_1, \dots, x_A)$ and $Y = (y_1, \dots, y_B)$ are two time-series sequences of length A and B respectively where $A, B \in \mathbb{N}$. Further, assume that the two sequences are from some vector space V such that $x, y \in V$. A cost metric is then defined as a function f that performs the following mapping: $f : V \times V \rightarrow \mathbb{R}^+$. Upon calculating the cost metric for all possible pairs of items in sequences X and Y , a cost matrix $F \in \mathbb{R}^{A \times B}$ results. The DTW algorithm attempts to find an alignment between sequences X and Y that minimizes the overall cost. The optimal cost of alignment becomes the DTW distance. For the obvious case where X and Y are the same sequence, the DTW distance will be zero. Figure 4.3 shows an example of two time series sequences with corresponding cost matrix, and Figure 4.4 shows the shortest path of alignment in the cost matrix.

²so far, we have denoted it as simply C_{seq}

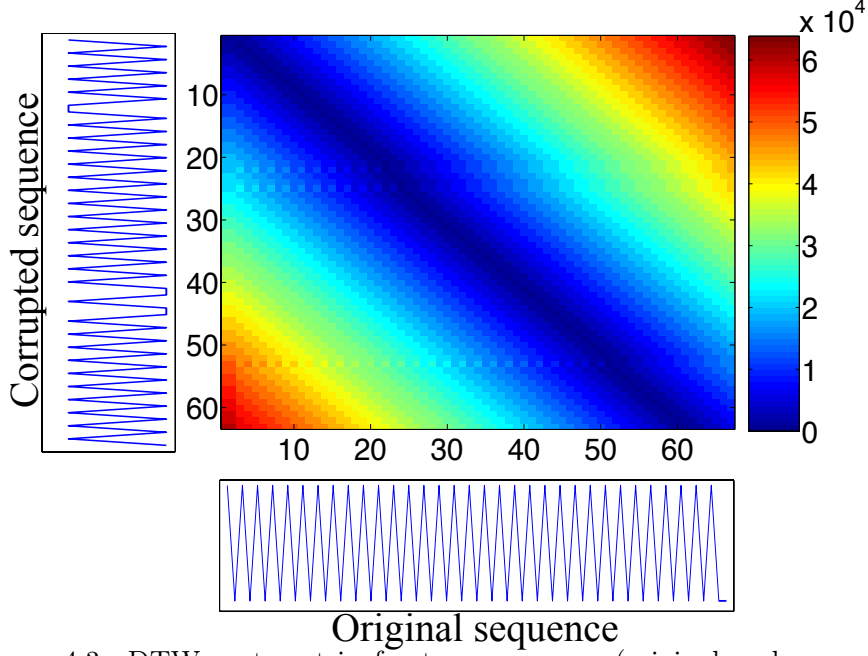


Figure 4.3: DTW cost matrix for two sequences (original and corrupted) for a washer (label: 3) in dataset REDD-1. The appliance only has two states, and hence the sequence fluctuates between two values.

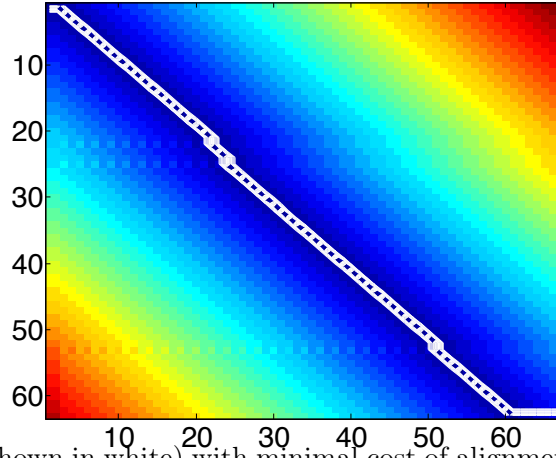


Figure 4.4: Path (shown in white) with minimal cost of alignment for the original sequence and corrupted sequence as shown in Figure 4.3. The DTW distance was 5808.

In addition, the error correction farmework imparts certain appliance specific constraints to C_{seq}^E to facilitate energy estimation. To evaluate the efficacy of those

constraints we compute the energy consumed by the appliance as given by C_{seq}^C (denoted by \hat{E}) and compare it with the actual energy consumed by the appliance (denoted by E). The evaluation metric here is simply a percentage error as given by

$$\mathbf{e} = \frac{E - \hat{E}}{E} \times 100\% \quad (4.5)$$

4.3.2 Evaluation on Simulated Data

We evaluated the framework on both real and simulated data based on the metrics discussed. For simulation, we created an appliance model which, in turn, generated the sequence of state transitions while preserving appliance like properties (e.g., ZLSC³). The idea behind the simulations was to study how different parameter values for cost constants affect the corrected sequence. In addition, for testing on real data, we evaluated our model on 43 different appliances collected from 19 different houses.

Appliance model

The following are the major components of the simulated appliance model; these are also summarized in Table 4.1.:

- Number of states (n_1): A random number of states of operation (between 2 and 9). $n_1 \sim \mathcal{U}\{2, 9\}$
- Power consumption (P_i): Random values between 50 and 3000 assigned to each state. $P_i \sim \mathcal{U}(50, 3000)$
- Transition probability (A_1): A doubly stochastic hollow matrix of dimension $n_1 \times n_1$ with random values between 0 and 1.

³ZLSC: The Zero Loop Sum Constraint, first coined by Hart, stipulates that the sum of all possible power transitions in a cycle should equal zero. A cycle occurs when an appliance in one state, undergoes some state transitions, and returns to the same state.

- Sequence length (m): A variable set to 500 to represent the number of events registered for the appliance. This is typically adequate to capture appliance behavior.
- State Sequence (C_{seq}^{st}): A sequence of dimension $m \times 1$ generated using **hmmgenerate** function in MATLAB given A_1 .
- Original State transition sequence (C_{seq}^O): $C_{seq}^O = Z \times C_{seq}^{st}$, where Z is an $m \times m$ matrix (sometimes also called a *difference matrix*) of the following form:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{pmatrix}$$

- Number of state transitions (n): Number of unique values in the state transitions sequence.
- Transition probability (A): A matrix of dimension $n \times n$, calculated using the the frequency of transition from each state to another in C_{seq}^O . This can also be computed using **hmmestimate** in MATLAB.
- Emission probability (B): A randomly generated doubly stochastic matrix of dimension $(n \times n)$ with diagonal elements set to random values close to 1. The rest of the matrix has random values less than 0.1. This ensures that most of observed sequence come out as the original sequence with potential for a few errors.

Table 4.1: Summary of the parameters used in the simulation model

Parameter	Value	Parameter	Value
# of states	$\mathbf{n}_1 \sim \mathcal{U}\{2, 9\}$	Power consumption	$\mathbf{P}_i \sim \mathcal{U}(50, 3000)$
Transition probability	$\mathbf{A}_1 \sim$ random doubly stochastic hollow matrix of size n_1	Sequence length	$\mathbf{m} = 500$
State Sequence	$\mathbf{C}_{seq}^{st} \sim \text{hmmgenerate}$ given A_1	Original state transition sequence	$\mathbf{C}_{seq}^O = Z \times \mathbf{C}_{seq}^{st}$
# of state transitions	$\mathbf{n} = \#$ of unique values in \mathbf{C}_{seq}^O	Transition probability	$\mathbf{A} \sim n \times n$ matrix computed using \mathbf{C}_{seq}^O
Emission probability	$\mathbf{B} \sim$ Randomly generated doubly stochastic matrix of size n	Observed state transition sequence	$\mathbf{C}_{seq}^E \sim$ Sequence sampled using \mathbf{B} and \mathbf{C}_{seq}^E
Duration	$\boldsymbol{\tau} \sim \{1, 7200\}$		

- Observed State Transition Sequences (\mathbf{C}_{seq}^E): A sequence with elements sampled from the probability distribution of corresponding elements in \mathbf{C}_{seq}^O given \mathbf{B} . When state 1 (out of n states) is observed in \mathbf{C}_{seq}^O but not in its corresponding position in \mathbf{C}_{seq}^E , it is an insertion error. Alternatively, if state 1 appears in \mathbf{C}_{seq}^O but not in its corresponding position in \mathbf{C}_{seq}^O , it is a deletion error. All state 1 values are subsequently removed from both sequences.
- Duration ($\boldsymbol{\tau}$): A sequence of length m with values between 1 and 7200 seconds, assigned to each state transition. $\boldsymbol{\tau} \sim \{1, 7200\}$.

Results

We ran the simulation on different combinations of Insertion, Deletion and Substitution cost constants (μ , ν , and γ). For each combination of cost constants (ranging from 1 to 50 each), we performed 100 simulations each, resulting in a total of 1.25×10^7 trials. The overall goal was to compare post error correcting framework sequence to a case without error correction, according to the metrics defined previously- namely DTW distance, RMS error, and final energy estimation results. Figures 4.5 and 4.6 show a color map of the effect of different μ and ν parameters on the energy estimation accuracy results for a specific instance of γ ($\gamma = 16$). The figure indicates that the energy estimation error improves consistently after the deletion cost crosses a certain threshold (roughly 15). This is most likely the point at which deletion corrections become costlier than substitution corrections, and hence more substitutions are made where possible. Given substitution errors are more likely to exist in the simulation, this improvement makes sense. Higher cost values, however, seem to be produce more unstable results as depicted by the standard deviation color map in Figure 4.6. Again, this makes intuitive sense, as once the cost values for a certain kind of correction cross a certain threshold, that type of correction is avoided, which results in a certain kind of error not being accounted for throughout the sequence.

