Filtering in Non-Intrusive Load Monitoring

by

Alejandro Rodriguez Silva

M.Sc. (Microelectronics), Instituto Politécnico Nacional, 2016
B.S. (Computer Engineering), Instituto Politécnico Nacional, 2012

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2021

Declaration of Committee

Name:	**Alejandro Rodriguez Silva**
Degree:	**Master of Applied Science**
Thesis title:	**Filtering in Non-Intrusive Load Monitoring**
Committee:	**Chair:** Ljiljana Trajkovic Professor, Engineering Science
	Stephen Makonin Co-Supervisor Adjunct Professor, Engineering Science
	Rodney Vaughan Co-Supervisor Professor, Engineering Science
	Atousa Hajshirmohammadi Examiner University Lecturer, Engineering Science

Abstract

Being able to track appliances energy usage without the need of sensors can help occupants reduce their energy consumption. Non-intrusive load monitoring (NILM) is one name for this topic. One of the hardest problems NILM faces is the ability to run unsupervised – discovering appliances without prior knowledge – and to run independent of the differences in appliance mixes and operational characteristics found in various countries and regions. This thesis showcases two filters that are used to denoise power signals, which results in better clustering accuracy for NILM event based methods. Both filters show to outperform a state-of-the-art denoising filter, in terms of run-time. A fully unsupervised NILM solution is presented, the algorithm is based on a hybrid knapsack problem with a Gaussian mixture model. Finally, a novel metric is developed to measure NILM disaggregation performance. The metric shows to be robust under a set of fundamental test cases.

Keywords: non-intrusive load monitoring; filtering; unsupervised; Gaussian mixture model; knapsack problem

Notes

At various points in this thesis, previously published work will be leveraged. The relevant sections/subsections for each of these are listed as follows:

- Hybrid knapsack-GMM: Section 3.2. Published [1].
- Steady-state block filter: Section 2.2. Published [2].

Acknowledgements

To my supervisors Stephen and Rodney, you gave me your hand when I needed it the most.

Podría escribir otra tesis de las personas que voy a mencionar a continuación, pero por limitaciones de espacio, seré breve.

A mis familia en México: Mary Carmen, Juan, Diego, Ángela y Ramón. Este documento es tanto mío como suyo. Ustedes son mi razón y mi inspiración.

A mis amigos: Juan David, Juan Gabriel, y Gerardo. No tengo palabras para agradecerles todo su ayuda. Durante todo este tiempo me sentía perdido y sin rumbo, pero ustedes siempre estuvieron ahí para mí.

A mi familia en Canadá (en orden alfabético para no meterme en problemas): Aldou, Astrid, Juan David, Juan Gabriel, Manu, y Richard (either learn Spanish or use the Google translator). Por todas las experiencias que tenemos juntos y por su amistad incondicional. Mi vida en Canadá sería algo lúgubre sin ustedes, pero ustedes me hacen sentir como en casa.

Contents

List of Tables

List of Figures

Acronyms

BSS *Blind Source Separation.* 27

CP Change Point. 4

CUSUM *Cumulative Sum.* 4

DSP *Digital Signal Processing.* 55

DTW *Dynamic Time Warping.* 27

EM *Expectation-Maximization.* 28

FIR *Finite Impulse Response.* 6

GMM *Gaussian Mixture Model.* 4

GSP *Graphical Signal Processing.* 7

HMM *Hidden Markov Model.* 6

i.i.d. *independent and identically distributed.* 11

KP *Knapsack Problem.* 4

MP *Matching Pursuit.* 27

NDE *Normalized Disaggregation Error.* 34

NILM *Non-Intrusive Load Monitoring.* 1

NMF *Non-negative Matrix Factorization.* 26

PAM *Power Assignation Metric.* 38

PF *Particle Filter.* 27

QP *Quadratic Programming*. 4

RMSE *Root-Mean-Square Error*. 15

SCP *Switch Continuity Principle*. 30

Chapter 1

Introduction

1.1 The Non-Intrusive Load Monitoring Problem

Non-Intrusive Load Monitoring (NILM) is a research field focused on breaking down the composed appliances that contribute to a total aggregate signal. The process is performed using only the aggregate signal, which is also known as appliance disaggregation. NILM is carried out using a building's smart meter to track or sense appliance usage (see Figure 1.1) [3]. The first peer-reviewed NILM publication [3] was in 1992 and with the recent release of publicly available datasets [4, 5, 6, 7, 8] there is a resurgence in interest in solving NILM.

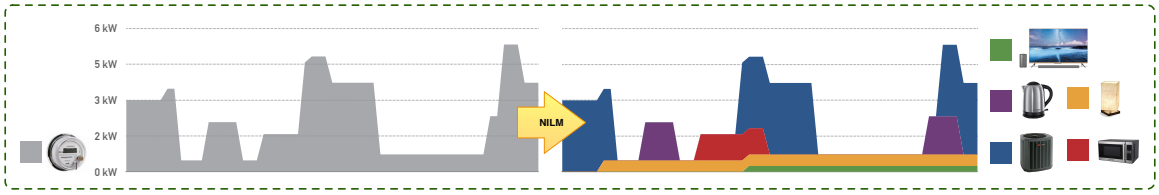

Figure 1.1: Sensing appliance power usage and consumption without sensors [1].

The task of disaggregating appliances in a system is not simple, due to the fact that appliances in residential buildings often overlap significantly [9]. The NILM problem can be formulated as solving for each $y_t^{(m)}$

$$y_t = \sum_{m=1}^{M} y_t^{(m)} + \epsilon, \tag{1.1}$$

where $y_t^{(m)}$ is the power consumption at time t associated with appliance m out of a total of M appliances that constitute the total power aggregate y_t, and ϵ is some measurement noise. Although there exists several methods which attempt solving NILM, in general the workflow consists of gathering data which, later, is processed in order to extract relevant features essential for appliance identification, and used to infer or approximate $y_t^{(m)}$. The first attempt for solving NILM was the seminal work presented in [3]. Since then, the NILM

research community has been tremendously active. The publications vary from methods to solve the problem to the recollection of data for its future release.

A typical architecture [9, 10, 11, 12, 13, 14, 15] for solving NILM is shown in Figure 1.2. Here, the aggregate y_t is composed of M ground-truth appliances $y_t^{(m)}$. Later, the aggregate signal is passed through a filtering process in order to remove noise and transient behaviour, we denote $\hat{y}_t$ to the filtered aggregate signal. Once power values are more consistent due to the steady-state filter, a source separation algorithm (typically consisting of an unsupervised clustering algorithm) extracts the edges from the filtered signal $\hat{y}_t$ and separates it into N disaggregated signals $\hat{y}_t^{(n)}$. Then, disaggregation performance is evaluated by comparing the source separator's behaviour, this is typically carried out by measuring the error between the ground-truth signals and the disaggregated signals. Finally, labels are applied to the disaggregated signals, which can be achieved via deep learning [9].

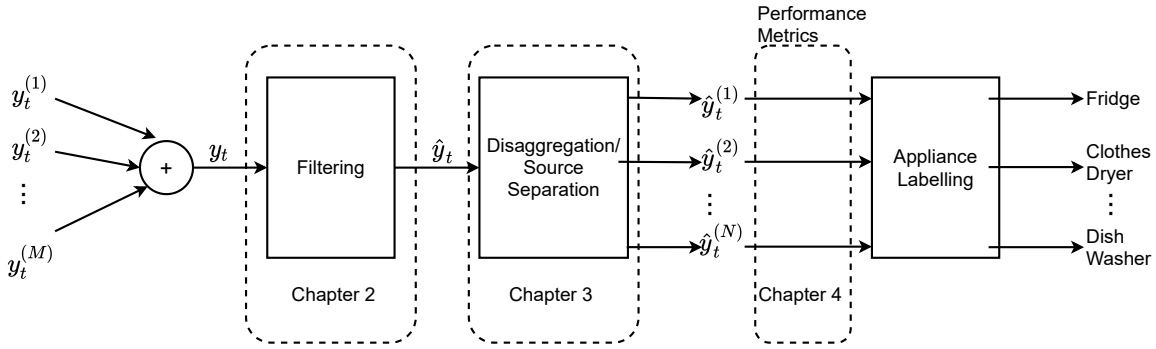

Figure 1.2: A typical NILM architecture.

Solving NILM is not an easy task. This is mainly due to the next reasons [16, 17]:

1. **Multiple, simultaneous load events:** Appliances switching ON/OFF or changing states can cause a system to miss-classify active loads. E.g., if we have two 100 Watts lights and one 200 Watts blender, turning the two lights ON simultaneously could lead to the system classifying the event as a the blender.

2. **Noisy power signals:** Electrical systems are noisy in nature. Less noisy power readings will result in a more accurate disaggregation system (see Section 2.1).

3. **Dynamic and changing usage:** The number of appliances in a home can increase or decrease over time. Also, the appliances can be replaced (i.e., an old fridge gets replaced with a newer and more energy efficient model). Furthermore, the occupant-home interaction significantly varies from one home to another. Therefore, a NILM system might have trouble when trying to generalize over data from different homes or other periods of time.

1.2 Previous Attempts for Solving NILM

NILM solutions can be categorised into supervised and unsupervised learning methods [18]. Supervised learning methods require labeled data from individual appliances in order to be trained [19, 20, 21]. Although these methods may show promise, they require a large amount data to train the models and sometimes having access to the data is not available or does not exist. Unsupervised learning methods, on the other hand, do not require individual appliance data. Moreover, these methods are more robust than supervised learning techniques due to the fact that they do not depend on the number of appliances and they can be further adapted in time (i.e., if a new appliance is added or an appliance is removed). Figure 1.3 shows a comparison between supervised and unsupervised learning techniques.

It is clear that at best, NILM is a semi-supervised method as labelled data is needed to properly identify what appliances have been disaggregated [9, 10]. By setting the labelling problem aside, one could potentially solve the disaggregation problem via a supervised or an unsupervised learning algorithm. Thus, an unsupervised NILM method refers to performing disaggregation by using aggregate data only. From now on, the scope of this thesis will be on the unsupervised learning methods.

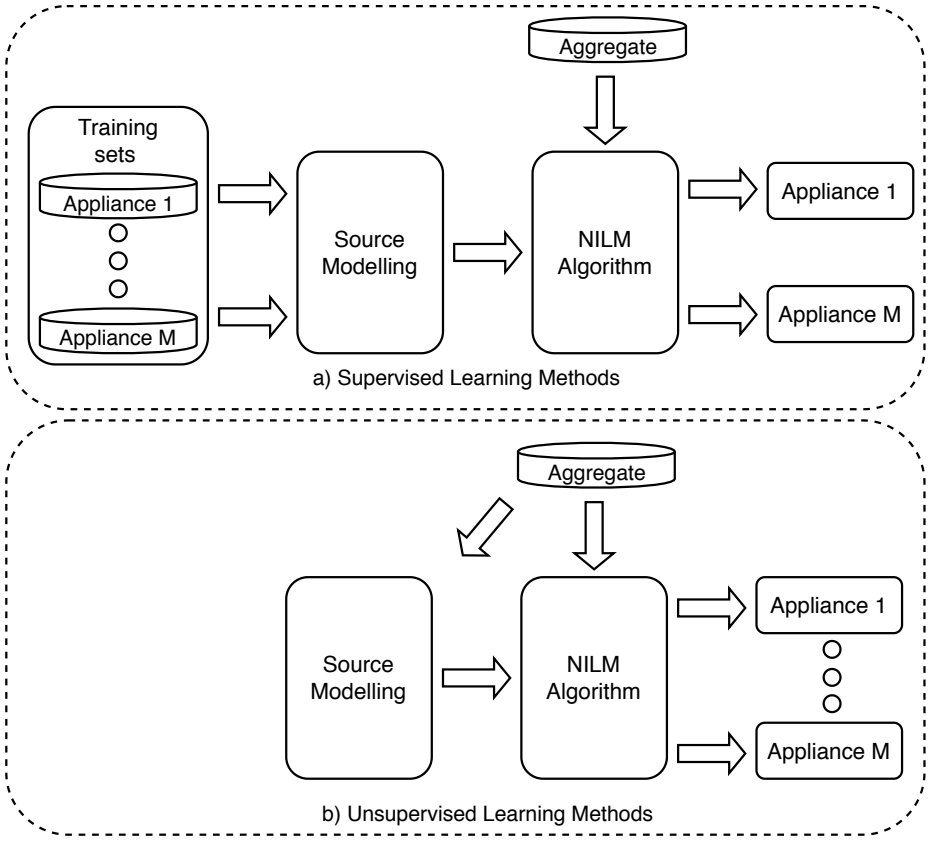

Figure 1.3: Comparison of supervised and unsupervised learning techniques, adapted from [18].

1.3 Our Contributions

This thesis attempts to solve NILM (leaving the labelling problem aside) with the workflow presented in Figure 1.2. First, we filter the aggregate signal y_t using a steady-state filter (see Chapter 2). Later, we attempt blind source separation to reconstruct the $\hat{y}_t^{(m)}$ disaggregated signals (See Chapter 3). Finally, we measure the disaggregator performance (Chapter 4). Finally, Chapter 5 summarizes the work in this thesis and highlights motivations for future work. We present three significant contributions that further advance the field of disaggregation. These are listed below.

- **Contribution 1:** We show the importance of advanced filtering techniques in event-based NILM methods with low sampling frequency (≤ 1 Hz). Chapter 2 presents two filtering approaches (see Subsections 2.2.2 and 2.2.3) to remove steady-state information and noise from aggregate signals. The first proposed filter is motivated by change point (CP) detection theory. Kalman filters and the *Cumulative Sum* (CUSUM) algorithm are the core of the second approach.

- **Contribution 2:** Another area that we have examined and published about is how to leverage the information of aggregate power signals when two or more appliances turn ON/OFF simultaneously. Chapter 3 is devoted to the development of an unsupervised clustering algorithm that does not require labelled individual appliance data. Once the raw aggregate data is passed through any of the filters in Chapter 2, the unsupervised algorithm is in charge of clustering appliances' activations/deactivations and later, it attempts to recover the disaggregated individual appliances that contribute to the aggregate. The unsupervised method is based on a hybrid *Gaussian Mixture Model* (GMM) and the *Knapsack Problem* (KP).

- **Contribution 3:** Our final contribution is our novel metric based on quadratic programming QP. In Chapter 4, the proposed metric assesses source separation performance in NILM. The metric shows to be robust under several fundamental test cases that are present in NILM architectures such as the one shown in Figure 1.2. Moreover, it automatically matches ground-truth signals with their respective inferred signals (the number and ordering between ground-truth signals and inferred signals is not always the same).

Chapter 2

Steady-State Filtering for NILM

This chapter presents a summary of the previous attempts for extracting steady-state information from aggregate data in order to allow for better clustering for appliance disaggregation. Further, we describe the methodology and implementation of three filtering approaches to remove steady-state information and noise from aggregate signals. The first filtering method is based on a combination of edge-preserving filters arranged sequentially. The second steady-state proposed filter is motivated by change point filter theory consists of a change point detection mechanism based on rolling statistics of the aggregate data. Lastly, the Kalman filter and the CUSUM algorithm are used together in order to detect changes in the aggregate signal to later remove transient or noise information.