Figure 4.7 shows how the three metrics vary before and after the framework is used for a specific instance of the three costs values ($\mu = 16$, $\nu = 16$, $\gamma = 12$). The lines in the middle of the box plots in Figure 4.7 indicate the median values of the logarithmic of the metrics of interest after 1000 trials each, with the box-margins denoting the limiting values for 75th and 25th percentile each. The

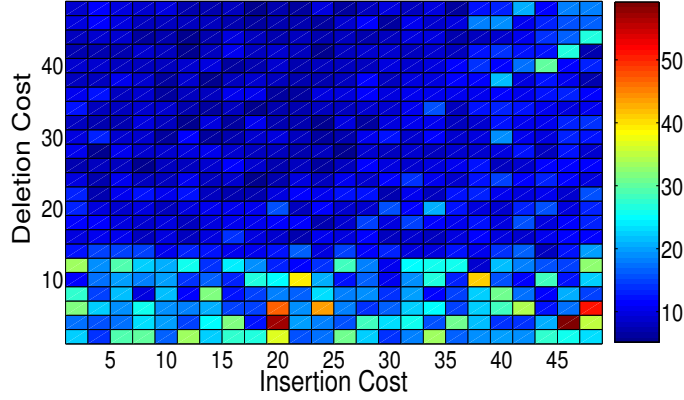


Figure 4.5: Median error in energy estimation (e) for varying values of μ and ν , with $\gamma = 16$. The medians were calculated after 100 simulations of each cost combination.

whiskers represent the minimum and maximum values in each of the simulation. Although, no statistically significant change is observed as the number of states are varied, it is obvious that using the error error correction improves results significantly as measured by all three metrics. The fact that the DTW metric is consistently less after the framework is applied means that the corrected sequence is closer to the original sequence than the corrupted sequence. In addition, we compared the power traces of the power signals created using the corrupted sequence and the the corrected sequence, and computed the RMS error. As expected, the corrected sequence had much lower RMS error values. All energy estimation errors were found to be less than 40% for corrected sequences, which highlights the utility of the method. Although these plots show the effect of the framework for one combination of the costs, the results from Figure 4.5 confirms that a wide range of parameter selection would still result in better energy estimates. The major takeaway from the simulation results is that error correction results in corrected sequences that are much closer to the original sequence than the corrupted sequence, and in much better energy estimates. Moreover, this performance is not limited to a significant value of the cost parameters, but is tolerant over a wide range.

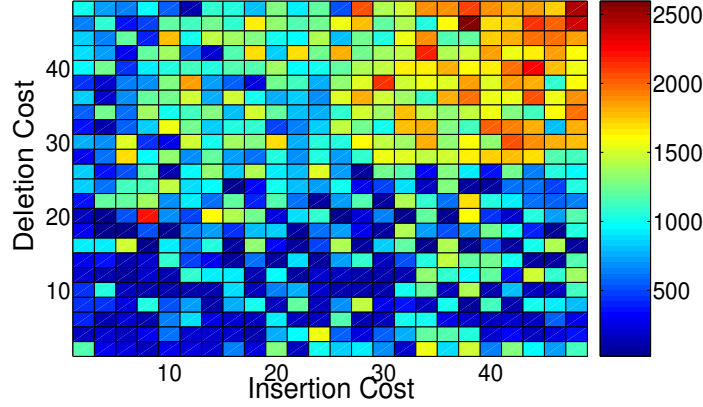


Figure 4.6: Standard Deviations for the median errors as reported in Figure 4.5 for 100 iterations of different cost combinations.

4.3.3 Evaluation on real data

We evaluated the error correction framework on 43 appliances from 2 different datasets comprising of 19 houses. To our knowledge, only one publicly available dataset (BLUED-1) has event-level ground truth data [2]. We have added two more datasets to the BLUED series, namely BLUED-2 and BLUED-3 which was collected by authors in [70]. All three datasets are collected in the same format (60 Hz real and reactive power). In addition, we took another publicly available dataset- REDD- and labelled all the events for appliances of interest in House 1 [37]. This dataset was collected at a frequency of 1 Hz real power. Since appliance-level ground truth was not available in this dataset, we had to label events at sub-circuit level. As a consequence, only appliances that had a dedicated sub-circuit were chosen for analysis. The appliance events observed at the aggregate power level were labelled manually using ground truth data collected at circuit-level. Inevitably, some errors get introduced in this process. Finally, we took 15 other houses from the REDD

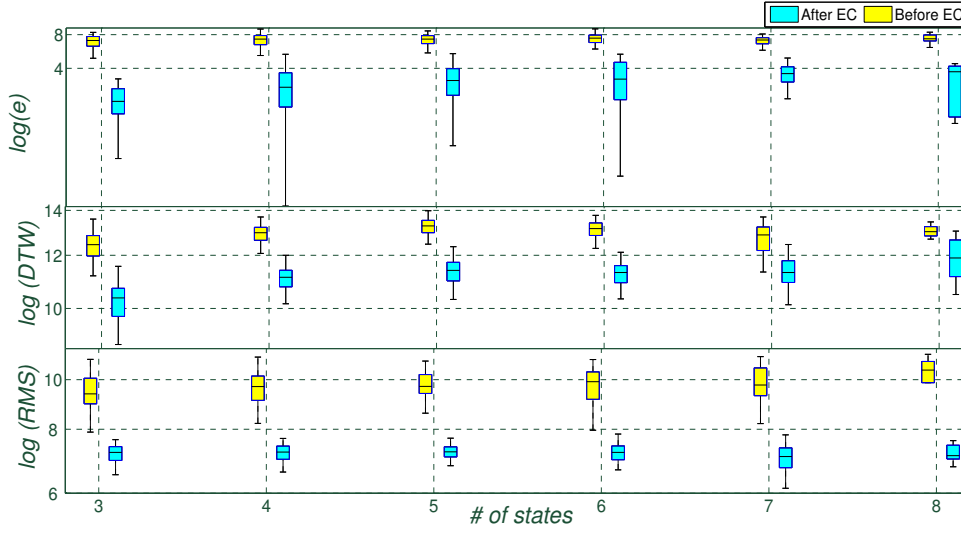


Figure 4.7: Box plots of logarithmic values of the energy estimation error: e (top), DTW distance of corrected sequences (middle), and RMS error of reconstructed power traces (bottom), with box margins representing the 25th and 75th percentile values, and median was calculated after 1000 simulation trials. EC stands for Error Correction.

series- a dataset that is not yet public- and labelled the events in the same way as House 1 in REDD. The data collection process in this dataset was the same as the one in the earlier REDD series. Two of the houses in the REDD series (23 and 24) had high frequency mains data (60Hz), and hence we utilized that instead of the low frequency data. Again, only appliances with dedicated sub-circuits were chosen which meant that only large appliances like the dishwasher, dryer, furnace, pool pumps, etc. were available for testing. Table 5.1 summarizes the datasets used in the analysis.

The framework described in [24] was used to generate the sequence of state transitions C_{seq} from the sequence of labelled events available for these appliances. This process also has the potential to introduce additional errors in the sequence. This could be because of miscalculation of state transition values due to simultaneous

events, or other factors like erroneous clustering, etc. Since no "ground truth" was available for C_{seq} , we only evaluated the framework for its ability to estimate energy, and compared it to the case when no error correction was done. The same methods described in [24] were used to compute energy values from resulting sequences. The error metric described in Equation 4.5 was used to compute error (e) after correction. In addition, we also computed the error in energy estimation without the error correction algorithm (denoted by e') using energy estimates done through the erroneous sequence C_{seq}^E . The energy estimate computed using C_{seq}^E is denoted by \hat{E}' .

$$\mathbf{e}' = \frac{E - \hat{E}'}{E} \times 100\% \quad (4.6)$$

To understand the effect of using the error correction framework, and its impact on the energy estimation errors, we created a metric which we call the improvement ratio (denoted by $\Delta\mathbf{e}$), defined as follows:

$$\Delta\mathbf{e} = \left| \frac{\mathbf{e} - \mathbf{e}'}{\mathbf{e}'} \right| \quad (4.7)$$

The improvement ratio denotes the fraction of the erroneous energy estimates that is corrected for by the use of the error correction algorithm. Values close to 1 are good because they indicate the error correction algorithm accounted for most of the errors, and values above 1 denote cases where the error correction algorithm worsened the energy estimates. Values of zero denote cases where the error correction algorithm does not affect the energy estimates. It is also to be noted that e' will rarely equal zero, as it is extremely unlikely that the energy estimates are precisely the same as the ground truth. Figure 4.8 summarizes the inputs and output of this evaluation process.

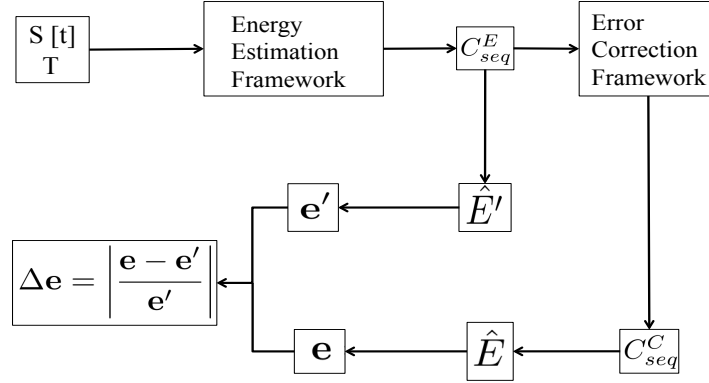


Figure 4.8: The inputs and outputs involved in the evaluation process when the error correction is applied on BLUED and REDD datasets. Energy estimation framework refers to the framework from Chapter 3, and Error correction framework is the one proposed in this chapter.

Results

The error correction framework was evaluated on the aforementioned datasets. Table 4.3 depicts the actual values after the framework was applied on 43 appliances. Again, the framework for energy estimation was the same as the one used by authors in [24]. As summarized in Figure 4.8, once the energy estimation framework generated C_{seq} , the values of energy were computed using C_{seq} only, and compared to the case when C_{seq} was corrected using the error correction framework. The error in energy estimation (e) was found to fluctuate between ± 70 for all appliances once error correction was applied. Without error correction, as showcased by Table 4.3, the error values e' can fluctuate drastically. On average, using the framework improved the energy estimates for 11 houses, did not have any impact on 6 houses, and impacted 2 houses negatively. The average negative impact $e - e'$ was -8.2%, which is a trivial trade-off compared to the positive improvements in energy estimation. Figure 4.9 illustrates these results in a log-log plot of the energy estimation error

Table 4.2: Summary of the datasets used for evaluation in the chapter. Dataset index B stands for BLUED and R stands for REDD. Appliances are indexed as follows: A-Refrigerator; B-Lights; C-TV; D-Computer; E-Laptop; F-AV System; G-Washer; H-Oven; I-Dishwasher; J-Microwave; K-Dryer; L-Furnace; M-PoolPump; N-HairDryer.

Data-set	Sampling Freq. (Hz)	Duration (days)	Appliances used	# of events
B-1	60	7	[A; B; C; D; E; F]	[616; 26; 54; 45; 14; 8]
B-2	60	7	[A; E; C; G]	[287; 53; 289; 1897]
B-3	60	7	[A; E; C; G]	[287; 53; 289; 1897]
R-1	1	26	[A; E; C; B; I; N]	[600; 47; 29; 46; 52; 22]
R-13	0.25	7	[I]	[109]
R-17	0.25	7	[I]	[69]
R-19	0.25	7	[K; J]	[435; 35]
R-20	0.25	7	[M]	[17]
R-21	0.25	7	[L]	[22]
R-23	60	7	[I]	[54]
R-24	60	7	[J; K]	[12; 227]
R-26	0.25	7	[I; F; K]	[242; 39; 251]
R-34	0.25	7	[H]	[80]
R-37	0.25	7	[A]	[306]
R-40	0.25	7	[J]	[92]
R-46	0.25	7	[H]	[72]
R-52	0.25	7	[L]	[292]
R-53	0.25	7	[M]	[25]
R-57	0.25	7	[J; K]	[36; 140]

without error correction (e') and the change in energy estimation because of the error correction framework ($|e - e'|$). The line ($y = x$) denotes the ideal case, where

all the errors in energy estimation process is accounted for by the error correction step. The shaded area: $x > y$ denotes regions where use of the error correction framework positively impacts the energy estimation process, by accounting for a portion of the errors. The zero values denote cases where there was no change in energy estimation error upon using the error correction framework. Points in the region: $x < y$ show cases where using error correction adversely affects the energy estimation results. Again, it can be seen that, on average, the framework has a positive effect on energy estimation for appliances.

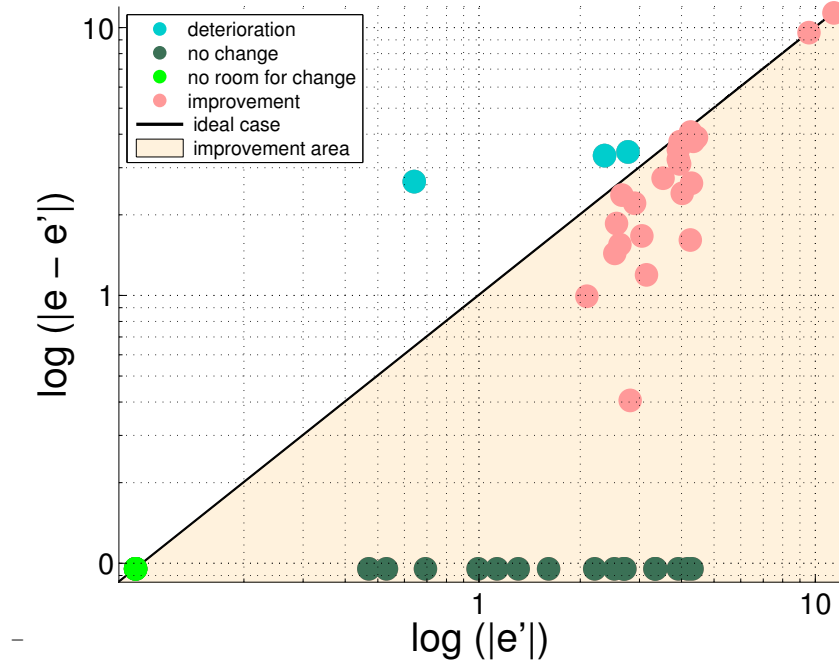


Figure 4.9: A log-log plot of the change in energy estimation error upon using the framework (y-axis) and the energy estimation error without the framework (x-axis) for 43 appliances.

In addition to error correction, we were also interested in understanding the effect of the appliance model (topology) learnt by our framework on the final energy estimation results. To study this, we first studied how cluster quality affected en-

ergy estimation results. For each appliance, we calculated a measure of intracluster dispersion weighted by intercluster distance, known as the Dunn Index. The higher the value of Dunn Index, the better the cluster quality. Figure 4.10 shows a plot of how error values changed as the cluster quality changed. Although one would expect the error to decrease as the cluster quality increases, no discernible impact of dispersion on energy estimation was observed. This could be due to limitations imparted by the sample size. One hypothesis is that erroneous events in the sequence affect the final error values more than the quality of clusters. Next, we wanted to understand the effect of the topology of the appliance on energy estimation errors. To do this, for each appliance, we used a variant of the affinity propagation algorithm (apclusterk) that allows the number of clusters to be specified using the same similarity measure. So, for each appliance, we varied the number of clusters and used the framework on the dataset. Figure 4.11 shows some of the error values for a sample set of appliances. The red circle denotes the number of clusters the algorithm converges to if it is not forced to assume a certain number of clusters. This analysis points towards there being certain tolerance to the number of clusters (and hence appliance model), but the error increases as you drift away from the *true* topology.

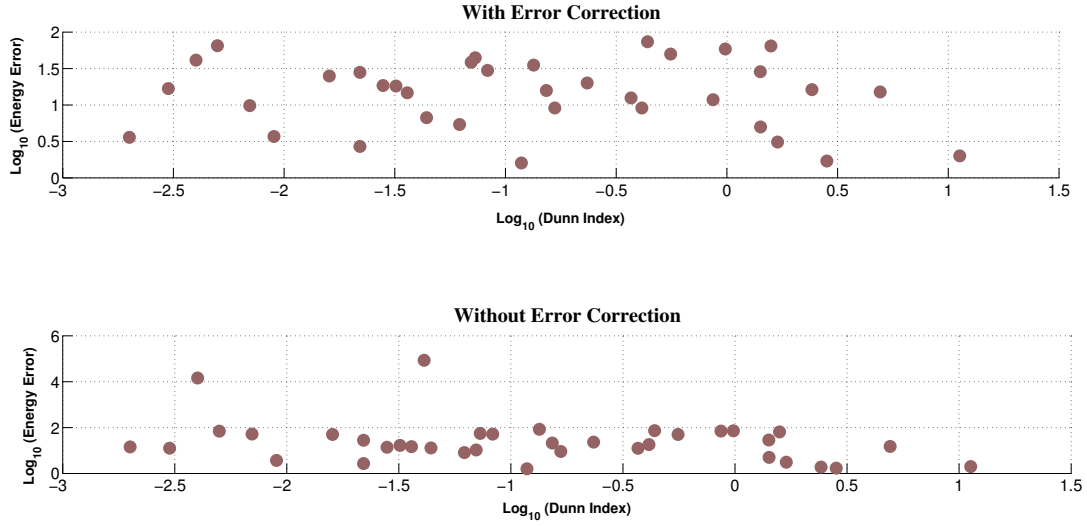


Figure 4.10: A log-log plot of the energy estimation error versus cluster quality as measured by Dunn Index. Each dot represents appliances in the same order as they appear in Table 4.3.

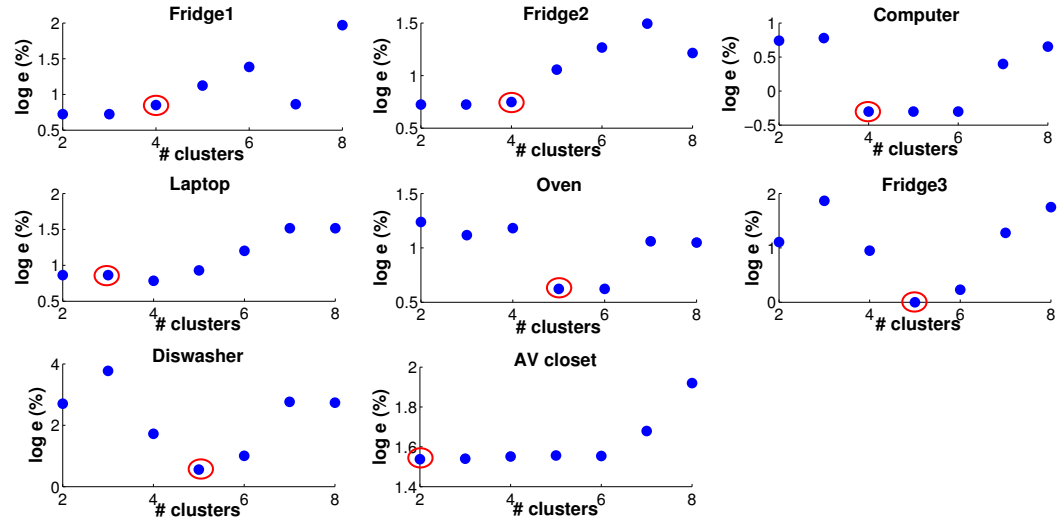


Figure 4.11: Change in energy estimation error (logarithmic) as the number of clusters were varied for a sample set of appliances from the dataset. The red circle denotes the number of clusters and error value the algorithm converges to if it is not forced to assume a certain number of clusters.

Table 4.3: Results of energy estimation on 43 appliances from 19 houses with and without the error correction framework. The sequence for error correction, and subsequent energy estimation was done using the framework discussed in Chapter 3. Dataset index B stands for BLUED and R stands for REDD.

Data-set	S.N.	Appliances (label)	E (kWh)	e (%)	e' (%)	Δe
B-1	1	Refrigerator (111)	6.5	-18.5	-13.8	0.3
	2	Lamp (101)	1.1	9.1	18.2	0.5
	3	TV (129)	6.4	-3.1	-3.1	0.0
	4	Computer (118)	2.0	5.0	5.0	0.0
	5	Laptop (120)	0.8	12.5	12.5	0.0
	6	AV System (112)	4.9	-2.0	-2.0	0.0
Average % Improvement						0.1
B-2	7	Refrigerator (111)	25.7	-38.6	10.6	2.6
	8	Washer (183)	0.1	-25.0	-50.0	0.5
	9	TV (129)	0.6	9.8	-52.5	0.4
	10	Computer (120)	5.3	29.8	-52.0	0.8
Average % Improvement						1.1
B-3	11	Refrigerator (111)	7.4	-5.4	-8.1	0.3
	12	Laptop (120)	0.9	-44.4	-55.6	0.2
	13	TV (129)	1.7	11.8	70.6	0.8
	14	Lamp (146)	3.0	20.0	-23.3	0.1
	15	Dishwasher (167)	1.9	-73.7	-73.7	0.0
	16	Hair Dryer (181)	0.1	0.0	85300.0	1.0
Average % Improvement						0.4
R-1	17	Oven (3)	3.4	14.7	14.7	0.0
	18	Refrigerator (5)	18.7	47.1	-16.0	1.9
	19	Dishwasher (6)	11.1	-2.7	-2.7	0.0
	20	Lighting (9)	18.6	33.3	76.3	0.6
	21	Washer (10)	1.7	41.2	14452.9	1.0
	22	Microwave (11)	9.7	18.6	-34.0	0.5
	23	Dryer (20)	14.0	3.6	14.3	0.8
Average % Improvement						0.7
R-13	30	Dishwasher (9)	18.6	-15.1	-15.1	0.0