2.1 Denoising Power Signals - A Review

Power signal noise considerably affects NILM algorithms' performance, in either of the two NILM sub-divisions: event-based or state-based [22, 23, 10, 24, 18, 25]. Event-based NILM methods rely on edge detection. A supervised or unsupervised clustering algorithm is applied to the detected edges. Ultimately, these activations (positive edges) are paired with their corresponding deactivations (negative edges). If the events or edges are not as consistent as possible, the event-based methods suffer from mis-classifying appliances that operate in a similar power range [23, 22, 10]. On the other hand, the performance of state-based NILM methods depends on the quality of the initialization of appliance state models. This means that either the a priori knowledge has to be very accurate or there has to be a large amount of data to train the models [26, 27].

There has been some work performed on denoising power signals. Below, we present what we consider to be some of the most relevant works in the context of denoising power signals.

An event-based detection NILM algorithm was proposed in [28]. The approach performed data preprocessing before performing disaggregation. The authors claimed that

they used seven signal processing algorithms in order to get rid of disaggregation noise. However, the implementation and the code are unavailable.

The authors in [29] developed an appliance energy model generator based on regression models. Their approach relied on a "software based electricity metering solution". They assumed the appliances are equipped with network interfaces and otherwise they suggest to use additional sensors (e.g,. microphones) to record data. Prior to feeding the data to the regression algorithms, the measured power values were pre-processed (in order to obtain smoother power traces) with "finite impulse response FIR filters with a rectangular kernel function" and different filter window sizes (2 to 10).

In [30], the authors presented a NILM approach based on edge detection. Before performing NILM, the method required to filter the powers signal due to oscillations and noise. The authors implemented mean, median, and Gaussian filters with different window sizes (3, 5, 60, 70, 100). They concluded that the best filter size was 5. The authors determined this was the best parameter for their application due to the trade-off between removing noise from the signal and being able to recognize small loads. The median, mean, and Gaussian filters are well-known filters in the signal processing area [31, 32, 33], these filters are good for noise removal and edge preserving (in the case of the median and mean filters). On the other hand, the Gaussian filter removes noise components, but it smooths the signal. This has a negative effect when a NILM algorithm relies on sharp edges for event detection [23].

The authors in [27, 34] presented variations of the *Hidden Markov Model* (HMM) for an unsupervised approach. A preprocessing step needs to be carried out before feeding the data to the NILM system, this was achieved using a median filter. In [34], the method was based on a particle filter in order to approximate the posterior density of the HMM. Although the median filter removes outliers and can smooth the signal, it does not successfully sharp the signal's edges.

From the literature review carried out in this work, several NILM researchers perform a denoising step prior to disaggregation. Nevertheless, they rely on basic smoothing filters such as the mean, median, Gaussian filters. More advanced filters that have the characteristics of smoothing the signals and preserving edges can be found in the image processing literature [33] and have been used for several decades. Some examples of these filter are:

- Anisotropic diffusion filter [35, 36]. This edge-preserving filtering technique was initially developed for images. The filter was proposed as a better solution to the Gaussian kernel, which has the disadvantage of being unable to select the important regions to remove in a signal. Anisotropic diffusion is able to remove noise while preserving important information such as edges in the image. This filter requires an iterative solver [37].

- Bilateral filter. This filter was developed to be able to smooth images while preserving edges. The procedure is non iterative. Furthermore, the method relies on a nonlinear

combination of nearby image values. The bilateral filter is an alternative to the anisotropic filter [36]. The latter, involves the solution of partial differential equations. Despite the fact that the filter was first developed for images, it can be applied to one-dimensional signals [1].

- Domain transform for edge-aware image and video-processing [37]. The new approach was developed as an enhancement of the anisotropic diffusion and bilateral filters for video and images. The filter has the characteristic of preserving the edges while being considerably less computationally complex than the anisotropic diffusion and the bilateral filters.

To our knowledge, the authors in [23] were one of the first researchers to introduce more advanced filtering techniques before performing disaggregation, where a NILM approach via graphical signal processing (GSP) was developed. Two filters based on GSP were implemented: total variation regularization and a bilateral filter. In order to succeed at the event classification task, primarily, edge detection should be correctly performed. Therefore, an edge sharpening procedure was implemented. The edge sharpening method consists of merging neighbouring peaks on the differential signal that exceeds a predetermined value.

Jones et al. [2] developed a novel filter that extracts steady-state information from aggregate data. The filter removes measurement noise and transient behaviour. Furthermore, the approach is capable of preserving sharp edges which allows for a more accurate clustering. Nevertheless, the filter relies on seven parameters, which hinders optimization tasks [2, 9]. The work performed by Jones et al. [2] was motivated by our filter pipeline method (see Subsection 2.2.1). Later, Jones et al. method served as a stepping stone and motivation to our two proposed methods for removing noise from NILM signals, the Change Point filter and the CUSUM Kalman filter.

2.2 Steady-State Filters

This section describes the methods implemented in order to preserve steady-state information from power signals. The methods we will focus on are: the filter pipeline, the CP filter and the CUSUM Kalman filter.

Although the filter pipeline showed promising results, its computational time was intractable which allowed us to develop better approaches such as the CUSUM Kalman filter and the CP filter show to be better alternatives. However, we decided to include some results as these series of filters arranged sequentially served as a motivation to what we consider is an important part of this thesis and NILM.

[1]The Python code for the 1-D bilateral filter can be accessed via: https://gitlab.com/snippets/16327

2.2.1 Filter pipeline

As discussed previously, each of the filtering approaches cannot accomplish to clean a signal that results into a sharp-edge and steady-state signal. Therefore, we propose to use a filter pipeline (see Figure 2.1) for denoising purposes. The filter pipeline exploits each of the benefits that the state-of-the-art methods cannot exploit individually. A standard median filter built into Matlab or Python/Numpy (or see [30, 34, 27, 38]) was used as the first step to remove large spikes (e.g., from fridge compressor start-up). Then, the result is sent to a bilateral filter [39], followed by an anisotropic diffusion filter [36]. Next, we adapted the work in [37] in order for the filter to process one-dimensional signals. The final step consists of applying edge sharpening as in [23]. Figures 2.2 and 2.3 show the 1Hz raw aggregate signal and the resulting aggregate respectively. For further details, Figure 2.4 shows the sequential inputs and outputs to the filter pipeline (starting with the median filter and finishing with the edge sharpening filter. The same workflow from Figure 2.1).

Although the filter pipeline method manages to remove transients and preserves edges, its computational time grows as the number of filters in the pipeline increases. Moreover, the filter in [37] results in a computational burden as it is much slower relative to the other filters (this will be discussed in Section 2.4). The filter pipeline was a good first iteration in terms of transient and noise removal. However, its computational speed results in a huge burden, which makes it undeployable for real-time applications. Our next task was to investigate filters that could provide similar noise removal and at the same time being able to denoise raw aggregate rapidly.

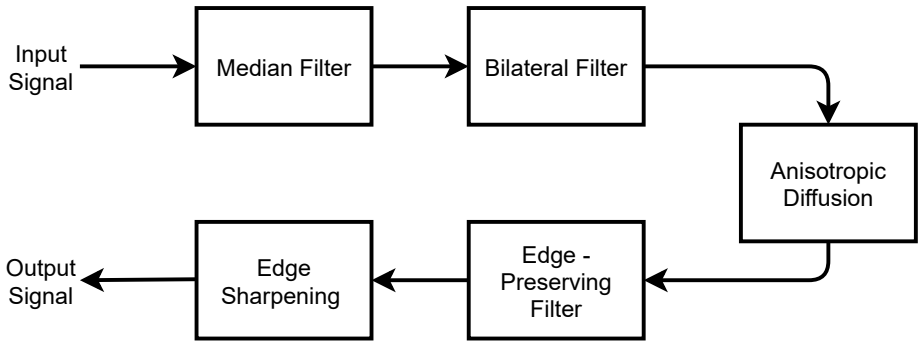

Figure 2.1: The series of filters used in our proposed filter pipeline. The order in which these filters are placed throughout the pipeline can be permuted.Filters can be added/removed.

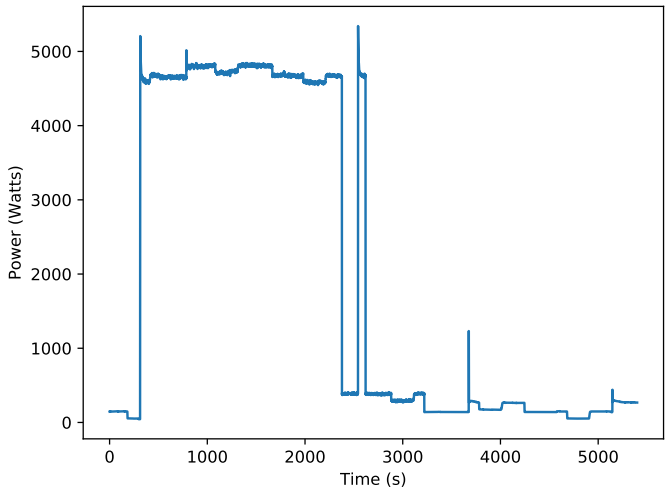

Figure 2.2: Raw 1Hz aggregate signal

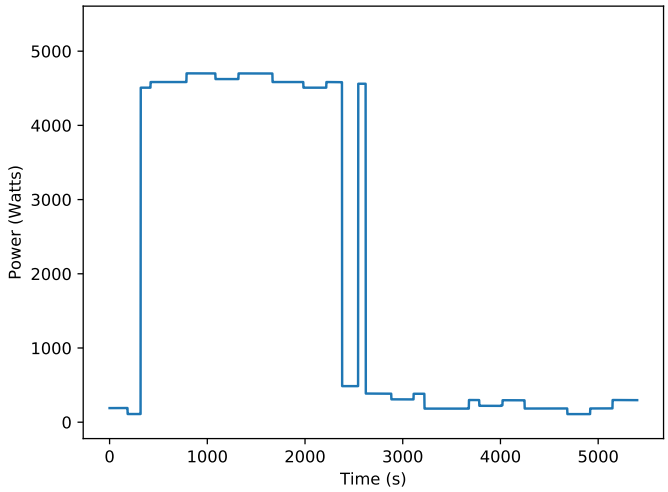

Figure 2.3: Output of our filter pipeline (Figure 2.1)

2.2.2 Change Point Filter

We want to detect when an appliance in the aggregate signal $\mathbf{y}$ change its state, e.g., it is turned ON/OFF or if it changes its operational mode. Hence, in this section, we develop a filter that is inspired by the idea of hypothesis testing [40]. We detect changes in the aggregate signal between time samples. When the change is large enough, we mark change points which we use to reconstruct the final filtered signal.

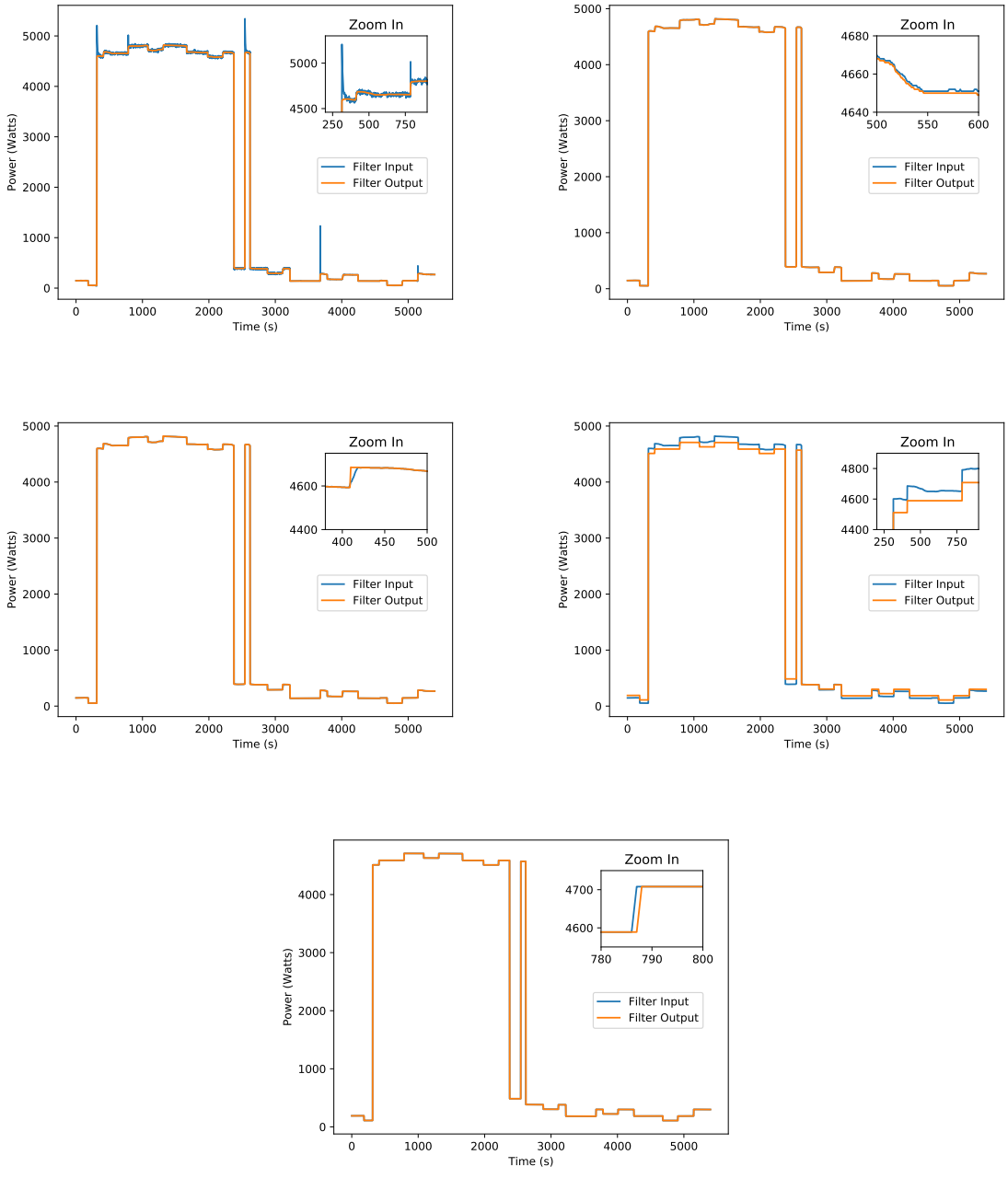

Figure 2.4: The consecutive inputs and outputs of the filter pipeline shown in Figure 2.1.

Let us suppose we have the aggregate signal $\mathbf{y}$. The signal is corrupted with additive white Gaussian noise, i.e., the noise values at any pair of times are independent and identically distributed (i.i.d.) Gaussian zero-mean. Moreover, we assume the aggregate signal $\mathbf{y}$ is piece-wise constant.

In order to detect the values of the piece-wise function $\mathbf{y}$ we need to determine what are the time instants where the aggregate signal changed. In other words, we seek the time instants where the probability of the next sample y_{t+1} coming from the same distribution of the past w samples is extremely low.

Let us introduce the moving average μ_t and the moving standard deviation σ_t of the aggregate signal $\mathbf{y}$. The moving average is given by

$$\mu_t = \frac{1}{w} \sum_{i=0}^{w-1} y_{t+i}. \tag{2.1}$$

On the other hand, the standard deviation can be calculated as

$$\sigma_t = \sqrt{\frac{1}{w-1} \sum_{i=0}^{w-1} (y_{t+i} - \mu_t)^2}. \tag{2.2}$$

A predetermined threshold $n \in \mathbb{R}^+$ is set, and if the distance (in terms of standard deviations) between the next sample y_{t+1} and the moving average μ_t exceeds the threshold, then an alarm is set. Mathematically, this can be expressed as

$$\frac{|y_{t+1} - \mu_t|}{\sigma_t} > n, \tag{2.3}$$

or equivalently, we could split the expression in (2.3) into two inequalities

$$y_{t+1} > \mu_t + n\sigma_t, \tag{2.4}$$

$$y_{t+1} < \mu_t - n\sigma_t. \tag{2.5}$$

An example of the idea is depicted in Figure 2.5. Here, the aggregate signal is composed of two cycles of the fridge. It can be observed how the right hand side of the inequalities from (2.4) and (2.5) form an envelope around the signal. However, when a sudden change occurs, the signal satisfies one of the two inequalities. Once the sudden changes in the signal have occurred, the time indexes where the events happened can be stored in a $D \times 1$ vector that we will denote $\mathbf{f}$.

Although the approach of verifying when the difference between the signal y_{t+1} and the moving average μ_t significantly increases, and recording the time where the sudden changes occurred is simple and seems promising for event detection. Problems occur when we have a group of events that happen closely and the first event's wattage is considerably larger than

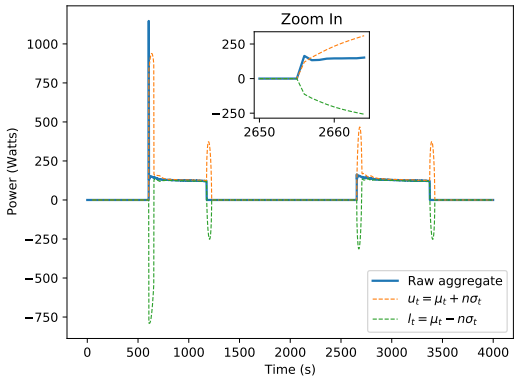

Figure 2.5: Example of a power signal. The right hand side of the inequalities in (2.4) and (2.5) form an envelope around the aggregate signal.

the preceding events. Therefore, the consecutive events will not be detected as the signal at that particular time will not exceed any of the two thresholds. Figure 2.6 shows an example of a fridge event that is not detected. The aggregate for this test case was composed of the fridge and the clothes dryer. It can be observed that when the clothes dryer turned ON approximately around $t = 240$ (peak of ≈ 5600 Watts), it caused the threshold to increase. Later, the fridge (which was already ON) turned OFF around $t = 275$. However, due to the high threshold that the clothes dryer event created, the fridge event did not manage to satisfy any of the two inequalities from the expression in (2.3).

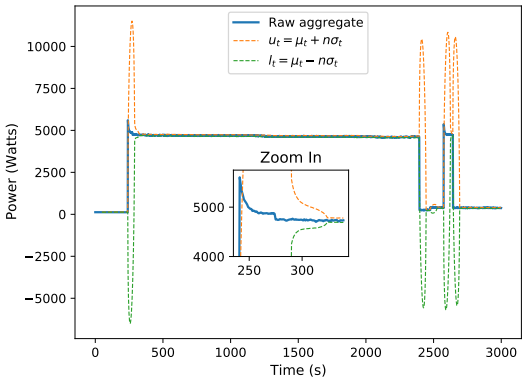

Figure 2.6: Example of a power signal with its respective upper and lower thresholds described in (2.4) and (2.5).

Errors occur when events are not detected due to large variances that may accompany former events. A first approach for solving this problem is to modify the window size w or the number of standard deviations n that we will consider for our threshold.

If there is a sudden change and the window size has a small value, the sample where the event happened will have a large contribution to the moving mean and the moving standard deviation and thus, the threshold. On the other hand, if the window size is chosen to be larger, the sample where the event occurred will have a smaller contribution to the mean and the standard deviation will also be smaller.

The parameter n determines the number of standard deviations the two inequalities from (2.4) and (2.5) will take into consideration, the smaller we choose it to be, the smaller the difference between the signal and the moving mean need to be, in order to satisfy any of the two inequalities. Hence, more changes in the signal will be considered as events. On the other hand, larger values of n will enlarge the absolute values of the thresholds and changes in the aggregate signal will not exceed the thresholds with easiness.

We can observe how the choice of the values for w and n has on detecting changing points in the signal. The number of detected events is a trade-off between these two parameters. However, these parameters are signal dependent, i.e., events in aggregate signals that change abruptly, might not be detected if the window size is large. On the other hand, if the window size is set to a smaller value, there will be changes in the signal that can be labeled as events and those events in the samples could be due to sensor noise (that we aim to remove).

To avoid missing events due to the memory problem that calculating moving statics carries, we calculate the moving mean and moving standard deviation from the aggregate signal $\mathbf{y}$ backwards. Meaning that instead of calculating the moving average and standard deviations starting from $t = 1$, the computation begins from sample $t = N, N-1, N-2, \ldots$ all the way to sample $t = 1$. This can be performed similarly to the expressions in (2.1) and (2.2). The times of detected events are stored in a $K \times 1$ vector denoted $\mathbf{b}$.

Figure 2.7 shows the test case presented in Figure 2.6 with the moving statistics computed backwards and thus, it can be observed how the negative event (fridge turning OFF) is accurately detected.

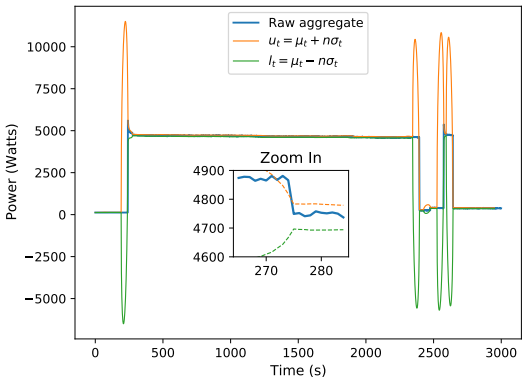

Figure 2.7: Example of an aggregate signal with moving statistics computed backwards (the event that could not be detected in Figure 2.6 is now detected).

Once the associated time indexes for the detected events are stored in $\mathbf{f}$ and $\mathbf{b}$. The vectors can be concatenated and sorted in ascending order and stored in a new vector denoted $\mathbf{e}$. Later, $\mathbf{e}$ is used to find regions between events or change points. Let us suppose that the vector $\mathbf{e} = [1500, 2300]$, the regions in the vector are: $\text{region}_1 = 1$ to 1499, $\text{region}_2 = 1500$ to 2299, and $\text{region}_3 = 2300$ to N. Figure 2.8 depicts an example of the aggregate signal with its respective change points or events.

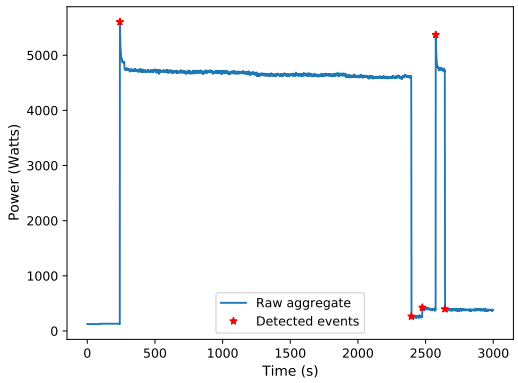

Figure 2.8: Example of an aggregate signal with its corresponding detected events (i.e., change points)

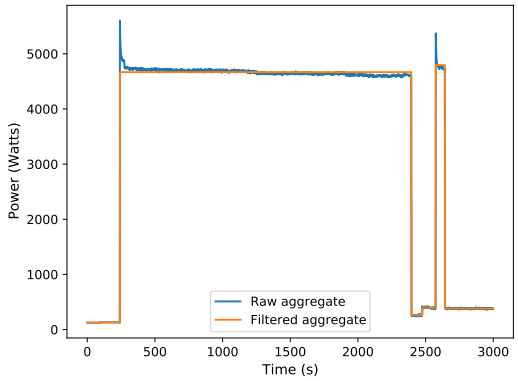

Figure 2.9: Example of a raw aggregate signal and the filtered output.

Finally, we calculate the mean between these regions to obtain our filtered signal. For example, let us suppose that the raw aggregate signal exceeded any of the inequalities in (2.4) or (2.5) at the times $t = 1500$ and $t = 2300$. We would calculate the mean between the samples $t = 1$ to $t = 1499$, $t = 1500$ to $t = 2499$, and $t = 2300$ to $t = N$. The raw aggregate $\mathbf{y}$ with its respective change points is displayed in Figure 2.8. It can be observed that only a few samples are considered to be events. Of course, the number of events detected will depend on the values of w and n. Although these parameters can be chosen heuristically,

later, we provide a discussion of one method to select the appropriate parameters by defining a loss function of the *Root-Mean-Square Error* (RMSE) and the Hoyer sparsity between the raw and filtered aggregates (see Section 2.3 for more details).

The filtered version of $\mathbf{y}$, denoted as $\hat{\mathbf{y}}$ is shown in Figure 2.9. The change point filter was able to remove transients and noise sensor. Moreover, the signal $\hat{\mathbf{y}}$ has sharp edges. All these signal properties will allow methods such as the one presented in [9] to cluster more accurately as the edge values are significantly more consistent across appliance activations.

2.2.3 CUSUM Kalman Filter

Our third noise and transient removal filter consists of fusing the idea of the Kalman filter and the CUSUM algorithm [2].

The Kalman filter, a recursive estimator, is widely employed in numerous applications, ranging from change detection, tracking moving targets, communication systems, guiding and navigation, and biomedical signal processing [41, 42, 43]. Intuitively, the Kalman filter helps us to keep track of the aggregate signal variations. Here the state transition matrix (which for the one-dimensional case is a scalar) is assumed to be one. Thus, the prediction step, more specifically equation (A.7) assumes that the aggregate signal is piece-wise constant. On the other hand, the CUSUM algorithm is in charge of accumulating the error between the previous output of the Kalman filter, denoted $\hat{x}_{t-1}$ and the current the current measurement y_t. Once this accumulation of errors exceed certain threshold h, the CUSUM algorithm "fires" an alarm, indicating that there has been an abrupt change in the signal. Figure 2.10 shows an example of the activations/deactivations detection using the CUSUM Kalman filter. It can be observed that when the absolute value of the accumulation error g_t surpasses the threshold h (or in other words, when $|g_t| > h$), an alarm is fired and we consider that as a change point or event.

Similarly to our previous steady-state filter (see Subsection 2.2.2), we store the time indexes when the CUSUM algorithm detects a change point. Later, we find the regions between these change points. Finally, the mean between these regioans is calculated in order to obtain our filtered signal. Figure 2.11 displays an example of the final result after applying the CUSUM Kalman filter to an aggregate signal composed of a fridge, furnace, and dishwasher signals from the RAE dataset. Contrary to the change point filter presented in 2.2.2, the CUSUM Kalman filter does not need information from the future, it only requires information from the past and the present.

[2]The reader is referred to Appendix A.3, for an extensive understanding of the Kalman filter theory and the CUSUM algorithm

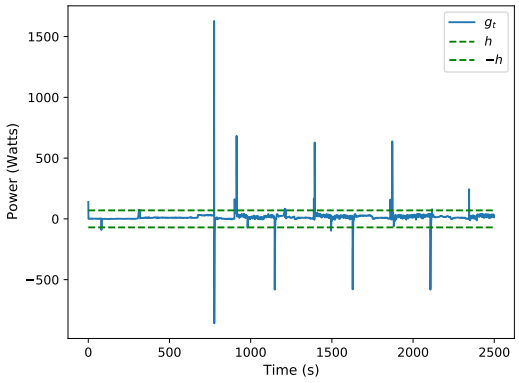

Figure 2.10: Example of change point detection in the CUSUM Kalman filter.

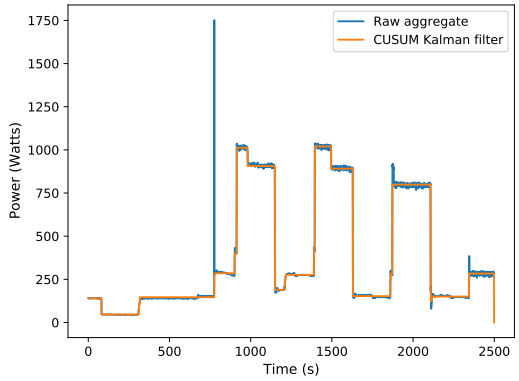

Figure 2.11: Output CUSUM Kalman filtered signal.

2.3 Experimental Setup

The most direct evaluation of the filters presented in this chapter, would be by examining the filtered signal relative to the raw signal. Nevertheless, for a steady-state filter to be performing successfully in NILM, the most reasonable criterion is disaggregation performance using a metric that is directly related to edge clustering such as the methods presented in [1, 9]. However, trying to optimize such types of metrics can be unfeasible due to the large state-space associated with the filter parameters. Jones [9] proposed a more relaxed objective function based on the RMSE and the Hoyer sparsity [44], given by

$$\mathcal{F} = \text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) - \lambda S(\hat{\mathbf{y}}), \tag{2.6}$$

where RMSE$(\cdot)$ is the root mean squared error (4.5) between the aggregate signal $\mathbf{y}$ and the filtered aggregate $\hat{\mathbf{y}}$, λ is the term controlling the importance of the Hoyer sparsity loss S, which is

$$S = \frac{\sqrt{T} - \frac{||\Delta\hat{\mathbf{y}}||_1}{||\Delta\hat{\mathbf{y}}||_2}}{\sqrt{T} - 1}. \tag{2.7}$$

Here, T is the number of samples of the filtered aggregate, $\Delta\hat{\mathbf{y}}$ is the discrete first-differences of the filtered signal, whereas $||\Delta\hat{\mathbf{y}}||_1$ and $||\Delta\hat{\mathbf{y}}||_2$ are their ℓ_1- and ℓ_2-norms, respectively.

Intuitively, we want to achieve the least possible variability between the filtered signal and the aggregate signal, while at the same time, we aim to recreate the signal with least possible edges. This idea can be achieved by minimizing the expression in (2.6). By minimizing the RMSE we will approximate the filtered signal as close as possible to the raw aggregate or in other words, will penalize large deviations. On the other hand, the term $\lambda S(\hat{\mathbf{y}})$ acts as a penalizing term, restricting the filtered signal to have less edges or activations/deactivations. The λ term governs the importance of the sparsity loss. Note that a filtered signal with a large amount of activations/deactivations will make $S(\hat{\mathbf{y}})$ to be close to 1. Oppositely, a filtered signal with fewer activations will result in $S(\hat{\mathbf{y}})$ being close to 0. Note how the range of the function in (2.6) is $[-\lambda, \text{RMSE}]$.