Data-set	S.N	Appliances (label)	E (kWh)	\mathbf{e} (%)	\mathbf{e}' (%)	$\Delta \mathbf{e}$
R-17	31	Dishwasher (11)	6.6	-18.2	-16.7	0.1
R-19	32	Dryer (12)	34.5	-6.7	-13.0	0.5
	33	Microwave (15)	0.1	-64.5	-64.5	0.0
Average % Improvement						0.3
R-20	34	PoolPump (0)	42.0	16.2	-1.9	7.5
R-21	35	Furnace (0)	2.9	58.6	72.4	0.2
R-23	24	Dishwasher (0)	1.9	-15.8	-21.1	0.3
R-24	25	Microwave (1)	1.2	0.0	0.0	0.0
	26	Dryer (12)	14.3	-16.8	-12.6	0.3
Average % Improvement						0.2
R-26	27	Dishwasher (1)	1.1	-9.1	-9.1	0.0
	28	AV Closet (17)	8.8	-35.2	-84.1	0.6
	29	Dryer (18)	8.2	-28.0	-28.0	0.0
Average % Improvement						0.2
R-34	36	Oven(17)	2.7	3.7	3.7	0.0
R-37	37	Refrigerator(3)	0.6	-16.7	-50.0	0.7
R-40	38	Microwave (4)	2.0	-50.0	-50.0	0.0
R-46	39	Oven (4)	2.7	0.0	0.0	0.0
R-52	40	Furnace (2)	2.0	-65.0	-70.0	0.1
R-53	41	Pool Pump (0)	31.9	1.6	1.6	0.0
R-57	42	Microwave (1)	0.7	-28.6	-28.6	0.0
	43	Dryer (12)	17.6	1.7	1.7	0.0
Average % Improvement						0.0

4.4 Discussion

The analysis on simulated and real data show that there is significant improvement in the process of energy estimation upon using an error correction framework. Figure 4.7, for instance, shows that the DTW distance of corrected sequence is consistently lower than that of corrupted sequence over varying number of states. This was found to be true for other parameters in the simulation, including sequence length, and probability threshold. Similarly, and perhaps, as expected, the same was found of energy estimation results and RMS distance. As can be seen in Figure 4.9, there are certain instances where using the framework negatively affects the energy estimation. Typically, these are instances where the event labels (and, consequently, sequence C_{seq}^E) were almost perfect, which is not reflective of a real world scenario. In addition, in some of these cases, the errors that are present are valid under the FSM model, and are hence not detected as errors by the algorithm. Overall, however, the positive gain in energy estimation instances (mean: 5266.8%, median 15.5 %), far outweighs the negative impacts (mean: 13.9 %, median: 9.5 %). For 17 appliances, using the framework does not change the energy estimation results. This is mostly because the labels are accurate enough to begin with and error correction is not required.

The major parameters that affect the energy estimation results are the cost constants for substitution, insertion and deletion. It was found that lower values of insertion and deletion costs mean that the variance is low in energy estimation errors as shown in Figure 4.6. The median error was low for most combinations of these two costs. The only information the error correction framework has about the noise in the channel is about state transitions that are not possible. Due to this

inherent difficulty of the problem, sometimes, there are erroneous sequences that satisfy the FSM under which the sequence is operating, and hence do not show up as errors. As a result, there is potential for errors to be accumulated and the energy estimates to be significantly off from ground truth. Hence, we used median of 100 trials as the representative of each cost combination, as means are swayed heavily by outliers. Figuring out sequences (or sequence segments) that are highly unlikely but still probable under operating FSM model of the appliance, and treating those are errors is an interesting extension to the problem. One way to achieve this is to create a likelihood metric for an observed sequence, and assume it to be erroneous if it lies under a certain threshold. Then, based on certain beliefs, assign specific points in the sequence as error points, and mold the problem into the framework described above. We leave such implementations as future work.

Another underlying but understated assumption that has been prevalent throughout the framework is that the sequences are long enough to reflect all possible appliance states and transitions enough times for a model to be created. Typically, this means at least a few days worth of data has to be collected from the appliance for energy to be estimated. Given the main goal of NILM is to give disaggregated feedback at the end of the a certain set time-frames (typically ranging from week to month) , assuming the availability of such data is not impractical. But there can be scenarios where a real time model generation with the available data and subsequent energy estimation might be required. Our models have not been tested for such implementations, and we leave such extension as future work as well. The model has also implicitly assumed that the initial and the final nodes of the sequence (c_1 and c_m) are fixed not erroneous. Obviously, this is not guaranteed. A more robust way

to find the starting and ending nodes for correction might be to utilize the the aggregate power information, and track it for baselines (also known as background power). Once a reasonable baseline has been established, everytime the aggregate power consumption reaches the baseline, it is guaranteed that the last event from the appliance is an *off* from its previous state, and the next event will be from *off* to a state of positive power consumption. Same information about baselines can also be utilized to constrain the energy estimates in the shortest path algorithm. Since finding appropriate baselines automatically in aggregate power consumption data is a complex problem in its own right, and an active field of research, we have steered away from utilizing that such information in our models at this point.

The way in which we have incorporated the FSM model into the error correction process was by iteratively and locally modifying the costs at each instance where the FSM states were violated. Although this method worked fairly well, a more efficient method is to modify the shortest path search in such a way that this check is performed at each node, and hence nodes that violate the maximum and minimum states level are not selected in the shortest path in the first iteration itself. The trade-offs of complexity of such an approach versus potential gains is an interesting study in itself, and we hope to explore it in the future.

As mentioned in Chapter 3, the problem of energy estimation can be modelled as an HMM, with the error correction being analogous to the inference step in HMMs. Algorithms like the Viterbi algorithm [40] could be then used to infer the true hidden sequence that follow the transitions dictated by the transition probability matrix. Both Viterbi inference and our algorithm work on transition probabilities between

consecutive states as the chief metric being optimized. We believe our formulation, in addition to removing the reliance on emission probabilities, allows for a more intuitive way to incorporate insertion and deletion errors. In addition, our formulation makes it easier to enforce appliance behavior specific constraints (e.g., net sum of power changes at a certain point cannot be more than the highest state in FSM), which would be very complex to integrate in a Viterbi formulation. We leave comparison of the performance of these two methods on sequences for energy estimation as future work.

One of the issues that has hindered comparison of different proposed NILM solutions and frameworks is the lack of standard datasets. In this thesis, we have used publicly available datasets where possible. Additionally, we have extended the repository of publicly available datasets for supervised NILM by labeling events manually when labels were not available. We will be making all the datasets, labels, and code utilized in this thesis publicly available as well. Energy estimation is an integral part of supervised NILM, and clearly a robust error correction framework can improve the results of the whole disaggregation process. Although our focus has been mostly on energy estimation, this method of using domain specific beliefs to correct for errors in sequences generated by processes whose models are known could be extended to other fields like DNA sequencing, etc.

4.5 Chapter Conclusion

In this chapter, we presented a framework for correcting errors in sequences generated for energy estimation in supervised NILM. Although we focus our analysis on the sequence generated by one particular framework, the method is actually agnostic

to how the sequences are generated. The method utilizes prior beliefs about how appliances behave in general and incorporates that to a network flow to minimize the cost of correcting errors in the sequence. We tested our framework on simulated data to study how different cost parameters affect the performance and found that over a wide range of values, using the framework yields better results. We further tested the framework on real data from 43 different appliances collected from 19 houses, and found that energy estimation results improve on average upon using the error correction framework. In addition to energy estimation in supervised NILM, this error correction framework can be utilized in energy auditing using external sensors, validating appliance labels generated by other processes, etc.

In the next chapter, we extend the use-case of our proposed frameworks beyond NILM energy estimation by using the combination of the energy estimation framework presented in Chapter 3 and the error correction framework from this chapter to perform virtual metering of electrical appliances.

Chapter 5

Virtual metering of electrical appliances

The frameworks presented so far for energy estimation follow two basic principles: (1) Using observed data to estimate the models of the underlying process that generates the data, (2) Using the estimated model to correct for any errors in the observed data. It incorporates expert (domain) knowledge about the process (appliance behavior, in this case) to constrain the problem. This is a generalizable approach to solving similar problems, and is not confined to energy estimation in supervised NILM. As an extension, in this chapter, we utilize the framework to virtually meter appliances using external sensors. The process is different in that while, before, it was the NILM steps of event detection and classification that were generating the event labels, now it will be the external sensors that will provide them.

This chapter is based on the manuscript in [71]

5.1 Literature Survey

In this section we review the state of the art in metering appliances using several direct and indirect techniques. Direct techniques refer to the use of active plug-meters to monitor the energy consumed by an appliance. Indirect techniques include the use of contactless sensors (e.g., sound, light, vibration, magnetic field, etc.) in order to infer the energy consumed by an appliance. NILM techniques for load-level disaggregation also fall under indirect techniques.

5.1.1 Direct metering techniques

At the time of writing this thesis, energy meters like The Energy Detective (TED) [72] and Episensor [73] are commercially available for measuring whole home electricity data with second to minute-level granularity. With electric utilities adopting smart meters aggressively, several software platforms are also currently available to access granular aggregate data at 15-60 minutes granularity. For sub-circuit level monitoring, devices like eMonitor [74], eGauge [75], FIDO [76], etc. are available in the market. Energy Hub [77], Watts-Up Pro [78], Kill-a-Watt [79], etc. are some commercially available plug meters that facilitate appliance level metering. These meters monitor power typically at a rate of 1 Hz or less. Several research projects have come up with their own version of plug-meters: ACme sensor mote [80], Fire-Fly Plug-meter [81], Synergy Energy meter [82], MIT Plug [83], REAM [84], etc., to name a few. As discussed before, the problem with the plug-meters available for energy audits currently is that they are cost-prohibitive, and typically cannot measure appliances that do not have a traditional plug, for instance, light fixtures, HVAC, dryers, water-heaters, dishwashers, etc. Use of other sensors along with the aggregate power consumption data as measured by energy meters provides a

convenient alternative to plug-meters for the purpose of electricity metering.

5.1.2 Indirect metering techniques

Clark et al. have proposed a framework called Deltaflow for submetering appliances using inexpensive and easily deployable sensors which emit radio pulses that correlate with an appliance's power draw [85]. The model assumes a quadratic relationship between the generated number of pulses and an appliance's power consumption, and auto-calibrates the sensors using a regression model. It is a promising approach in terms of ease of implementation. The regression model is, however, not good at handling simultaneous events resulting from multiple appliances, which could lead to major errors in energy estimation. It also has not been evaluated on complex and multi-state appliances, and scenarios where multiple such appliances are operating simultaneously, which would test some of the more simplistic assumptions in the model. Finally, the fact that the framework only works with specially designed pulse sensors limits its generalizability, at least in the short term. It is worth noting that the framework we utilize in this chapter is not a direct alternative to the Deltaflow framework. In fact, once a relationship between the change in number of pulse readings and state change of an appliance is established (this process calibrates the sensor to detect events), our method can also be utilized in order to meter the appliance.

In [86], authors present an algorithm for metering appliances using aggregate electricity measurements and perfect knowledge about ON-OFF states, the kind most contactless sensors would be able to provide. Their method relies on decomposing the aggregate power by solving a linear optimization problem involving knowledge about the states of appliances. In addition to the possibility of the opti-

mization problem failing to converge (especially as the number of loads increases). Their setup also requires perfect knowledge about all states of all appliances at all times. Although in their experiment, the authors acquire this using clean data from plug-meters, deployment of an actual sensor would inevitably result in several false signals. This would seem to severely undermine their performance of the optimization framework to converge to the right solution. The method also appears incapable of processing appliances with states beyond simple ON-OFF.