Contrary to [9], we opted for minimizing the loss function in (2.6) using the grid search method for the CP filter and the CUSUM Kalman filter [3]. This is because oppositely to [9], the parameter space for these filters is considerable smaller than in Jones's method. Considering we only have two parameters for each of the steady-state filters, i.e., The CP filter is governed by the parameters w and n, while the CUSUM Kalman filter can be operated using the parameters h and d. We varied the two parameters for each of the filters and evaluated the loss function in (2.6) at each of the parameters. The lower and upper bounds for the parameter searching were chosen by inspection. E.g., we know that for the CP filter, setting n to a small value will allow do detect a large number of activations/deactivations in the aggregate signal. Therefore, if λ is small in (2.6) (meaning that the metric will not penalize if the filtered signal also contains a large number of activations/deactivations), a good minimum of the loss function will be for small n. Oppositely, if λ has a large value (we are restricting the filtered signal to be composed of less activations/deactivations), a good minimum of the loss function will be for a large n.

[3] Although the Kalman filter depends on designing parameters such as the process noise covariance $\mathbf{Q}_t$, the measurement noise $\mathbf{R}_t$ (see Appendix A.1), there is no evidence in order to choose the parameters heuristically. Therefore, for this work we decided to set $\mathbf{Q} = 1 \times 10^{-4}$ and $\mathbf{R}_t = 1 \times 10^{-4}$ and decided to optimize the designing parameters for the CUSUM algorithm h and d. Although it could be claimed that the parameters $\mathbf{Q}_t$ and $\mathbf{R}_t$ are important for the Kalman filter, there is no intuitive sense of how to select these values. Optimizing these extra parameters for the CUSUM Kalman filter might lead up to more accurate results (see 2.4) for denoising signals. This is left to future work.

We performed the parameter optimization of the CP, CUSUM Kalman and steady-state block filters [2] for the whole RAE dataset which is sampled at 1 Hz. The optimization was performed on the "mains" submeter of both house 1 and 2 (which contains the aggregate signals). House 1, block 1 contains $\sim$ 9 days worth of data, whereas House 1, block 2 and House 2, block 1 contain $\sim$ 63 days per block.

2.4 Experimental Results

This sections presents a discussion of the results obtained from our two filtering approaches: the CP filter and the CUSUM Kalman filter. Further, the two filters are compared against a state-of-the-art filter presented in [2].

Figure 2.12 depicts the aggregate signal with its respective filtered signals: the CP filter, the CUSUM Kalman filter, and the steady-state block filter [2]. Figures 2.13 and 2.14 present a more detailed examination of Figure 2.12. Figure 2.13 shows the filters acting on a clothes dryer activation, it can be observed that the three filters were able to detect the activation. However, the Kalman filter detected an extra deactivation ($\sim$ 320) seconds, whereas the steady-state block filter [9] and our proposed CP filter have a similar behaviour. Later, around the sample 415, the furnace was activated, both the CP filter and the Kalman filter were able to detect the change. Finally, Figure 2.14 depicts the cycle of the fridge. Here, the three filters were able to detect the activation and deactivation, correctly.

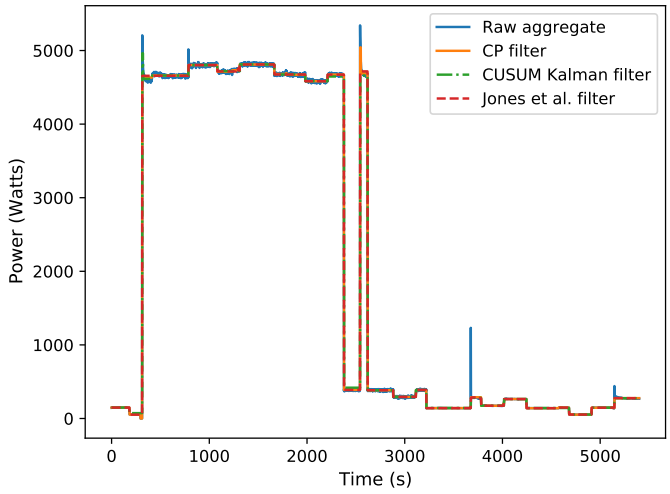

Figure 2.12: Visual comparison between our two steady-state filters and the steady-state block filter [2].

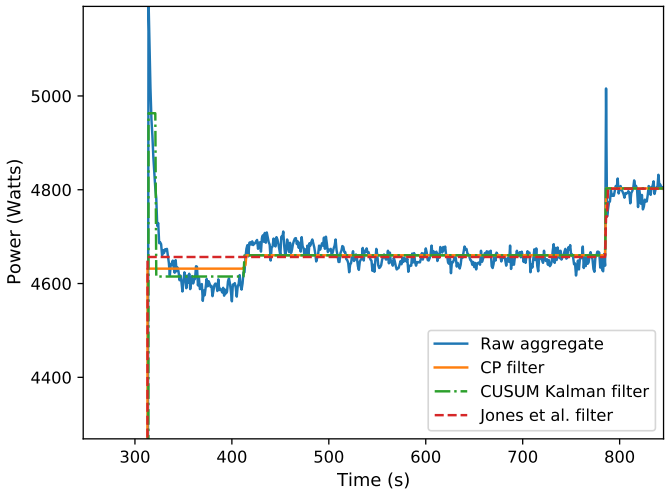

Figure 2.13: Zoom to Figure 2.12. Filters acting on a clothes dryer activation ($\sim$ 310 seconds), followed by the furnace activation ($\sim$ 415 seconds).

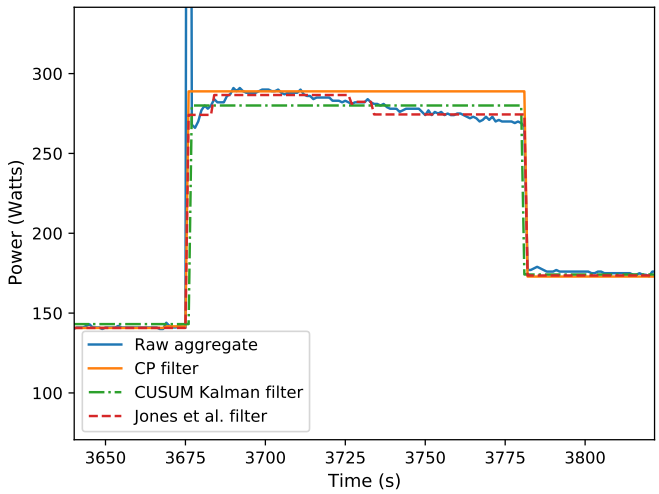

Figure 2.14: Zoom to Figure 2.12. Filters acting on a fridge activation ($\sim$ 310 seconds).

Figure 2.15 shows an example of the loss function surface in (2.6), with $\lambda = 1000$. The error surface was computed using the CP filter over the mains of house 1, block 1 of the RAE dataset ($\sim$ 9 days).

Tables 2.1, 2.2, and 2.3 summarize the CP filter parameters after minimizing the metric in expression (2.6), the parameters were computed with different λ weights on the Hoyer loss. It can be observed that the larger the λ weight, the more "abrupt" the filter acts on the aggregate signal. Also, the larger the λ weight, the larger the parameters n and w

resulted. This suggests that if we aim to have less activations/deactivations in our filtered aggregate signal, the values of w and n should be enlarged. A visual comparison of filtered aggregate signals using the optimum parameters for the CP filter in the Table 2.1 can be observed in Figure 2.16.

Figure 2.17 displays a visual comparison for the CUSUM Kalman filter local optimum parameters (in terms of the RMSE and Hoyer sparsity function) shown in Table 2.4. As expected, larger λ weights in the Hoyer sparsity impose a stronger or more abrupt filtering in the aggregate. Therefore, the filtered signal will have less activations/deactivations which can be beneficial in some scenarios. E.g., appliances like the clothes dryer or heat pump which have large power values and remain ON for large time intervals. However, on cases where the appliances change their power draw abruptly (e.g., the dish washer), a filter optimized with large λ weights will not be able to keep track of these abrupt changes in the signal. Tables 2.5 and 2.6 summarizes the CUSUM Kalman filter parameters after performing the minimization for the equation in (2.6). We can observe how increasing λ weights suggest the enlargement of the d threshold. This is not surprising as only large activations would exceed d.

Tables 2.7, 2.8, and 2.9 show the resulting "optimal" parameters for the loss function in (2.7). The optimization was performed using the `dual_annealing` library from the `scipy.optimize` package (see [9] for more details).

Table 2.10 shows a comparison of the run-times for our two proposed approaches (the CP and the CUSUM Kalman filter versus the approach in [2]). The run-times were averaged over 100 trials. It can be observed how the CP filter and the CUSUM Kalman filter outperformed the steady-state block filter in terms of run-times for the three houses in RAE. Although the "mains" from RAE house 1, block 2 and house 2, block 1 are composed of the same amount of data ($\sim$ 63 days), it can be seen how the run-times for the three filters are lower for house 2, block 1. This could be due to the varying complexity of aggregate signals from both houses. Moreover, it can be observed that for house 1, block 2, the steady-state block filter [9] runs at least $\sim 7\times$ slower than ther CP and the CUSUM Kalman filters. Which suggests that the greater the complexity in the aggregate signal from the house, the steady-state block filter [2] faces with more challenge (in terms of running time) in order to operate.

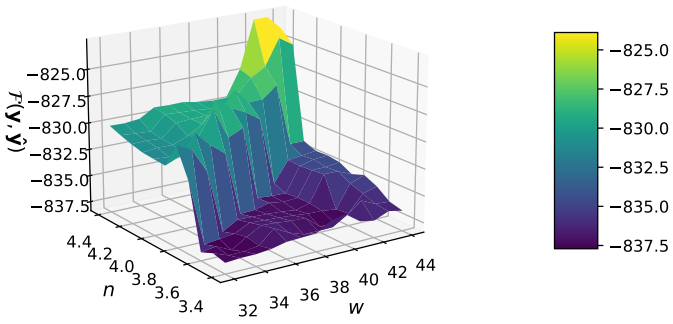

Figure 2.15: Surface of loss function in (2.6) for the CP filter. House 1, block 1 ($\sim$ 9 days).

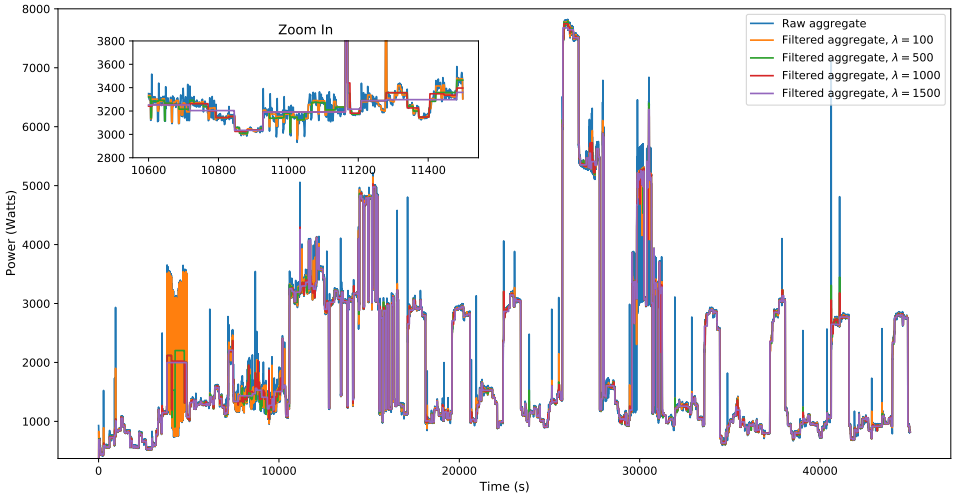

Figure 2.16: Example of various "optimized" filtered outputs (CP filter), for increasing λ weights. Mains from House 1, block 1 ($\sim$ 9 days).

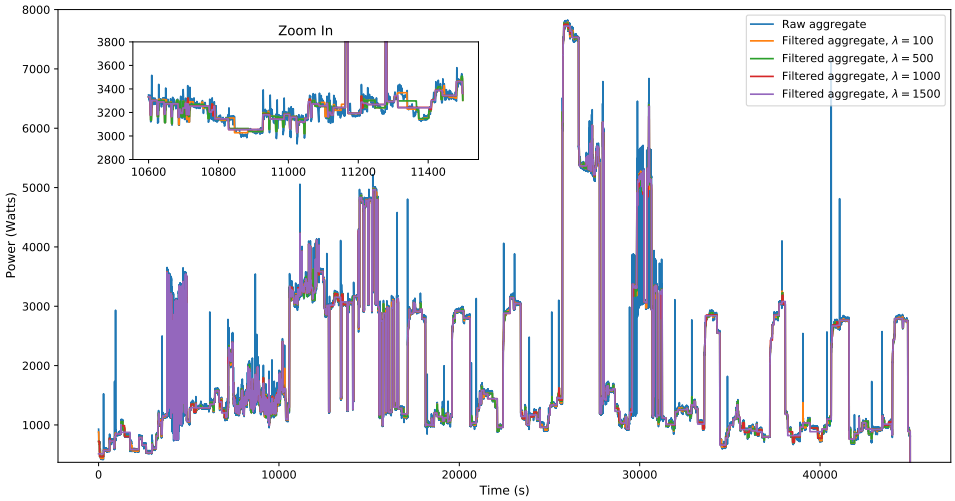

Figure 2.17: Example of various "optimized" filtered outputs (CUSUM Kalman filter), for increasing λ weights. Mains from House 1, block 1 ($\sim$ 9 days).

Table 2.1: CP filter: Optimal parameters for the loss function in (2.6). RAE House 1, block 1 ($\sim$ 9 days).

λ	w	n	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	5	1.6	-27.4
500	18	2.5	-368.5
1000	28	2.9	-840.7
1500	39	4	-1315.3

Table 2.2: CP filter: Optimal parameters for the loss function in (2.6). RAE House 1, block 2 ($\sim$ 63 days).

λ	w	n	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	5	1.6	-34.4
500	18	2.5	-384.2
1000	26	3.6	-854.3
1500	30	3.9	-1332.3

Table 2.3: CP filter: Optimal parameters for the loss function in (2.6). RAE House 2, block 1 ($\sim$ 63 days).

λ	w	n	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	5	1.7	-64.2
500	18	2.9	-441.0
1000	26	3.7	-916.6
1500	30	4	-1406.1

Table 2.4: CUSUM Kalman filter: Optimal parameters for the loss function in (2.6). RAE House 1, block 1 ($\sim$ 9 days).

λ	h	d	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	64	0	9.7
500	74	2	-368.0
1000	111	3.5	-845.5
1500	124	4.5	-1320.6

Table 2.5: CUSUM Kalman filter: Optimal parameters for the loss function in (2.6). RAE House 1, block 2 ($\sim$ 63 days).

λ	h	d	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	64	0.5	0.3
500	74	1.5	-378.9
1000	111	4	-858.0
1500	120	4	-1333.4

Table 2.6: CUSUM Kalman filter: Optimal parameters for the loss function in (2.6). RAE House 2, block 1 ($\sim$ 63 days).

λ	h	d	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	58	0	-31.5
500	66	1.5	-416.5
1000	85	5	-899.0
1500	174	2	-1381.6

Table 2.7: Steady-state block filter [2] in (2.6). RAE House 1, block 1 ($\sim$ 9 days).