Authors in [35] present a method based on a regression model to self-calibrate easily deployable sensors (sound, light, magnetic-field) for the purpose of energy metering. Each kind of sensor is assigned a different calibration equation and the power is estimated by solving for parameters. Their architecture allows for several sensors to combine in order to estimate power consumption of appliances with multiple states. The energy estimation part, as noted in [86] as well, is treated as a black box, and the focus of the paper is on calibration of sensors. The ability of the framework to handle complex appliances and to allow for unmonitored appliances make it appealing. However sensitivity to ambient noise and sensor placement are major issues if electricity is to be metered using direct sensor readings. An ideal solution would have the ability to detect the occurrence of unlikely events based on some prior knowledge about the appliance's behavior, and utilize that to counter ambient noise. It would also be independent of sensor placement parameters like orientation and distance, as that limits the practicality of the solution.

Authors in [36] have also used magnetic field sensors to indirectly meter appliances. But again, the focus is on the sensor and not the energy estimation process.

Other authors, e.g., [20, 21, 87] have proposed the use of similar sensors to facilitate virtual energy metering through the use NILM techniques. Authors in [88] have devised a cheap plug-level sensor that only provides on-off labels for a similar purpose. Such solutions typically involve learning of appliance operation patterns using labels provided by sensors, and using that to estimate energy. As we reviewed in Chapter 1, NILM is still an active area of research, and has not yet reached a stage where it can be used in for real world applications [19, 22]. Earlier in this thesis, we explored part of the NILM problem in providing a framework for energy estimation, but other parts like event detection, classification are still active areas of research. The currently available literature in the field leads us to conclude that the available or proposed solutions for appliance metering have one or more of the following drawbacks: (1) cost-prohibitive (2) specific to a sensor-network architecture which limits scalability (3) unable to handle complex devices with multiple states (4) do not address the issue of energy estimation from annotated labels. In this chapter, we address this knowledge gap by utilizing a framework for energy metering that is sensor-agnostic and only requires basic state change labels of appliances to estimate energy.

5.2 Framework for analyzing sensor data

We utilize the framework proposed in Chapter 3, and supplement it with the error correction algorithm Chapter 4 to perform metering from sensor data¹. The different steps involved are summarized in Figure 3.2 [24]. Basically, the idea is to collect event labels corresponding to state changes in appliances using a contactless

¹a combination that will be referred to as the virtual metering framework throughout the chapter

sensor. Labels refer to the identity of the sensor/appliance that logged an event at a certain timestamp, and are represented as tuples of the form (y_i, t_j) where y_i is the appliance ID for the appliance that caused the event, and t_j is the timestamp at which the event occurred. As an example, if a fan goes from *off* to *low*, *low* to *high* and *high* to *off* at timestamps t_1, t_2 , and t_3 , respectively, and a vibration sensor is placed next to it, the sensor, based on changes in vibration that it receives, logs these three times as event times, and conveys that information to some computational platform. Meanwhile, power consumption at the aggregate level is also being monitored by an energy-meter that relays the information to the same platform. The event labels and the aggregate power consumption, are inputs to the framework. Figure 5.1 outlines the general idea of the setup for appliances that need to be monitored.

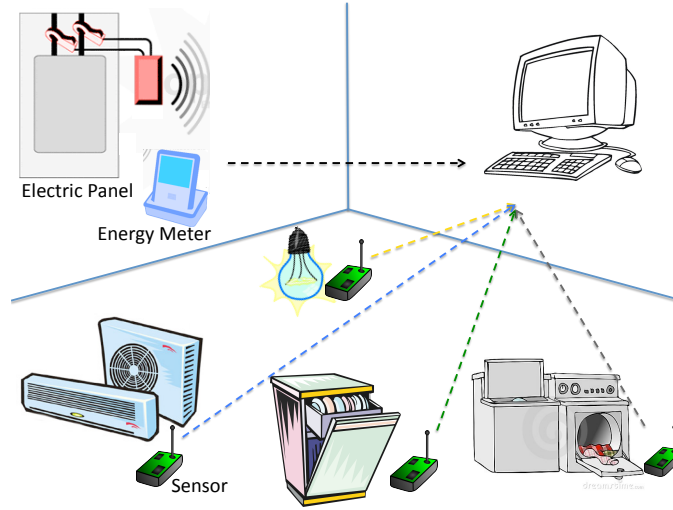


Figure 5.1: The vision for a contactless appliance-level energy metering setup. The sensors log state changes in appliances, and send that info to a central computational platform. The platform also receives aggregate electricity data sampled using some energy meter. Combining the two, it calculates energy for each appliance.

The framework reads an aggregate power trace, as measured by the energy

meter, denoted as $P[t]$, and event timestamps for a particular appliance (y_j), denoted by the set $T = \{t_1, \dots, t_m\}$. Based on the timestamps, it extracts features $X = \{\vec{x}_1, \dots, \vec{x}_m\}$ from the aggregate data; these features are usually step changes in real power as observed in the mains during events. Once enough events are logged, and respective features extracted, the features are clustered using a clustering algorithm in order to find the representative power values observed during state transitions for the appliance. This gives a sequence (C_{seq}), which is then passed as input to the error correction framework. In this chapter, we take the appliance model and error correction algorithm from these two frameworks, and utilize it for energy metering through sensors. In the next section, we share our results from a case study on 6 appliances.

5.3 Experiment and Results

To test the efficacy of the framework in accurately metering electricity through contactless sensors, we monitor 6 appliances inside a commercial building. The appliances are operated periodically during the week to ensure enough events were accrued. A HOB0 UX90-005 Motor On/Off Logger is used to sense the magnetic fields around these appliances [89]. The sensor logs binary values every time it senses a significant change in an appliance's power consumption. Values of 1 correspond to a state with the appliance's motor on, while 0 corresponds to a case where it is off. The sensors are placed close to the appliances so that they can sense the electromagnetic field arising as a result of current flow in the appliance. This is the same operating principle as the one used by our custom-designed sensor in the case study in Chapter 2. But compared to this commercial product, our EMF sensor



Figure 5.2: HOBOT motor on/off loggers (sensors) placed next to 6 appliances for virtual metering. The red rectangle shows the location in the appliance where the sensors were placed.

is more prone to noise and false events, and has less memory, and as a result, it needs a wireless communication network to relay state change information in real time. However, with further improvements and iterations of the device, it has the potential to be a cheap and effective option for virtual metering. Figure 5.2 shows the appliances along with the sensors used for virtual metering in the experiment.

Table 5.1 summarizes the kinds of appliances and number of events observed for each in the experiment. A FireFly Plugmeter [81] is used to collect ground truth data for these appliances; in cases where plugs were not available, the sub-circuit panel was monitored using an NI-9215 Data Acquisition card, and TED Current Transducers (CT) [72]. The aggregate data are monitored through the electrical panel using a TED-1000 device [72]. The Measurement and Transmittance Unit (MTU) in the device receives signals from the CTs attached to each panel, and communicates power readings to the Display Unit which is then connected to a computer. At the end of the collection period, the sensors are connected to a

computer as well, and all the logged values are downloaded. A Windows PC with 4 GB RAM, and 2.6 GHz Intel processor was used for computation. The events associated with each appliance are then extracted using a simple threshold-based event detector. This results in the two required inputs for the model- event labels, and aggregate power. To evaluate the framework, we compute the energy consumed by the appliances as given by their sequence C_{seq}^C (denoted by \hat{E}) and compare it with the actual energy consumed by the appliance as collected by the ground truth plug-meters (denoted by E). The evaluation metric here is simply a percentage error as given by $\mathbf{e} = \frac{|E - \hat{E}|}{E} \times 100\%$.

Table 5.1: A summary of the appliances used in the experiment and the events associated with each appliance along with the results from energy estimation using the virtual metering framework.

Appliance	# of events	\hat{E} (kWh)	E (kWh)	\mathbf{e} (%)
Refrigerator	63	2.23	2.24	0.44
Washer	408	1.03	0.91	13.2
Dryer	18	2.00	1.94	3.09
Air Conditioner	45	2.06	2.59	20.4
Vacuum Cleaner	54	1.33	1.06	25.4
Fan	32	1.24	1.21	2.50
Average Error				12.5

5.4 Discussion

The results of using the virtual metering framework on the data collected by the sensors are summarized in Table 5.1. The framework is able to meter power consumption for the refrigerator and the fan fairly precisely ($\mathbf{e} < 3\%$). Appliances like

the washer and dryer that have more complex states of operation exhibit slightly higher error ranges. Figure 5.3 displays a portion of the ground truth signal for these two appliances. The energy estimation framework (Figure 3.2) estimates the states for the washer as $\{0, 530, 800\}$ Watts. This approximation is a result of steps (A , D and E) as displayed in Figure 3.2. As the ground truth data reveals, this is a good approximation of the actual states of power consumption for the washer. It appears that during the final cycle of the washer (the rinse cycle), there is a spike before the power consumption stabilizes. The spike is around 800 Watts, but the stabilized power value² is slightly below 800 Watts. Because the features extracted for the purpose of energy estimation (X) were change in power consumption (ΔP), the algorithm uses the 800 Watts value. This results in the energy estimation being slightly higher than actual values. This problem could be remedied by the selection of a larger window size during feature extraction (step A in Figure 3.2), but that comes at the cost of the possibility of multiple events within the selected window coming from appliances other than the one for which energy is being estimated. This adversely affects the energy estimation.