λ	τ_{base}	m_d	s_d	m_e	s_a	f_p	f_h	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	21.7	152.0	19.7	2164.0	59.3	0.4	0.8	-18.4
500	30.9	116.5	19.2	4957.8	75.3	0.8	0.9	-394.7
1000	48.9	217.2	34.1	4792.0	92.3	0.9	0.3	-865.5
1500	47.3	105.1	20.7	4880.7	22.5	0.4	0.8	-1340.7

Table 2.8: Steady-state block filter [2] in (2.6). RAE House 1, block 2 ($\sim$ 63 days).

λ	τ_{base}	m_d	s_d	m_e	s_a	f_p	f_h	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	15.0	142.4	16.1	4083.0	32.2	0.1	0.05	-6.8
500	13.5	183.0	14.5	3528.5	18.0	0.2	0.0	-372.6
1000	15.0	211.7	21.7	1200.0	18.0	0.2	0.0	-832.2
1500	15.0	193.0	22.9	1200.0	18.0	0.1	0.0	-1292.9

Table 2.9: Steady-state block filter [2] in (2.6). RAE House 2, block 1 ($\sim$ 63 days).

λ	τ_{base}	m_d	s_d	m_e	s_a	f_p	f_h	$\min \mathcal{F}(\mathbf{y}, \hat{\mathbf{y}})$
100	32.7	167.7	38.2	4543.8	25.0	0.6	0.8	-68.4
500	36.1	159.7	34.3	4649.1	53.3	0.8	0.9	-453.1
1000	31.6	193.0	28.8	1965.9	37.8	0.4	0.9	-934.6
1500	41.6	170.6	24.2	4221.1	11.0	0.9	0.9	-1416.5

2.5 Discussions

In general, the CP filter is a good alternative to filtering method proposed by Jones et al. [2]. It achieves results as good as the steady-state block filter (in terms of the loss function defined in equation (2.6)). Moreover, it only depends on two parameters which allow optimization tasks to be easier and faster. Furthermore, the computational time of the CP filter is proven to be $\sim$ 8 times faster than the steady-state block filter. On the other hand, the CUSUM Kalman filter output depends only on past and present inputs. Therefore, it is a good alternative as it offers similar run-times to the ones achieved by the CP filter and can be used in systems that require processing in real-time. However, it does not offer low loss values as the ones provided by the CP and steady-state block filters.

Table 2.10: Experimental run-times: RAE dataset

RAE-House 1, Block 1 (~ 9 days)	
Filtering Method	**Run-time (seconds)**
CP filter	5.2826
CUSUM Kalman filter	4.0520
Steady-state block filter [2]	12.8397
RAE-House 1, Block 2 (~ 63 days)	
CP filter	30.0079
CUSUM Kalman filter	28.6012
Steady-state block filter [2]	222.5080
RAE-House 2, Block 1 (~ 63 days)	
CP filter	16.20378
CUSUM Kalman filter	25.39075
Steady-state block filter [2]	53.94789

Chapter 3

NILM - Unsupervised Learning

This chapter focuses on unsupervised learning methods for NILM. Section 3.1 provides a summary of the state-of-the-art unsupervised algorithms applied to the appliance disaggregation field. Later, we describe our proposed unsupervised method for decomposing aggregate power signals into a series of uni-modal set of appliances.

3.1 Related Methods

The following section is a summary of the related NILM unsupervised methods. Some of the methods described here, were motivating work for our proposed method described in Section 3.2. Further, our proposed method based on a hybrid KP-GMM was a stepping stone for the work presented by Jones in [9].

Jones in [9] proposed an unsupervised clustering algorithm based on non-parametric GMMs. The algorithm grouped appliances' activations/deactivations (events). Later, the author developed an efficient method to pair each activation with its corresponding deactivation in order to decompose the aggregate as uni-modal appliances. The algorithm does not require prior information of the number of clusters. It is acknowledged that the work developed by Jones is leveraged throughout this thesis.

The work in [45] describes how NILM can be solved using the KP and a genetic algorithm. Despite the idea seems simple and promising, it has some drawbacks: genetic algorithms suffer from a long execution time [46] and it is impossible to guarantee that they find the solution to the problem we are trying to solve [43]. Another disadvantage of using the approach proposed by Egarter et al. is mainly the assumption of the aggregate signal being perfectly rectangular-shaped and this is far from reality in electrical signals.

In [47], the authors presented a source separation NILM approach based on *Non-negative Matrix Factorization* (NMF). First, the method performs NMF so that the most important features can be extracted. Finally, the disaggregation is performed via factorizing the associated matrix, where the models previously learned need to be consider. The authors

used Source Separation Via Tensor Factorization ([sic] STMF), which is a technique that exploits further information regarding dependencies among the appliances.

In [48], the authors presented a NILM approach based on source separation. The proposal is to, instead of supervision, make use of input features for each source signal. Later, with the aid of convex optimization methods, the authors find estimators of the correlations between the input features and the unobserved signal decomposition. The results demonstrate that in order to perform disaggregation accurately, the features for different signals should be linearly independent.

The authors in [49] presented an blind source separation (BSS) approach. First, clusters of steady-state changes are created using a genetic K-means algorithm. Later, the original power signals are recreated by means of a matching pursuit (MP) algorithm which uses the clusters that were previously found as the sources.

The authors in [50] introduced an unsupervised method for low sampling frequency power measurements. The work presented four variations of HMMs: Factorial HMM (FHMM), Conditional FHMM (CFHMM), factorial hidden semi-Markov model (FHSMM), and conditional factorial hidden semi-Markov model (CFSHMM). The authors concluded the best variation was CFSHMM. The appliances were modelled as two state (ON/OFF) appliances. The aggregate signal was composed of appliances such as TV, game consoles, monitor, home theater, fridge, and laptop. The authors rely on features such as daily schedule of the occupants which could have negative effects due to the random behaviour of humans.

The author in [51] presented an unsupervised NILM approach. The method was magnitude-based and it was built using a standard HMM. The work novelty revolves around the application of "segmented" version of the Viterbi algorithm.

In [52], an"unsupervised" approach is presented. The method does not require to collect training data for the individual appliances. However, the approach uses a "one-off" supervised method which requires priors. Moreover, the method requires an unsupervised learning method over unlabelled aggregate data that later on will be tuned for an specific household. More specifically, the approach learns a general appliance behavioural model which then will generalise to previously unseen households, also known as transferability [53].

The authors in [54] presented an unsupervised NILM algorithm based on FHMM and particle filtering (PF). PF is a technique suitable for tackling nonlinear and non-Gaussian problems. Due to the nonlinear behaviour of some appliances and non-Gaussian nature of the NILM noise [23], PF can be used for appliance state estimation. Although methods such as the Viterbi algorithm are commonly used to compute inference, it no longer can be used for performing the same task in FHMM.

In [55], the authors presented an unsupervised algorithm based on dynamic time warping (DTW) for NILM. The approach uses low-resolution time data from the smart meter

(greater than 1 Hz). Although the approach is unsupervised, it requires a training phase (over the aggregate data). Moreover, these training periods are necessary in order to build a library of appliance signatures to finally, perform pattern matching.

The authors in [12] presented an unsupervised method for load disaggregation based on Non-parametric FHMM. the approach models the appliances as only two state (ON/OFF) Gaussian appliances. Furthermore, the authors developed an inference algorithm, the algorithm is able to detect the number of loads from the data and simultaneously, separate the aggregate power signal.

An unsupervised NILM technique based on GSP was presented in [10]. The approach does not need any training before NILM and is based on edge detection. once the edges are detected, the algorithm pairs positive edges with negatives edges exploiting magnitude and time interval information.

3.2 Hybrid Knapsack-GMM(KP-GMM)

With the aggregate signal filtered, appliance states are now far less challenging to be modelled with Gaussian distributions. Moreover, Egarter et al. [45] showed how the KP can be mapped to the NILM problem. Figure 3.1 depicts this idea. Intuitively, at each time step t we have a knapsack with a maximum capacity (the aggregate power) and a set of items that we can choose from in order to fill-up the knapsack (the possible appliances that could contribute the best to form that aggregate).

Despite the mapping of the KP to the NILM problem seems straightforward, in reality, we do not have the perfectly rectangular-shaped signals as the ones depicted in Figure 3.1. The steady-state filters presented in Subsections 2.2.2 and 2.2.3 can provide us with aggregate signals with no transients and sharp edges. However, the variability in the power signals, even after applying the filtering step, can still be significant. Therefore, we will not use the KP to formulate the NILM problem as Egarter et al. [45] did, instead, we leverage its use for solving a special case that arises when two or more appliances turn ON/OFF simultaneously.

3.2.1 Modelling the Appliance States

Trying to cluster the differences in the aggregate signal without filtering would be very challenging as the power values would highly overlap, therefore, leading to miss-classification in the appliance states. Nevertheless, after filtering the raw aggregate signal with the aid of a steady-state filter, the differences from the filtered signal are notably more uniform over appliance activations/deactivations.

The general idea behind the hybrid knapsack-GMM is to model the first-differences or edges as Gaussian distributions, perform clustering over the edges and update their means and covariances using the *Expectation-Maximization* (EM) algorithm. With the aid of the

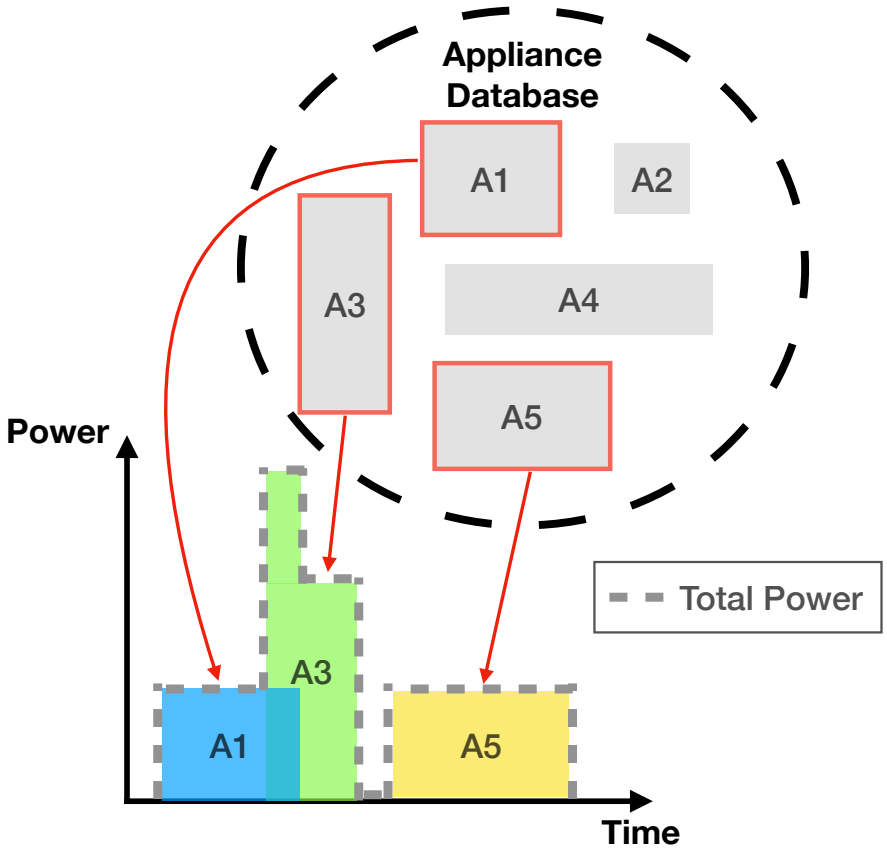

Figure 3.1: Basic principle of the KP problem mapped to NILM [45].

KP algorithm we are able to distinguish when multiple appliances turn ON or OFF at the same time. All of the process is performed in real-time. Moreover, in order to reconstruct the signals after performing clustering, we pair each activation or positive edge with its corresponding deactivation.

The first-differences $\Delta\hat{y} = \hat{y}_t - \hat{y}_{t-1}$ are calculated from the filtered signal $\hat{\mathbf{y}}$. All differences whose absolute value exceeds ± 50 Watts [10] are considered events for this work. In simple words, we model these events or changes in the aggregate signal as uni-modal Gaussian distributions (i.e., the appliances' states can only have two modes of operation: ON or OFF) using EM algorithm for GMM (see Appendix for more details). Moreover for each Gaussian component, the average transient is also calculated, the reason to do so is to be able differentiate appliance modes with similar power values, as we will address later throughout this work.

Later, we attempt to pair each of these activations with their corresponding deactivations to reconstruct the disaggregated signals. The number of Gaussians proportionally grows as the number of distinct events is discovered. E.g., if an appliance state with an activation of 150 Watts is discovered, this will be modelled with a Gaussian distribution of $\mathcal{N}(\mu_1 = 150, \sigma_1 = 0.05 \cdot \mu_1)$. We decided to initialize the standard deviation to be the 5% of the mean

as we thought it was a good starting point based on evidence from house 1, block 1 from the RAE dataset. Later, if another ON event is discovered with an activation value of 2050 Watts, the appliance state will be modelled with a Gaussian denoted $\mathcal{N}(\mu_2 = 2050, \sigma_2 = 0.05 \cdot \mu_2)$. However, if a new activation of 155 Watts is discovered, this activation will belong to the appliance state previously modeled as the $\mathcal{N}(\mu = 150, \sigma = 0.05 \cdot \mu)$ Gaussian distribution. The Gaussians' means and standard deviations are updated based on the EM described in algorithm 3 [1].

For an event to be modelled with a Gaussian distribution, it has to pass a series of requirements. However, the most important condition for the new event to meet is that it has to be more than 3 standard deviations away from the mean (i.e., the Mahalanobis distance [56, 57] has to be more than 3) of each of the previously instantiated Gaussians. The Mahalanobis distance for a one-dimensional distribution is given by:

$$D_M(\Delta\hat{y}) = \sqrt{\frac{(\Delta\hat{y} - \mu_k)^2}{\sigma_k^2}}, \tag{3.1}$$

where, μ_k and σ_k^2 are the mean and the variance of the k-th Gaussian or appliance state, respectively.

In order to reconstruct the disaggregated signals based on the events modelled as Gaussian distributions, we need to carry out a pairing of ON events with their respective OFF events. In principle, we try to pair each ON event with its next OFF event. If the OFF event is less than 3 standard deviations away from the mean of the possible appliance states that were ON, then the events are paired and the appliance state absolute value mean is recalculated between the ON and OFF event's times.

Makonin [17] identified that most of the researchers assume the *Switch Continuity Principle* (SCP), which states that only one appliance ever changes its state at any given point in time. However, the findings revealed that the SCP may not hold true.