Authors in [47] present a data-driven algorithm to separate the spikes (known as transient phase) from the steady-states using adaptive window sizes. They note that these two states of power consumption would cluster separately if the features are mapped on 2-D plane using a density-based clustering algorithm. Once this is achieved, the ΔP values can trivially estimated using the steady-state clusters, which would be ideal for the energy estimation framework. Figure 5.4, illustrates this process. A 900 samples long event window of real power (P) and reactive power

²also known as steady-state of power consumption

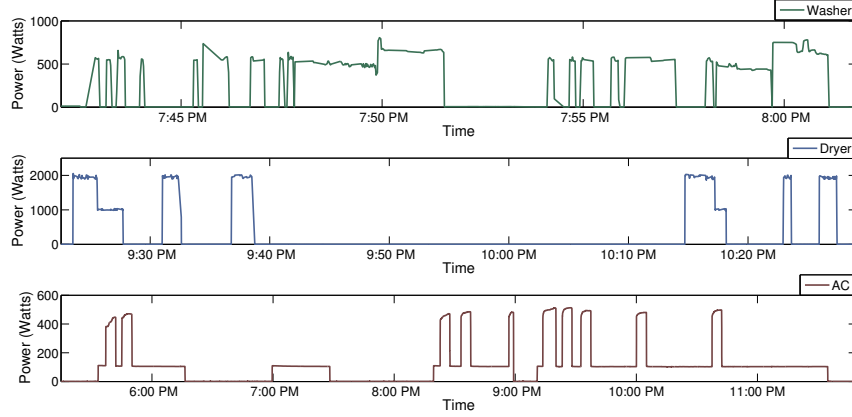


Figure 5.3: Ground truth power traces for two cycles each of washer (top), dryer (middle), and AC (bottom) as measured by Firefly plug meters on a specific day (24th February 2015)

(Q) for a refrigerator are taken from a publicly available dataset of residential power consumption called BLUED [2]. We then map it to the PQ plane, where it is observed that the two steady states separate as clusters, and the transient appears as noise. We used a density based clustering algorithm called DBSCAN to detect the appropriate clusters [52]. The difference between the means (or exemplars) of the P values of the two clusters gives the required ΔP value, which is not affected by transient spikes. The computational complexity of their method, as is, has dissuaded us from incorporating their framework into the virtual metering framework. The method was also designed for high frequency data (~ 60 Hz), and requires reactive power values as well. We leave the modification of this particular method to suit lower frequency data (~ 1 Hz) and its incorporation into the virtual metering framework as future work. It is also worth noting that the energy estimation framework assumes power values to be piece-wise constants between events. Clearly, Figure 5.3 shows that the power values exhibit considerable fluctuation between events. This most likely explains part of the observed error. The errors observed for fan and

refrigerator can also most likely be attributed to this limitation. As a quick test, we passed the ground truth signal through a one-dimensional median filter of window size 100 to simulate a case where the fluctuations were minimized. As a result, the error rates for the refrigerator decreased, while the error for the ran remained the same. As the authors note in [24], there has been some work done on modeling power consumption behavior between events for different kinds of appliances [59]. Incorporation of such models would likely reduce some of this error.

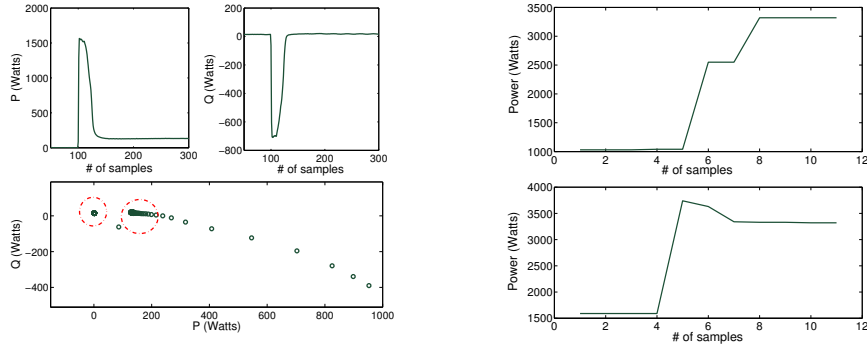


Figure 5.4: **Left (a):** P and Q power trace of a refrigerator mapped in the PQ plane to extract steady states. **Right(b):** Simultaneous events occurring within a window (above) and transient spikes that can affect ΔP computation (below) for two event instances of a dryer.

The error in the dryer appears to result from similar limitations imparted by transient spikes in the framework. The perturbation step (Step B) estimates one of the power transitions at 2270 Watts, which is higher than the observed values in Figure 5.3. This is also captured in Figure 5.4, where clearly the spike (that influences the ΔP calculation) is at a higher level of power consumption than the steady state. In addition, the error observed in the dryer highlights another limitation of the framework: its handling of simultaneous events. Typically, when an event occurs, ΔP is calculated over a certain window size. But there is a possibility

that some other appliance might have caused an event within that window, as shown in Figure 5.4. This affects the estimation of ΔP , which in turn affects the results. Although the energy estimation framework has the ability to discard noisy events during estimation of ΔP values, if the number of events are small to begin with (like it is in the case of the dryer), it can be affected by such results. This issue, however, will resolve itself as the system continues to collect more data, and consequently, more events. In the de-noising step in the energy estimation framework (Chapter 3.3), any cluster with less than a certain number of elements (ν) is classified as noise. Here, we refer to state transitions corrupted by simultaneous events, slow loading transients, or spikes resulting in an inaccurate ΔP value computation as noisy transitions, and the rest as true state transitions. So, if the total number of true state transitions is comparable to or less than the number of noisy values, the clustering algorithm will find clusters that includes noise as well. However, once the number of true events is greater than ν , the noisy transitions will not be included in the cluster. One of the benefits of the energy estimation framework is that it builds FSM models for the appliance being metered. Once built, this gives the system a good idea of what kinds of power transitions to expect. So, in cases where there are multiple events within the window, as in Figure 5.4, it can choose the event that the appliance of interest most likely caused. Use of adaptive windows as opposed to fixed ones, as mentioned in the method we talked about previously, would also most likely resolve this issue.

The error in AC appear to come from the inability of the sensors to detect events that correspond to the state where the compressor automatically switches off but the fan is still working. As can be seen in Figure 5.3, when only the fan is operating,

the AC has a consistent power draw of ~ 105 Watts. But because the fan is inside the AC box and its magnetic field signal is not strong enough, the HOBO sensor is unable to detect it most of the times. This shortcoming could be addressed in multiple ways, either by using a more powerful magnetic field sensor (for e.g., [21]), or by using other sensors (for e.g., sound) to detect events. This issue underscores the importance of the choice of sensors being used in virtual metering; the sensor (or a combination of them) has to be powerful enough to detect all possible state changes within the appliance being metered. The error in energy computation for the vacuum can again be attributed to improper ΔP computations resulting from simultaneous events. In this particular case, these results are made worse by a drift in the internal clock of the sensor which resulted in the events being recorded sometimes at slightly different timestamps than the one logged by the main meter. Proper clock synchronization is another important issue that needs to be considered when appliances are virtually metered.

The experiment did not involve electric appliances that have variable power draws. The virtual metering framework is event-based, in that it extracts features everytime an event of interest occurs. Variable loads (like a variable speed drive) will not have distinct events but different levels of power consumption that continuously vary through its operation. The proposed method is not able to handle such cases. Similarly, certain electronic devices (for e.g., a Router) might not have distinct events but continuously varying states of power draw. The framework is not able to meter such appliances either. In addition, the error correction framework that has been utilized has a specific definition for errors- namely state transitions that violate a given FSM. Sometimes, errors can occur that do not necessarily vi-

olate a FSM, but still result in a highly unlikely sequence of events. Ideally, the framework should be able to detect these and correct for them. This requires the framework to have seen enough data to correctly identify which sequences are less likely, as defined by some probabilistic likelihood criteria. As it stands right now, the framework is unable to account for such errors. Similarly, an understated but implicit assumption has been that the framework receives enough events to build a realistic FSM model of appliances. If the framework has not received enough data prior to building an FSM model, then each time it sees an unobserved event (state change), it is likely to be classified as an error, which can adversely impact the energy estimates. Automatically learning when enough new events are observed and adapting the model to accomodate them is another significant extension that the framework could use.

Despite the error values, the results from the experiment indicate that virtual metering using the utilized framework holds promise as a practical and accurate alternative to plugs. The validation scenario was made as realistic as possible, with multiple appliances being turned on and off inside the room while the appliances of interest were operating. Enough events were observed to extract all states of operations for such appliances. Although the sensors used in the case study were chosen because of ease of access and deployment, cheaper alternatives exist which makes the proposition of virtual metering more enticing. The sensors were priced at \$92 a piece which is comparable to the prices of plug-meters. But similar sensors, like the ones we proposed in Chapter 2, are estimated at roughly \$2 a piece, and can be used identically with the framework used in this chapter. Similarly, although only magnetic field sensors were used as binary indicators of state changes,

other sensors like light, sound, etc. could have been used, and we expect similar results. In addition to ease of deployment and cost effectiveness, the proposed form of virtual metering is also practical because it requires minimal calibration³, and it facilitates both real time and offline computations. It does not necessarily require a networking interface to send events as they get sensed, and a logger, much like the one used in the experiments proposed, works just as well. The framework also does not require the sensor nodes to have any advanced computational capabilities other than the ability to log binary values in memory. These features add to the practicality of the solution. The setup does, however, need to know which sensors are placed to monitor what kinds of appliances. So, even though no work is required for calibration, there is initial work involved on the part of the user in labeling the sensors nodes appropriately. A user-friendly interface that can receive data from a wide range of sensors and easily associate it with appliances would be a good solution to counteract some of the inertia associated with manual intervention.

Another feature of the framework that is particularly useful is its ability to create a model for the appliance and correct for errors in the data sent by the contactless sensors. The models contain information about the kinds of state transitions that are possible and use that to correct any violations of said model in the sequence of events that get received. This helps counteract the effect of errors due to ambient noise, data corruption during communication, false events, etc. The same model can also be used for possible fault detection and diagnosis purposes, or to alert users for performance degradation issues and pre-emptive maintenance. We believe that until appliances become smart enough to report their own power consumption, or plug-

³it requires basic thresholding of the virtual sensors, and calibration of the electric meter at the mains

meters become cost-effective, virtual metering is a good alternative with multiple benefits. It can integrate seamlessly into existing infrastructure and provide tangible benefits.

5.5 Chapter Conclusion

In this chapter, we present a case for virtual metering of appliances using data provided by external sensors, also referred to as contactless sensors (for e.g., sound, light, magnetic field, vibration, etc.). We utilize the frameworks in Chapters 3 and 4 and propose its use as the computational platform for processing binary state change data as reported by contactless sensors in order to meter appliances. In a case study involving 6 appliances, we show that the framework, using appliance state change data from magnetic field sensors and aggregate data from the main electrical panel, can perform accurate metering. This setup for energy metering requires minimal calibration and is easy to deploy and is cost-effective. Hence, it holds promise as a practical solution for energy metering of electrical appliances. Future work needs to look at making user-friendly interfaces for seamless labeling of sensors, and integration of sensor data with aggregate data.