In order to take into consideration when a set of appliances change their states at the same time, we make used a modification of the KP problem. I.e., if a new ON event is discovered and it can be considered a new appliance state, we first verify if there were two or more previously observed and OFF appliances that could fit in this "knapsack". Moreover, if there is a OFF event that we have not observed, we verify if it was not a combination of two or more appliance states that were ON. E.g., if an a new event of +250 Watts is discovered, and there were two appliances previously discovered (at the time t of the new incoming event the two appliances have to be in OFF operational mode) and modelled as Gaussians. The knapsack capacity (see Appendix A.5) will be $c_t = 250$, here c_t denotes the knapsack capacity at the t time instant. Let us say the first appliance is modelled as $\mathcal{N}(\mu_1 = 150, \sigma_1 = 0.05 \cdot \mu_1)$ and the second appliance is modelled as $\mathcal{N}(\mu_2 = 100, \sigma_2 = 0.05 \cdot \mu_2)$, then these

[1] The reader is referred to Appendix A.4 for a more complete explanation of GMMs and the EM algorithm

two Gaussians will be considered two "items" that could be packed in the knapsack. Each appliance or Gaussian has a set of "weights" and "profits". Let us explore the particular case for the Gaussian $\mathcal{N}(\mu_1 = 150, \sigma_1 = 0.05 \cdot \mu_1)$ will have a set of weights $w_1 = \{\mu_1 - |i|\}$, where $i = \texttt{round}(-\sigma_1), \ldots, 0, \ldots . \texttt{round}(\sigma_1)$. Further, the set of profits per appliance was created so that the maximum profit was equal to 1. Thus, for this particular example $p_1 = \{\frac{\mu_1 - |i|}{\mu_1}\}$. A similar approach can be taken for creating the weights and the profits for as many appliances as there appear in the system. E.g., the Gaussian $\mathcal{N}(\mu_1 = 150, \sigma_1 = 0.05 \cdot \mu_1)$ will have a set of weights $w_1 = \{\mu_1 - \sigma_1, \ldots, \mu_1 - 1, \mu_1, \mu_1 + 1, \ldots, \mu_1 + \sigma_1\}$ by restricting the values in the sets of weights to only take integer values. Further, the set of profits per appliance was created so that the maximum profit was equal to 1. Thus for this particular example, $p_1 = \{\frac{\mu_1 - |\sigma_1|}{\mu_1}, \ldots, \frac{\mu_1 - |-1|}{\mu_1}, 1, \frac{\mu_1 - |+1|}{\mu_1}, \ldots, \frac{\mu_1 - |\sigma_1|}{\mu_1}\}$. A similar approach can be taken for as many appliances as there appear in the system [2].

Finally, even though the steady-state filter greatly alleviates the problem of the ON/OFF events not being consistent, it is inevitable to not having overlapping power values between appliance states. For every ON event, the difference between the first 200 samples from the raw aggregate signal and the filtered signal is computed, and it is stored as an average. We selected this number of samples as we thought the transient behaviour of an appliance can be captured within this time span. Having access to transient information helps to differentiate between appliance states when the activations have similar means. Therefore, we calculate the similarity between the transient associated to a new activation denoted τ_Δ ,to the running average transient associated to each of the Gaussian components denoted $\bar{\tau}_k$. We perform the computation of the transients similarity measure with the `matchTemplate` function contained in the `OpenCV` library, which calculates the normalized cross-correlation [58] between the the two transients [59]. When the transients are "perfectly similar" the value retrieved by the `matchTemplate` function will be 1. On the other hand a value close to 0 indicates the transients are distinct. For our work, we chose a value of 0.9 to include an incoming event to be part of an already instantiated Gaussian distribution. The value was selected based on the calculation between several transients of the same appliance. Figure 3.2 shows three different transients for the fridge, furnace, clothes dryer and the heat pump for the house 1, block 1 of the RAE dataset. The algorithm 1 summarises the hybrid KP-GMM.

3.3 Experimental Setup

We take data from House-1, Block-1, (comprised of nine day's worth of data) of the RAE dataset [8] which is sampled at 1Hz. Our disaggregator was coded in Python 3.6. We chose

[2]The modification to the KP algorithm described here can be found at: https://github.com/compsust/FilterDesign/tree/master/APPEEC

Algorithm 1 The Hybrid KP-GMM Algorithm for appliance disaggregation

Assuming the raw aggregate $\mathbf{y}$ has been pre-processed using a steady-state filter from . The resulting filtered signal denoted $\hat{\mathbf{y}}$

$K = 0$, where K is the number of discovered appliances/clusters
for $t = 2, \ldots, T$ **do**
    $\Delta\hat{y} = \hat{y}_t - \hat{y}_t - 1$
    if $\Delta\hat{y} > 50$ **then**
        if First event **then**
            $\mathcal{N}(\mu_1 = \Delta\hat{y}, \sigma_1 = 0.05 \cdot \mu_1)$
            $\mathcal{N}(\mu_1 = \Delta\hat{y}, \sigma_1 = 0.05 \cdot \mu_1)$.mode()=ON
            Store transient τ_1 (200 samples)
            $K = K + 1$
        else if $\left[\text{argmin}_k(D_M(\Delta\hat{y}) \leq 3)\ \&\&\ \mathcal{N}(\mu_k, \sigma_k)\text{.mode()==OFF}\ \&\&\ \texttt{matchTemplate}(\tau_\Delta, \tau_k) \geq 0.9\right]\ \forall k \in K$ **then**
            Compute EM Algorithm
            $\mathcal{N}(\mu_k, \sigma_k)$.mode()=ON
            Recalculate $\bar{\tau}_k$ using τ_Δ
        else if KP($\Delta\hat{y}$,$\mathcal{N}(\mu_k, \sigma_k)$.mode() == OFF) $\geq 90\ \forall k \in K$ **then**
            $\mathcal{N}(\mu_k, \sigma_k)$.mode()=ON
        else
            $K = K + 1$
            $\mathcal{N}(\mu_K = \Delta\hat{y}, \sigma_K = 0.05 \cdot \mu_K)$
            Compute EM Algorithm
            $\mathcal{N}(\mu_K, \sigma_K)$.mode()=ON
            Store transient $\bar{\tau}_K$
        end if
    end if
    if $\Delta y < -50$ **then**
        if KP($|\Delta\hat{y}|$,$\mathcal{N}(\mu_k, \sigma_k)$.mode() == ON) $\geq 90\ \forall k \in K$ **then**
            $\mathcal{N}(\mu_k, \sigma_k)$.mode()=OFF
            Compute EM Algorithm adding $|\Delta\hat{y}|$
        else if $\text{argmin}_k(D_M(|\Delta\hat{y}|) \leq 3)\ \forall k \in K$ **then**
            Compute EM algorithm
            $\mathcal{N}(\mu_k, \sigma_k)$.mode()=OFF
        end if
    end if
end for

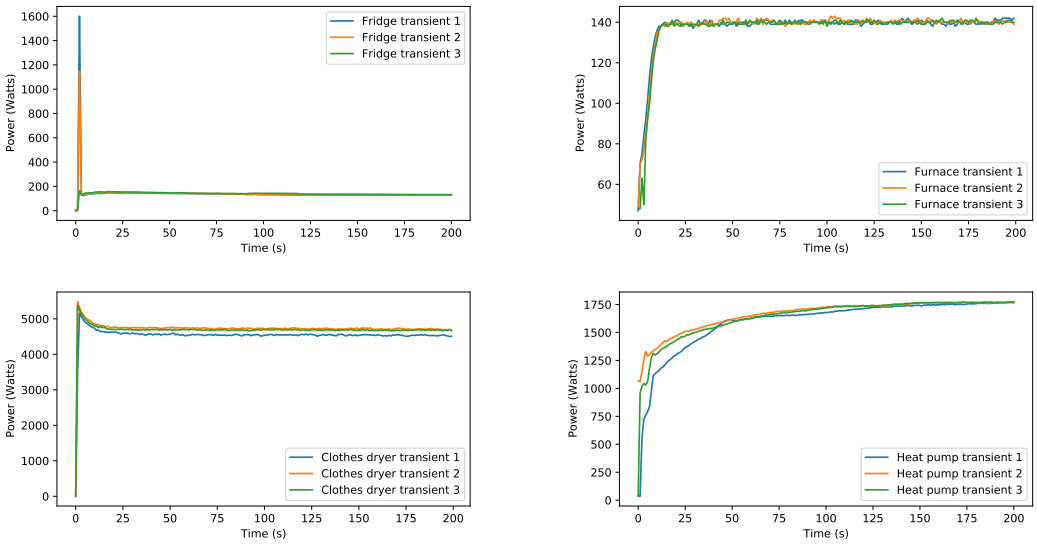

Figure 3.2: Examples of different transients captured for the fridge, furnace, clothes dryer and the heat pump. These transient information helps to identify appliances with similar power values.

Python because of its array manipulation capabilities and digital signal processing libraries; moreover, its convenience when coding rapid prototypes that can get quickly translated to faster coding languages (e.g., C). All tests ran on a Mac Pro (2017 model) with a 2.3GHz Intel Core i5 processor and 8GB of memory. Furthermore, we run all our tests in one day's worth of data without any prior knowledge.

3.4 Experimental Results

Figures 3.3 and 3.4 show the individual appliances, ground truth and disaggregated respectively. We can observe that the disaggregator detected four appliances (one more appliance than in the ground truth). Nevertheless, with our proposed "power assignation metric" from Section 4.2 or with a deep learning method such as the one presented in [9], this two unlabelled appliances, can be merged together.

Further, the output from the disaggregator (see Figure 3.5), ignored the clothes dryer from sample 2620 to sample 3220. It was not able to track the approximately 200 Watts operational state the clothes dryer was running on. Therefore, the disaggregator ignored this OFF event. Although the system was not able to track this part of the signal, the power consumption from the clothes dryer at that interval was very small and did not considerably affect the overall accuracy score.

To determine energy tracked, we integrate our 1 Hz power (in Watts) samples to energy measured in kWh, as such:

$$energy = \int_0^T \frac{y_t \times \frac{1}{3600}}{1000}\, dt. \tag{3.2}$$

Figure 3.5 depicts both how close the total energy tracked was as compared to the raw ground truth aggregated signal. The system was able to track from 93.7% to 97.1% of the total aggregate energy without the use of prior information (see Table 3.1). This accuracy measure is similar *Normalized Disaggregation Error* (NDE) [5, 60]. We define the accuracy measure as:

$$accuracy = \frac{energy(\hat{\mathbf{y}})}{energy(\mathbf{y})} \times 100. \tag{3.3}$$

Using the same nomenclature defined in previous sections. Some of the contributing factored to a lower accuracy for the fridge (87.3%) was due to the fact that the ON-spikes from the compressor were filtered out by the filter pipeline method. However, the CP filter and CUSUM Kalman filter are able to remove transient information and at the same time preserving the energy from the raw aggregate signal. In reality the energy contributions from these ON-spikes is negligible, but for the small sample period in this case. Lower accuracy for the furnace may have been caused by the fact that it has a constantly-ON power reading of 44 Watts.

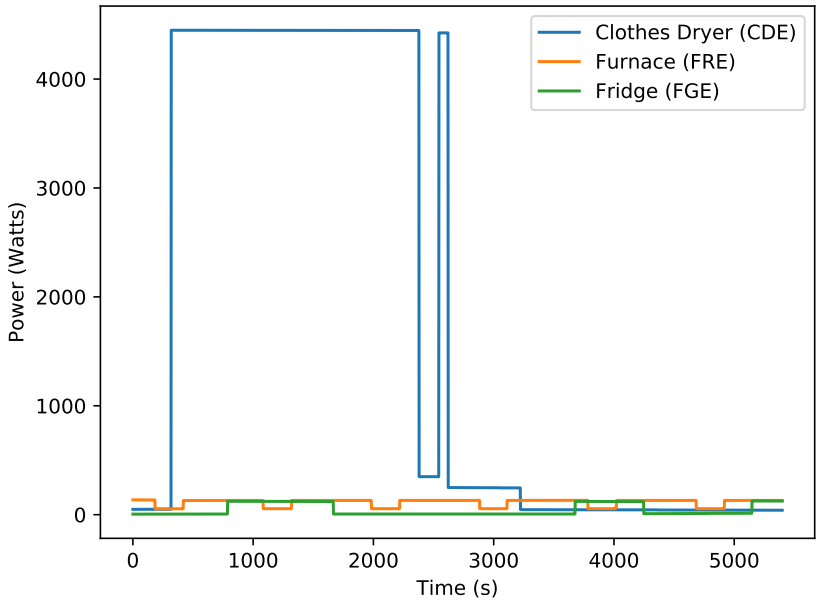

Figure 3.3: Ground truth sub-metered appliance signals.

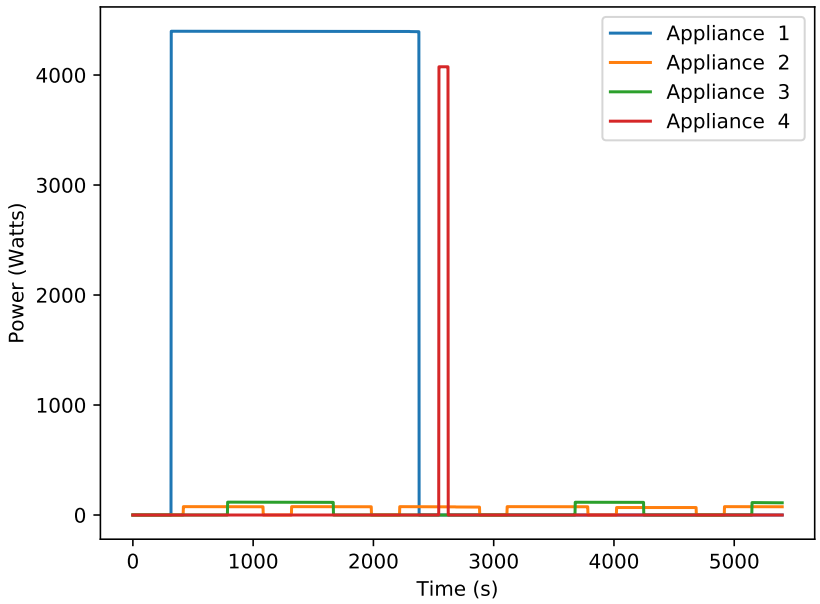

Figure 3.4: Hybrid KP-GMM algorithm disaggregated appliances.

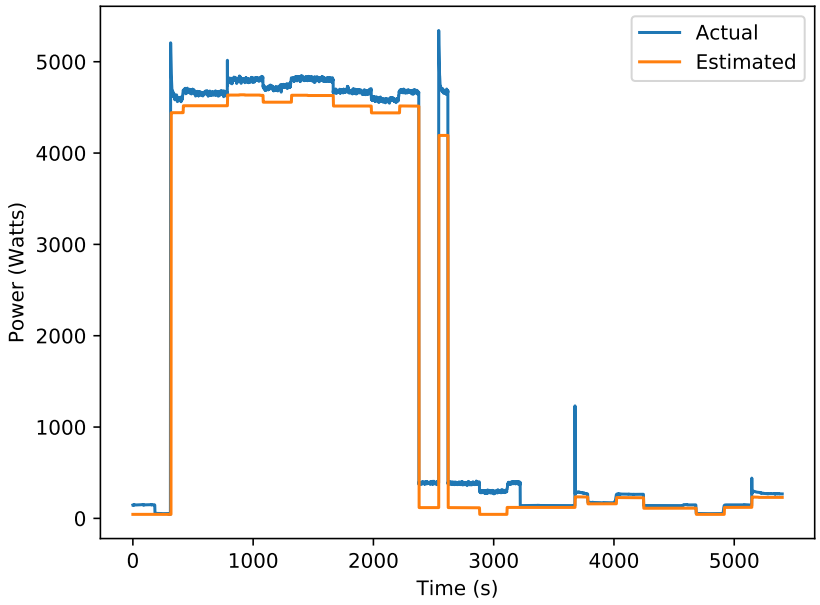

Figure 3.5: Comparing the ground truth aggregate to the disaggregated aggregate showing 93.7% accuracy in tracking.