Chapter 6

Summary

6.1 Summary and Broader Impact

In summary, this thesis explored the problem of energy estimation in supervised NILM in considerable detail and proposed an algorithmic framework for estimating energy that overcomes some of the challenges traditionally associated with event-based NILM. The model is agnostic to how the event labels are generated, which allows it to be used with any NILM event detection and classification algorithms. One of the major highlights of the framework is its ability to learn the FSM models for appliances through a completely data driven approach. This knowledge can then allow the framework to (1) identify certain kinds of errors caused during preceding steps (2) constrain the resulting energy values to lie between a certain range. The error correction algorithm stands as an independent entity, which can be used for correcting errors in sequences generated by processes other than appliance state changes as well. Our ultimate contribution, if summarized in one line, is *a framework for using a time-series data that has been coded to assume a finite number of states and defined transitions, and has passed through a channel which introduces*

certain types of errors (insertion, deletion or substitution), to (1) decode the underlying model of states, state transitions, and transition probabilities using prior beliefs about the process (2) correct for errors imparted by the channel on the observed sequence based, again, on prior beliefs. As mentioned previously, we believe, parts of this approach (notably decoding FSMs and error correction using prior beliefs) can be translated to parallel fields like DNA sequencing, speech recognition, etc., where the system assumes distinct states, the channel imparts known kinds of errors on the observed sequence, and the coding scheme is not always known. We have extended the use-case of the framework to the field of virtual metering and shown that it provides a practical and accurate way to meter appliances of interest. In addition to utilizing publicly available datasets where available, we have also collected our own data when needed, and labelled existing datasets when applicable, thereby increasing the repository of available datasets in the field considerably. Because we are introducing the first formalized framework dedicated for energy estimation, comparison with other methods or even benchmarking against a standard is not meaningful. But we hope we have opened up the avenue for more interesting contributions and extensions in this particular field.

6.2 Future Work

Future improvements and extensions to our proposed methods are corollaries to their existing limitations. Most of them have been discussed in appropriate chapters throughout this thesis. In this section, we summarize them for easy reference.

1. We believe the most important extension to the framework is the incorporation of a data-driven way to calculate the ΔP values, as discussed in Chapter 5. We have used a fixed window size in this thesis, and it is one of the more sensitive

parameters in the framework. We have discussed some promising alternatives in Section 5.4. Related area of extension is the framework’s handling of simultaneous events, particularly when dealing with low-frequency (~ 1 Hz level) data. Again, as discussed in Chapter 5, we believe the learnt FSM model, and a correlation technique can be used to pick out the most likely event if multiple events occur during an event window.

2. We have not utilized the time duration of states in the framework. This is a very significant piece of information that has the potential to improve error correction results, and increase the types of errors the framework can detect. In addition, this also allows the framework to provide more accurate points to insert missing states during the error correction step. Time duration modeling also allows the extension of the use-case of the framework to areas of anomalous appliance behavior detection. Along the same theme, the FSM models have the potential to be utilized for fault detection of appliances (for e.g., constant transition to impossible states). It can also be used to track deviation of FSM over time, which can given users some idea of how the appliance is deteriorating. These topics are not well studied in literature and provide rich content for research. Similarly, the learnt FSM model can also be used as a check to validate user generated appliance labels in certain use cases.
3. The error correction framework has a very particular definition of error that is Markovian in nature. Allowing for other kinds of errors (perhaps unlikely but possible state transitions over multiple lengths) is an interesting research problem that can generalize well beyond NILM problems. It has also implicitly assumed that the initial and the final nodes of the sequence (c_1 and c_m)

are fixed and not erroneous. Obviously, this is not guaranteed. A more robust way to find the starting and ending nodes for correction might be to utilize the the aggregate power information, and track it for baselines (/background power). Once a reasonable baseline has been established, everytime the aggregate power consumption reaches the baseline, it is guaranteed that the last event from the appliance is an *off* from its previous state, and the next event will be from *off* to a state of positive power consumption. Tracking baselines is an ongoing area of research, and error correction promises to be an interesting use case.

4. As discussed in Chapter 3, we have assumed power levels to be piece-wise constants between consecutive events throughout the framework. We believe (as discussed in Chapter 5), part of the observed errors in energy estimation are probably due to this assumption. Incorporation of appliance specific power consumption patterns between events promises to be an interesting addition as well. The framework is also completely event based, and is unable to handle cases where loads exhibit variable power consumption patterns (e.g., variable speed drives). Incorporation of such appliance types into this (or a similar) framework promises to be a challenging, albeit interesting research task.
5. When utilized for virtual metering, the framework can be augmented by an interface that makes it easy to label virtual sensors. Although not a particularly difficult problem, this can have significant impact in lowering the barrier of adoption to this solution. Future work in this area needs to consider this extension.

Bibliography

- [1] K. Ehrhardt-Martinez, K. Donnelly, J. A. S. Laitner, D. York, J. Talbot, and K. Friedrich, “Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Economy-wide Electricity Savings.” American Council for an Energy-Efficient Economy, Washington, D.C., Tech. Rep. E105, Jun. 2010.
- [2] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, “Blued: A fully labeled public dataset for {Event-Based} {Non-Intrusive} load monitoring research,” 2012.
- [3] O. Parson, S. Ghosh, M. Weal, and A. Rogers, “Non-intrusive load monitoring using prior models of general appliance types,” in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, Dec. 2012. [Online]. Available: <http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4809>
- [4] United States, Department of Energy, and Office of Energy Efficiency and Renewable Energy, “Buildings energy databook,” *Buildings energy databook*, 2011.
- [5] W. B. Seaver and A. H. Patterson, “Decreasing fuel-oil consumption through feedback and social commendation,” *Journal of Applied Behavior*

- Analysis*, vol. 9, no. 2, pp. 147–152, 1976. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1311919/>
- [6] S. Darby, “The effectiveness of Feedback on energy Consumption,” *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, April, 2006.
 - [7] M. L. Dennis, E. Jonathan, W. S. Koncinski, and B. Cavanaugh, “Effective dissemination of energy-related information: Applying social psychology and evaluation research,” *American Psychologist*, vol. 45, no. 10, pp. 1109–1117, 1990.
 - [8] J. Dobson and J. Griffin, “Conservation Effect of Immediate Electricity Cost Feedback on Residential Consumption Behaviour,” in *Proceedings of the ACEEE 1992 Summer Study on Energy Efficiency in Buildings*, vol. 10, Washington, D.C., 1992, pp. 33–35.
 - [9] T. Ueno, R. Inada, O. Saeki, and K. Tsuji, “Effectiveness of an energy-consumption information system for residential buildings,” *Applied Energy*, vol. 83, no. 8, pp. 868–883, Aug. 2006. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S030626190500125X>
 - [10] T. Ueno, K. Tsuji, and Y. Nakano, “Effectiveness of Displaying Energy Consumption Data in Residential Buildings: To Know is to Change,” in *Proceedings of the ACEEE 2006 Summer Study on Energy Efficiency in Buildings*, vol. 7, Washington, D.C., Aug. 2006, pp. 264–277.
 - [11] G. Wood and M. Newborough, “Influencing user behaviour with energy information display systems for intelligent homes,” *International Journal*

- of Energy Research*, vol. 31, no. 1, pp. 56–78, 2007. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/er.1228/abstract>
- [12] P. Karbo and T. Larsen, “Use of online measurement data for electricity savings in Denmark,” in *Proceedings of the ECEEE 2005 Summer Study on Energy Efficiency*, vol. 1, France, 2005, p. 1804. [Online]. Available: http://www.ecee.org/library/conference_proceedings/ecee_Summer_Studies/2005c/Panel_1/1180karbo
- [13] M. E. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, “Enhancing Electricity Audits in Residential Buildings with Nonintrusive Load Monitoring,” *Journal of Industrial Ecology*, vol. 14, no. 5, pp. 844–858, Oct. 2010. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1111/j.1530-9290.2010.00280.x/abstract>
- [14] G. Hart, “Nonintrusive appliance load monitoring,” *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.
- [15] L. K. Norford and S. B. Leeb, “Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms,” *Energy and Buildings*, vol. 24, no. 1, pp. 51–64, 1996. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0378778895009582>
- [16] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd, “At the flick of a switch: Detecting and classifying unique electrical events on the residential power line,” in *UbiComp 2007: Ubiquitous Computing*, 2007, pp. 271–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74853-3_16

- [17] S. Gupta, M. S. Reynolds, and 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*, ser. Ubicomp ’10. New York, NY, USA: ACM, 2010, pp. 139–148. [Online]. Available: <http://doi.acm.org/10.1145/1864349.1864375>
- [18] J. Liang, S. Ng, G. Kendall, and J. Cheng, “Load signature study part I: Basic concept, structure and methodology,” in *2010 IEEE Power and Energy Society General Meeting*, Jul. 2010, p. 1.
- [19] M. Zeifman and K. Roth, “Nonintrusive appliance load monitoring: Review and outlook,” in *2011 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2011, pp. 239–240.
- [20] A. Schoofs, A. Guerrieri, D. Delaney, G. O’Hare, and A. Ruzzelli, “ANNOT: Automated Electricity Data Annotation Using Wireless Sensor Networks,” in *2010 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, Jun. 2010, pp. 1–9.
- [21] A. Rowe, M. Berges, and R. Rajkumar, “Contactless sensing of appliance state transitions through variations in electromagnetic fields,” in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, ser. BuildSys ’10. New York, NY, USA: ACM, 2010, pp. 19–24. [Online]. Available: <http://doi.acm.org/10.1145/1878431.1878437>
- [22] A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar, “Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey,” *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, Dec. 2012. [Online]. Available: <http://www.mdpi.com/1424-8220/12/12/16838>

- [23] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert, “Is disaggregation the holy grail of energy efficiency? the case of electricity,” *Energy Policy*, vol. 52, pp. 213–234, Jan. 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0301421512007446>
- [24] S. Giri and M. Berges, “An energy estimation framework for event-based methods in Non-Intrusive Load Monitoring,” *Energy Conversion and Management*, vol. 90, pp. 488–498, Jan. 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0196890414010164>
- [25] K. Anderson, J. Moura, and M. BergÅls, “Unsupervised approximate power trace decomposition algorithm,” in *Non-Intrusive Load Monitoring Workshop 2014*, Austin, Texas, USA, Jun. 2014.
- [26] M. J. Johnson and A. S. Willsky, “Bayesian nonparametric hidden semi-markov models,” arXiv e-print 1203.1365, Mar. 2012. [Online]. Available: <http://arxiv.org/abs/1203.1365>
- [27] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, “Unsupervised disaggregation of low frequency power measurements,” in *SIAM International Conference on Data Mining (SDM 11)*, Mesa, Arizona, Apr. 2011.
- [28] J. Z. Kolter and T. Jaakkola, “Approximate inference in additive factorial HMMs with application to energy disaggregation,” in *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 1472–1482. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/papers/AISTATS2012_KolterJ12