Table 3.1: Comparison of Energy Truth/Filtered/Tracked (in kWh)

Filter pipeline				
Appliance	**G.Truth**	**Filtered**	**Est/Tracked**	**Truth *vs* Est**
Clothes Dryer	2.753	2.752	2.604	94.5%
Fridge	0.063	0.062	0.055	87.3%
Furnace	0.174	0.149	0.144	82.8%
Aggregate	2.990	2.961	2.803	93.7%
CP filter				
Appliance	**G.Truth**	**Filtered**	**Est/Tracked**	**Truth *vs* Est**
Clothes Dryer	2.753	2.752	2.698	98.0%
Fridge	0.063	0.063	0.061	96.8%
Furnace	0.174	0.174	0.163	93.6%
Aggregate	2.990	2.989	2.922	97.7%
CUSUM Kalman filter				
Appliance	**G.Truth**	**Filtered**	**Est/Tracked**	**Truth *vs* Est**
Clothes Dryer	2.753	2.752	2.695	97.8%
Fridge	0.063	0.063	0.062	98.4%
Furnace	0.174	0.173	0.149	84.8%
Aggregate	2.990	2.988	2.906	97.1%
Steady-state block filter [2]				
Appliance	**G.Truth**	**Filtered**	**Est/Tracked**	**Truth *vs* Est**
Clothes Dryer	2.753	2.751	2.698	98.0%
Fridge	0.063	0.063	0.061	96.8%
Furnace	0.174	0.174	0.165	94.8%
Aggregate	2.990	2.988	2.924	97.8%

3.5 Discussions

Our disaggregation method can decompose aggregate power signals that are conformed of appliances that periodically turn ON/OFF. E.g., fridge, clothes dryer, heat pump (when it has a periodical behaviour). However, it suffers from some drawbacks such as mis-classifying appliances. Some of the mis-assignations are explained as follows: In an aggregate signal, there could be an ON event of $\Delta\hat{y}$ Watts and if we previously have two appliances modeled as Gaussians with similar means μ_1 and μ_2 (both appliances OFF at the time we detected the $\Delta\hat{y}$) event. Because both appliances could "fit" in the knapsack, then they will both get

turned ON and incorrectly classify the appliances. Also, the fact that there is no transient information for OFF events, could lead to poor disaggregation performance for appliances with similar power values (e.g., fridge and the furnace), as our proposed method will mix the pairing for such appliances. Another important case to consider when attempting to pair ON/OFF events is the fact that an appliance turning ON with a specific power value, let us say $\Delta\hat{y}$, does not necessarily translate to the appliance turning OFF with that negative power value ($-\Delta\hat{y}$). In other words, an appliance switching OFF could be composed of multiple OFF events (at different times) which their sum approximate the $-\Delta\hat{y}$ value. Although these drawbacks might seem discouraging, they served as stepping stones for Jones [9] to find a solution to these problems that present when attempting edge-pairing of clustered events. The author exploited duration and energy conservation information in order to have more reliable edge-pairing, the method was denoted as "first-edge pairing" (see [9] for more details).

Chapter 4

NILM Metrics

This section presents a summary of the most used performance metrics for evaluating NILM. More specifically, we design a novel metric for evaluating disaggregation performance. The metric can be helpful in scenarios where NILM is performed with an architecture such as the one presented in Figure 1.2. Our proposed metric is based on QP [61] and it shows to be robust under various fundamental test cases where it shows promising results.

4.1 Metrics for NILM evaluation

Since the beginning of energy disaggregation research, performance metrics for evaluating NILM have been a topic of debate. Some of the most widely used NILM evaluation metrics will be discussed in a short summary. This will motivate one of our research contributions: the *Power Assignation Metric* (PAM), which will be presented later in this chapter. The performance metrics are borrowed from two previous works [60] and [62], which performed extensive research and insightful debates for reporting accuracy in NILM systems.

NILM performance metrics can be divided in classification and estimation accuracy. The classification metrics measure the how accurately NILM algorithms can predict what appliance is running in each state [60]. On the other hand, the estimation metrics provide with a measure of how accurately NILM algorithms perform in terms of power estimation.

The authors in [62] concluded that the NILM research community is still far from consensus regarding which metrics should be used to report results. Many of the performance metrics are adopted from other areas and applied to NILM without a lack of deep understanding on the behaviour of the metrics.

According to [62], all the classification metrics assume all the events are equally important, which is not realistic since all not appliances require the same energy. Thus, we can already observe how these metrics result problematic as they only contain information of the state appliances being ON/OFF without providing any power information. On the other hand, the estimation metrics allow to measure how accurately the disaggregation algorithm performs by providing overall errors between the ground-truth signals and the

disaggregated signals. Note how these metrics assume that the matching between the disaggregated signals with their respective ground-truth signals has been previously done. To our knowledge, this matching is always performed manually.

In the notation that follows, $y_t^{(m)}$ refers to the ground truth for appliance m at time t, whereas $\hat{y}_t^{(m)}$ refers to the inferred power draw for appliance m at time t. In this section, we present a number of classification and estimation accuracy metrics. The metrics chosen to be reported in this work are based on the most cited by several authors. Later, in Section 4.2, we provide with more detailed examples of were some of the classification and estimation metrics that are widely used in the NILM literature fail in cases that could be presented when attempting source separation.

4.1.1 Classification Metrics

The following metrics are based on binary classification. However, they can be extended to their multi-state form [60]. The terminology used for this part of the works is as follows: tp is true-positives (correctly predicted that the appliance was ON), fp is false-positives (predicted appliance was ON but was OFF), fn is false-negatives (appliance was ON by was predicted OFF), and tn (correctly predicted that the appliance was OFF). It is worth noting that these measures (tp, fp, fn, tn) are accumulations over a period.

Accuracy

$$\text{Acc.} = \frac{tp + tn}{tp + fp + fn + tn} \tag{4.1}$$

Precision

$$\text{precision} = \frac{tp}{tp + fp} \tag{4.2}$$

Recall

$$\text{recall} = \frac{tp}{tp + fn} \tag{4.3}$$

F1-score

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.4}$$

4.1.2 Estimation Metrics

The following estimation metrics (for m individual appliances) are described as follows:

Root Mean-Square Error

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{T} \sum_{m=1}^{M} (\hat{y}_t^{(m)} - y_t^{(m)})^2}{T}} \tag{4.5}$$

Normalized Disaggregation Error

$$\text{NDE} = \frac{\sum_{t=1}^{T}\sum_{m=1}^{M}|\hat{y}_t^{(m)} - y_t^{(m)}|}{\sum_{t=1}^{T}\sum_{m=1}^{M}\left(y_t^{(m)}\right)^2} \tag{4.6}$$

Mean Absolute Error

$$\text{MAE} = \frac{\sum_{t=1}^{T}\sum_{m=1}^{M}|\hat{y}_t^{(m)} - y_t^{(m)}|}{T} \tag{4.7}$$

Estimation Accuracy

$$\text{Est. Acc.} = 1 - \frac{\sum_{t=1}^{T}\sum_{m=1}^{M}|\hat{y}_t^{(m)} - y_t^{(m)}|}{2 \cdot \sum_{t=1}^{T}\sum_{m=1}^{M} y_t^{(m)}} \tag{4.8}$$

4.2 Power Assignation Metric

A metric for the NILM problem should have, among others, the following attributes:

1. **Tracking power assignation**. The metric should be able to measure how well the output power from each of the disaggregated signals matches the power from their corresponding ground-truth signal.

2. **Output permutation robustness**. The metric value should be invariant due to permutations in the ground-truth and/or disaggregated signals.

3. **Input cardinality agnostic**. As the number disaggregated outputs does not necessarily match the number of ground-truth signals, the metric should be able to evaluate somehow the match of each disaggregated signal to a ground-truth appliance.

Given T observations of M appliances $\{y_t^{(m)}\}_{m=1,t=1}^{M,T}$, the set of ground truth appliances, and the associated N inferred disaggregated signals $\{\hat{y}_t^{(n)}\}_{n=1,t=1}^{N,T}$. We define the following error measurement

$$E\left(y,\hat{y}\right) = \min_{\alpha_{mn}}\Bigg[\overbrace{\sum_{t=1}^{T}\sum_{m=1}^{M}\left(y_t^{(m)} - \sum_{n=1}^{N}\alpha_{mn}\hat{y}_t^{(n)}\right)^2}^{\text{Part 1}} + \overbrace{\lambda\sum_{m=1}^{M}\sum_{n=1}^{N}\alpha_{mn}}^{\text{Part 2}}\Bigg] + \underbrace{\min_{\beta_{mn}}\left[\sum_{t=1}^{T}\sum_{m=1}^{M}\left(y_t^{(m)} - \sum_{n=1}^{N}\beta_{mn}\hat{y}_t^{(n)}\right)^2\right]}_{\text{Part 3}} \tag{4.9}$$

subject to

$$\sum_{m=1}^{M} \beta_{mn} = 1 \quad \forall n, \tag{4.10}$$

$$\alpha_{mn}, \beta_{mn} \in \{0, 1\} \quad \forall m, n. \tag{4.11}$$

The objective is to find the set of coefficients α_{mn} and β_{mn} that minimize the error expression in (4.9), which can be formulated as a QP problem [61]. The nature of the problem is of binary assignation.

Intuitively, the α_{mn} coefficients in "part 1" of expression (4.9), are binary indicators to whether the inferred output signal $\hat{y}_t^{(n)}$ should be assigned or not to the ground truth signal $y_t^{(m)}$. We aim to find the minimum error between each of the ground truth signals and the sum of the disaggregated output signals by assigning each of the output signals to one input signal.

The term "part 2" of equation (4.9) is a penalizing term. It penalizes appliances that are disaggregated in their single state components. The term "part 1" will aim to assign the inferred signals to their corresponding ground-truth signals. However, this comes with a "cost", as the term "part 2" will increase as more inferred signals are assigned to the ground-truth signals. If the parameter λ is set to be very large, all of the terms in "part 1" and "part 2" will get zeroed out, except for the ground-truth signals $y_t^{(m)}$ in the term "part 1". Therefore, the minimization will be the sum of this $y_t^{(m)}$ terms. On the other hand, if the parameter λ is set to a value close to zero, it lessens the penalization for disaggregating multi-state appliances.

The term "part 3" of the expression in (4.9) operates similarly to "part 1" of the minimization problem. The β_{mn} coefficients are binary indicators to whether the inferred output signal $\hat{y}_t^{(n)}$ should be assigned or not to the ground truth signal $y_t^{(m)}$. However, the sum of the β_{mn} over all M the possible ground-truth signals should be equal to 1, for all the inferred signals. In simple words, we are forcing the $\hat{y}_t^{(n)}$ inferred signal to be assigned to one $y_t^{(m)}$ ground-truth signal (4.10). In "part 1", we did not have to constrain the α_{mn} coefficients to have the same condition as the as that would not allow for the regularization term in "part 2" to perform its task.

Although the metric error metric in expression (4.9) seems complete (at least mathematically), if suffers from numerical instability. Due to the fact that we are working with power signals and we have a squared term for "part 1" and "part 3". Moreover, we are summing over all the appliances and all the time instants. Therefore, the equation in (4.9) grows rapidly. In order to avoid numerical instability, we introduced a scaling factor γ. This scaling factor consists of summing over all the total energy square of each of the $y_t^{(m)}$ ground truth signals. The scaling factor is given by

$$\gamma = \sum_{m=1}^{M} \sum_{t=1}^{T} (y_t^{(m)})^2 + \epsilon, \tag{4.12}$$

where ϵ is a small number, for this work it was set to $\approx 1 \times 10^{-3}$. Therefore, the error metric described in (4.9) is divided by the scaling factor introduced in (4.12) in order to allow for numerical stability. Further, due to the numerical solver that was employed (see Section 4.3) suffers from high computational times for integer constraints, the condition in (4.11) had to be relaxed for the α_{mn} and β_{mn} coefficients to take values in the interval $[0, 1]$. Thus, the error metric in (4.9), that we denoted *power assignation metric*, can be rewritten (by incorporating the scaling factor in (4.12) and by relaxing the constraint in (4.9)) as

$$\begin{aligned} E_s\left(y, \hat{y}\right) = \min_{\alpha_{mn}} \Bigg[& \frac{\sum_{t=1}^{T} \sum_{m=1}^{M} \left(y_t^{(m)} - \sum_{n=1}^{N} \alpha_{mn} \hat{y}_t^{(n)} \right)^2}{\gamma} \\ & + \frac{\lambda \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_{mn}}{\gamma} \Bigg] \\ + \min_{\beta_{mn}} \Bigg[& \frac{\sum_{t=1}^{T} \sum_{m=1}^{M} \left(y_t^{(m)} - \sum_{n=1}^{N} \beta_{mn} \hat{y}_t^{(n)} \right)^2}{\gamma} \Bigg] \end{aligned} \tag{4.13}$$

subject to

$$\sum_{m=1}^{M} \beta_{mn} = 1 \quad \forall n, \tag{4.14}$$

$$\alpha_{mn}, \beta_{mn} \in [0, 1] \quad \forall m, n. \tag{4.15}$$

4.3 Experimental Setup

Our metric was coded and tested in Python 3.7. All tests ran on a Mac Pro (2017 model) with a 2.3GHz Intel Core i5 processor and 8GB of memory. The software `CVXPY` [63, 64] was employed for solving the metric in (4.13), constrained to conditions (4.14) and (4.15) (4.15). This numerical solver allows to integrate QP with Python. Moreover, it allows to relax the condition in (4.11), meaning that the α_{mn} and β_{mn} coefficients can now be $0 \geq \alpha_{mn} \geq 1$, $0 \geq \beta_{mn} \geq 1$.

We take data from House-1, Block-1, (comprised of nine day's worth of data) of the RAE dataset [8] which is sampled at 1 Hz.

For Section 4.4, the optimizer was fed with signals containing 2300 samples ($\sim$ 40 minutes), no preprocessing was applied to the signals.

For Section 4.5, 12 hours of data (43,200 samples) were selected for three appliances: the fridge, the clothes dryer and the dishwasher. The data was partitioned in 3 windows of

4 hours. Each window contained 14,400 samples for each of the appliances. the optimizer was fed with these windows of data.

4.4 Results of Fundamental Test Cases

We want to test how well the metric introduced in (4.15) performs under several possible test case scenarios that appear when attempting appliance disaggregation in architectures such as the one presented in Figure 1.2. Although the signals that were used are composed of short samples from the RAE dataset (see Section 4.3), they help us to illustrate the performance of our metric in several fundamental cases in order to test its robustness.

For this section, we will work with two appliances: the fridge and dishwasher. Figure 4.1 displays the power signal of one cycle of the fridge. Figure 4.2 shows the dishwasher signal, composed of two operational modes (see Figures 4.3 and 4.4).