- [29] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Elsevier, Jan. 1977.
- [30] X. Yang, S. P. Chockalingam, and S. Aluru, “A survey of error-correction methods for next-generation sequencing,” *Briefings in Bioinformatics*, vol. 14, no. 1, pp. 56–66, Jan. 2013.
- [31] G. Hart and A. Bouloutas, “Correcting dependent errors in sequences generated by finite-state processes,” *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1249–1260, 1993.
- [32] G. W. Hart, “Minimum information estimation of structure,” Thesis, Massachusetts Institute of Technology, 1987, thesis (Ph. D.)—Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1987. [Online]. Available: <http://dspace.mit.edu/handle/1721.1/14792>
- [33] S. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, “Power-analysis attack on an ASIC AES implementation,” in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004*, vol. 2, Apr. 2004, pp. 546–552 Vol.2.
- [34] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, “Towards Sound Approaches to Counteract Power-Analysis Attacks,” in *Advances in Cryptology — CRYPTO 99*, ser. Lecture Notes in Computer Science, M. Wiener, Ed. Springer Berlin Heidelberg, 1999, no. 1666, pp. 398–412. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-48405-1_26
- [35] Y. Kim, T. Schmid, Z. Charbiwala, and M. Srivastava, “ViridiScope: design and implementation of a fine grained power monitoring system

- for homes.” ACM, 2009, pp. 245–254. [Online]. Available: <http://dx.doi.org/10.1145/1620545.1620582>
- [36] N. Rajagopal, S. Giri, A. Rowe, and M. Berges, “A Magnetic Field-based Appliance Metering System,” in *Third International Conference on Cyber-Physical Systems (ICCPS)*, Philadelphia, PA, 2013.
- [37] Z. Kolter and M. Johnson, “Redd: A public data set for energy disaggregation research,” in *In proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 2011.
- [38] M. Ostendorf and H. Singer, “Hmm topology design using maximum likelihood successive state splitting,” *Computer Speech & Language*, vol. 11, no. 1, pp. 17–41, 1997.
- [39] C. Li and G. Biswas, “Temporal pattern generation using hidden markov model based unsupervised classification,” in *Advances in Intelligent data analysis*. Springer, 1999, pp. 245–256.
- [40] L. R. Rabiner, “Readings in speech recognition,” A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296. [Online]. Available: <http://dl.acm.org/citation.cfm?id=108235.108253>
- [41] Buevich, M., Rowe, A., Rajkumar, R., “Tracking and visualization of building energ,” *1st International Workshop on Cyber-Physical Systems, Networks, and Applications (CPSNA 11)*, 2011.

- [42] Rowe A., Berges M., Bhatia G., Goldman E., Rajkumar R., Soibelman L., Garrett J., Moura J. , “Demonstrating Sensor Andrew: Large-Scale Campus-Wide Sensing and Actuation,” *Demo Abstract, IPSN*, 2009.
- [43] M. Baranski and J. Voss, “Genetic algorithm for pattern detection in NIALM systems,” in *Systems, Man and Cybernetics, 2004 IEEE*, vol. 4. The Hague, The Netherlands: IEEE, 2004, pp. 3462–3468 vol.4. [Online]. Available: <http://ieeexplore.ieee.org/iel5/9622/30424/01400878.pdf>
- [44] —, “Detecting patterns of appliances from total load data using a dynamic programming approach,” in *Fourth IEEE International Conference on Data Mining (ICDM’04)*, Brighton, UK, 2004, pp. 327–330. [Online]. Available: <http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F9681%2F30565%2F01410302.pdf%3Farnumber%3D1410302&authDecision=-203>
- [45] R. Streubel and B. Yang, “Identification of electrical appliances via analysis of power consumption,” in *Universities Power Engineering Conference (UPEC), 2012 47th International*, Sep. 2012, pp. 1–6.
- [46] K. Barsim, R. Streubel, and B. Yang, “An approach for unsupervised non-intrusive load monitoring of residential appliances,” in *In proceedings of the 2nd Workshop for Non-Intrusive Load Monitoring*, Austin, Texas, USA, Jun. 2014. [Online]. Available: http://nilmworkshop14.files.wordpress.com/2014/05/barsim_approach.pdf
- [47] —, “Unsupervised adaptive event detection for building-level energy disaggregation,” in *In proceedings of Power and Energy Student Summit*

- (PESS), Stuttgart, Germany, Jan. 2014. [Online]. Available: http://www.iss.uni-stuttgart.de/forschung/veroeffentlichungen/barsim_pess2014.pdf
- [48] R. Xu and I. Wunsch, D., “Survey of clustering algorithms,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.
- [49] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, Jan. 2001. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1111/1467-9868.00293/abstract>
- [50] D. Arthur and S. Vassilvitskii, “K-means++: the advantages of careful seeding,” in *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.
- [51] K. Kerdprasop, N. Kerdprasop, and P. Sattayatham, “Weighted k-means for density-biased clustering,” in *In DaWaK*, 2005, p. 488–497.
- [52] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *KDD*, vol. 96, 1996, pp. 226–231.
- [53] M. Ankerst, M. M. Breunig, H.-p. Kriegel, and J. Sander, “OPTICS: ordering points to identify the clustering structure.” ACM Press, 1999, p. 49–60.
- [54] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, p. 849–856.

- [55] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007, PMID: 17218491. [Online]. Available: <http://www.sciencemag.org/content/315/5814/972>
- [56] L. Zelnik-manor and P. Perona, “Self-tuning spectral clustering,” in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004, p. 1601–1608.
- [57] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, p. 849–856.
- [58] R. Tarjan, “Depth-first search and linear graph algorithms,” *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, Jun. 1972. [Online]. Available: <http://epubs.siam.org/doi/abs/10.1137/0201010>
- [59] S. Barker, S. Kalra, D. Irwin, and P. Shenoy, “Empirical characterization and modeling of electrical loads in smart homes,” in *Green Computing Conference (IGCC), 2013 International*, Jun. 2013, pp. 1–10.
- [60] S. Siddiqi, G. Gordon, and A. Moore, “Fast state discovery for hmm model selection and learning,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.
- [61] J. Ramos, S. Siddiqi, A. Dubrawski, G. Gordon, and A. Sharma, “Automatic state discovery for unstructured audio scene classification,” in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2154–2157.
- [62] M. Marceau and R. Zmeureanu, “Nonintrusive load disaggregation computer program to estimate the energy consumption of major end uses in residential

- buildings,” *Energy Conversion and Management*, vol. 41, no. 13, pp. 1389 – 1403, 2000. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0196890499001739>
- [63] S. Giri and M. Berges, “An error correction framework for sequences resulting from known state-transition models in non-intrusive load monitoring (under review),” *Advanced Engineering Informatics*, vol. NA, no. 1, p. NA, Mar. 2015.
- [64] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. John Wiley & Sons, Sep. 2011.
- [65] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959. [Online]. Available: <http://link.springer.com/article/10.1007/BF01386390>
- [66] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan, “Faster algorithms for the shortest path problem,” *Journal of the Association for Computing Machinery*, vol. 37, no. 2, pp. 213–223, 1990. [Online]. Available: <http://cat.inist.fr/?aModele=afficheN&cpsidt=19245968>
- [67] M. L. Fredman and R. E. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *J. ACM*, vol. 34, no. 3, pp. 596–615, Jul. 1987. [Online]. Available: <http://doi.acm.org/10.1145/28869.28874>
- [68] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

- [69] D. J. Burr, “Designing a handwriting reader,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 5, pp. 554–559, May 1983. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.1983.4767435>
- [70] F. Jazizadeh, B. Becerik-Gerber, M. Berges, and L. Soibelman, “An unsupervised hierarchical clustering based heuristic algorithm for facilitated training of electricity consumption disaggregation systems,” *Advanced Engineering Informatics*, vol. 28, no. 4, pp. 311–326, Oct. 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1474034614000913>
- [71] S. Giri and M. Berges, “Virtual metering of electrical appliances through analysis of data from contactless sensing (under review),” *Energy and Buildings*, vol. NA, no. 1, p. NA, Mar. 2015.
- [72] TED Inc., “The energy detective.” [Online]. Available: <http://www.theenergydetective.com/>
- [73] “Wireless 3-Phase Electricity Monitor.” [Online]. Available: <http://episensor.com/products/zem-61-three-phase-electricity-monitor/>
- [74] “Emonitor.” [Online]. Available: <http://powerhousedynamics.com/>
- [75] “eGauge EG 3000.” [Online]. Available: <https://www.egauge.net/products/>
- [76] “FIDO.” [Online]. Available: <https://www.egauge.net/products/>
- [77] “Energy hub.” [Online]. Available: <http://www.energyhub.com/>
- [78] “Watts up.” [Online]. Available: <https://www.wattsupmeters.com/secure>
- [79] “Kill A Watt.” [Online]. Available: <http://www.p3international.com/products>

- [80] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler, “Design and implementation of a high-fidelity AC metering network,” in *International Conference on Information Processing in Sensor Networks, 2009. IPSN 2009*, Apr. 2009, pp. 253–264.
- [81] R. Mangharam, A. Rowe, and R. Rajkumar, “FireFly: a cross-layer platform for real-time embedded wireless networks,” *Real-Time Systems*, vol. 37, no. 3, pp. 183–231, Dec. 2007. [Online]. Available: <http://dx.doi.org/10.1007/s11241-007-9028-z>
- [82] Y. Agarwal and T. Weng, “From Buildings to Smart Buildings- Sensing and Actuation to Improve Energy Efficiency,” *IEEE Design & Test of Computers*, vol. 29, no. 4, pp. 36–44, 2012.
- [83] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. Paradiso, “A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments,” in *6th International Symposium on Information Processing in Sensor Networks, 2007. IPSN 2007*, Apr. 2007, pp. 119–127.
- [84] S. O’Connell, J. Barton, E. O’Connell, B. O’Flynn, E. Popovici, S. O’Mathuna, A. Schoofs, A. Ruzzelli, and G. O’Hare, “Remote Electricity Actuation and Monitoring mote,” in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, Jun. 2011, pp. 1–6.
- [85] M. Clark, B. Campbell, and P. Dutta, “Deltaflow: Submetering by Synthesizing Uncalibrated Pulse Sensor Streams,” in *Proceedings of the 5th International Conference on Future Energy Systems*, ser. e-Energy ’14. New York, NY, USA: ACM, 2014, pp. 301–311. [Online]. Available: <http://doi.acm.org/10.1145/2602044.2602070>

- [86] D. Jung and A. Savvides, “Estimating Building Consumption Breakdowns Using ON/OFF State Sensing and Incremental Sub-meter Deployment,” in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 225–238. [Online]. Available: <http://doi.acm.org/10.1145/1869983.1870006>
- [87] S. Giri, M. Berges, and A. Rowe, “Towards automated appliance recognition using an EMF sensor in NILM platforms,” *Advanced Engineering Informatics*, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1474034613000268>
- [88] T. Wu, “Low-cost Appliance State Sensing for Energy Disaggregation,” *eScholarship*, Jan. 2012. [Online]. Available: <http://escholarship.org/uc/item/8352r9g5>
- [89] “HOBO Data Logger.” [Online]. Available: <http://www.onsetcomp.com/products/data-loggers/ux90-004>