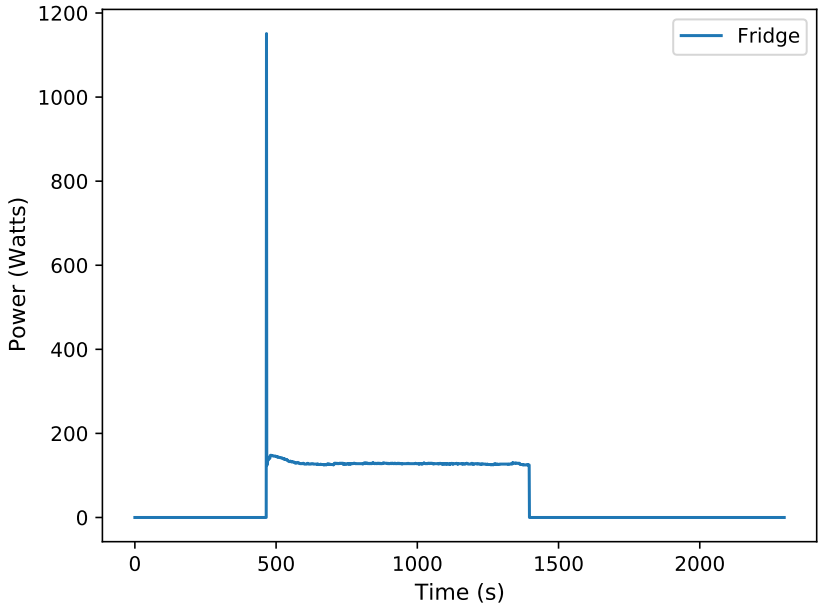

Figure 4.1: An example of one cycle of the fridge signal from RAE [8].

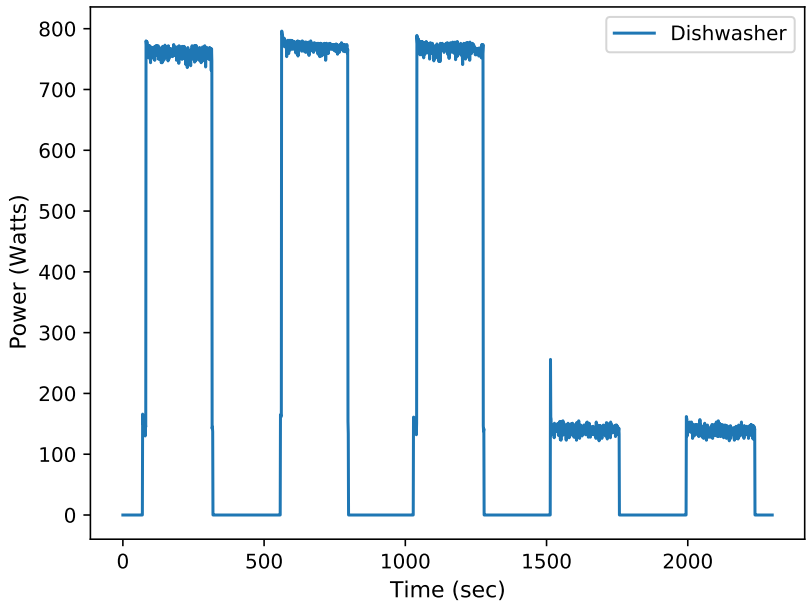

Figure 4.2: An example of two states of the dishwasher appliances from RAE [8]

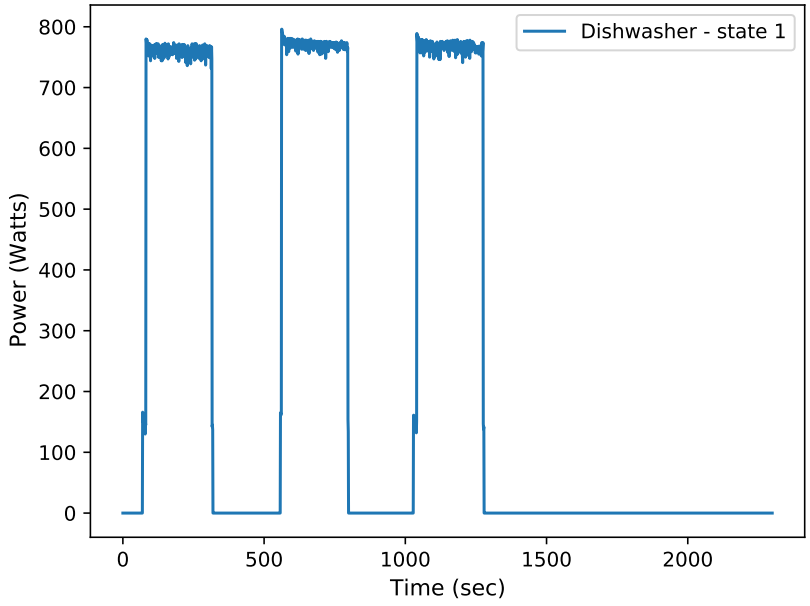

Figure 4.3: Three cycles of the first operational mode of the dishwasher in Figure 4.2

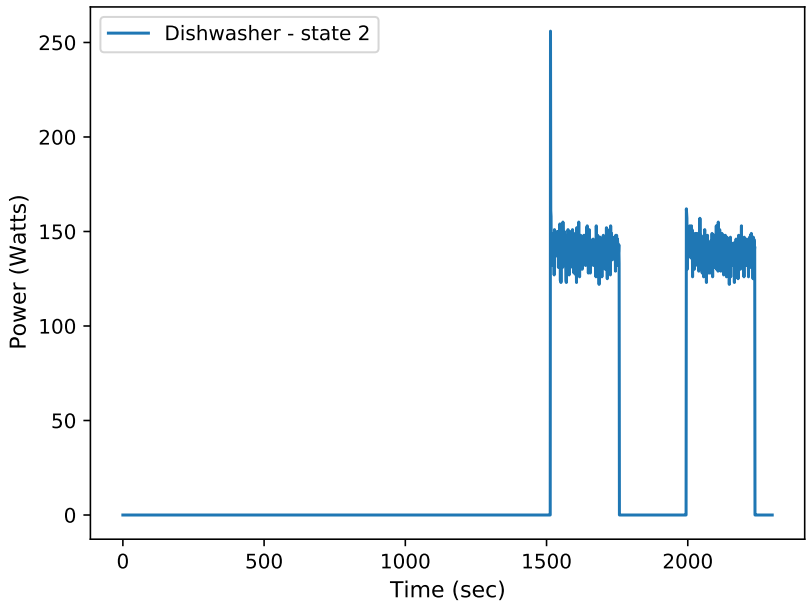

Figure 4.4: Two cycles of the second operational mode of the dishwasher in Figure 4.2.

Test Case 1

Let us suppose the aggregate signal y_t is composed of two ground-truth appliances $y_t^{(1)}$ and $y_t^{(2)}$, shown in Figures 4.1 and 4.2 respectively. Later, we perform disaggregation with an unsupervised algorithm (such as the ones presented in [9, 10, 11] [1]) using the architecture from Figure 1.2 and obtain the inferred signal $\hat{y}_t^{(1)}$ which is composed by the sum of the two ground-truth appliances, as depicted in Figure 4.5. We refer to this disaggregator as *disaggregator 1*.

To evaluate how our *disaggregator 1* performed, we could be tempted to simply compute the RMSE. The RMSE value would be zero. Therefore, we could conclude the disaggregator is performing with no error. However, this is completely misleading, the disaggregator is just aggregating both input ground-truth signals as shown in Figure 4.6. On the other hand, the ideal disaggregation would separate the signals as depicted in Figure 4.7. We refer to this disaggregator as *disaggregator 2*.

Note that unless specified, the penalizing term λ is set to be equal to 1. Table 4.1 shows a comparison between the RMSE and our proposed metric computed to both *disaggregator 1* and *disaggregator 2*. Our proposed metric is able to penalize when the disaggregation is

[1]Note how the following inputs/outputs for the following *disaggregators* are formulated hypothetically. This does not mean the referenced works necessarily provide with these results. However a robust metric should be able to differentiate between these hypothetical test cases.

performed poorly, something that the RMSE is not able to detect. Moreover, due to the nature of our error performance metric to be characterized as an assignation problem, the β_{mn} coefficients can be interpreted later, as to "what disaggregated signal corresponds to what ground-truth signal", i.e., our metric automatically assigns the inferred or disaggregated signals to the ground-truth signals. Further, we can express these β_{mn} coefficients in a $(M \times N)$ matrix. Let us denote this matrix $\mathbf{B}$. The entries for the matrix $\mathbf{B}$ are given by

$$\mathbf{B} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1N} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{M1} & \beta_{M2} & \cdots & \beta_{MN} \end{bmatrix}. \tag{4.16}$$

We now show how $\mathbf{B}$ can be interpreted for *disaggregator 2*. The resulting matrix after computing the metric in (4.13) can be expressed as

$$\mathbf{B}_{disaggregator\ 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.17}$$

where the matrix rows represent the m ground-truth signal and the columns represent the n inferred signal. For the particular case of *disaggregator 2*, there are 2 ground-truth signals and 2 inferred signals. Thus, the $\beta_{11} = 1$ coefficient can be interpreted as "the inferred signal $\hat{y}_t^{(1)}$ can be assigned to the ground-truth signal $y_t^{(1)}$". Moreover, the coefficient $\beta_{22} = 1$ can be similarly interpreted as "the inferred signal $\hat{y}_t^{(2)}$ can be assigned to the ground-truth signal $y_t^{(2)}$". We can see how the coefficients $\beta_{12}, \beta_{21} = 0$ can be interpreted as "the inferred signal $\hat{y}_t^{(1)}$ cannot be assigned to the ground-truth signal $y_t^{(2)}$" and "the inferred signal $\hat{y}_t^{(2)}$ cannot be assigned to the ground-truth signal $y_t^{(1)}$", respectively. A similar reasoning can be applied to different β_{mn} coefficients for different M and N values. From this point of the thesis, the β_{mn} coefficients will not be written for each of the test cases. Nevertheless, the reader can assume the inferred signals were matched correctly with their corresponding input signals. In other words, the β_{mn} coefficients have been assigned correctly. In Section 4.5 we discuss the interpretation of the β_{mn} coefficients for signals composed of larger number of ground-truth and/or inferred signals.

Table 4.1: RMSE and proposed metric values for *disaggregator 1* and *disaggregator 2*.

Error Metric	**Disaggregator 1**	**Disaggregator 2**
RMSE	0	0
Proposed	0.08	0

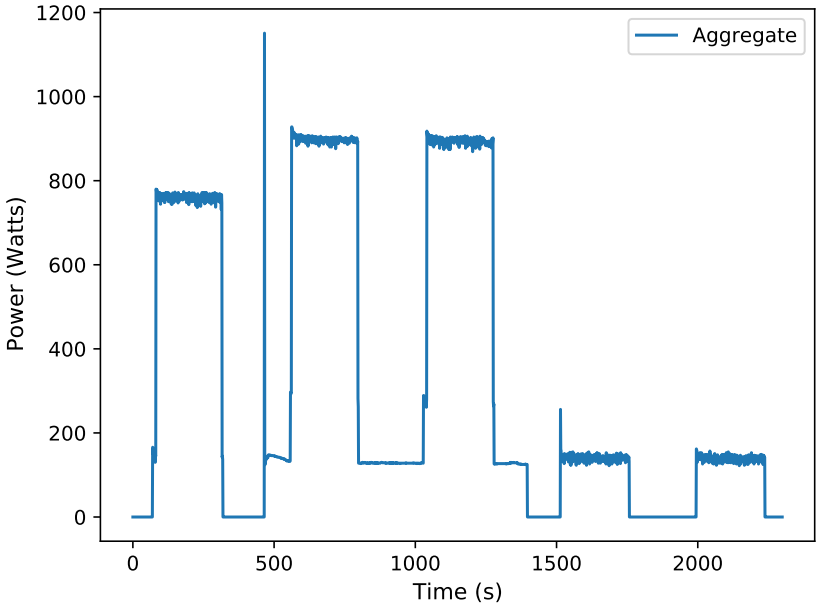

Figure 4.5: Aggregate signal composed of the fridge and the dishwasher signals (shown in Figures 4.1 and 4.2, respectively).

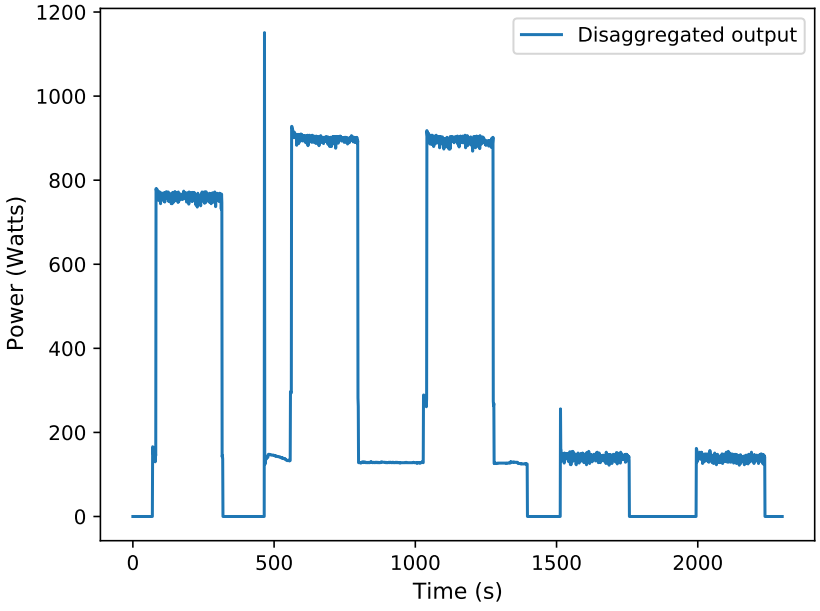

Figure 4.6: An example of poor disaggregation performance. The inferred signal is the same as the aggregate fed into the disaggregator.

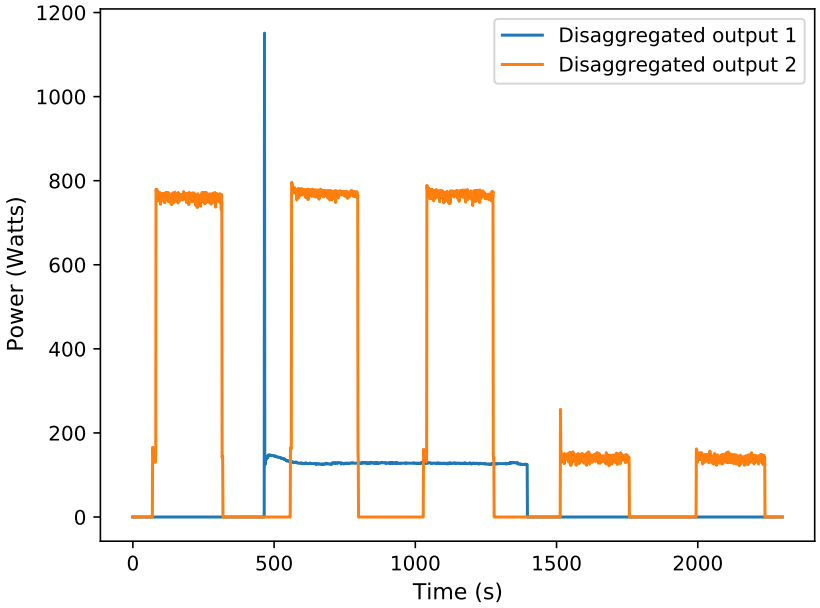

Figure 4.7: An example of optimal disaggregation. The inferred signals match the ground-truth signals.

Test Case 2

A metric that evaluates disaggregation should be invariant to the order of the ground-truth and inferred signals, i.e., the error should be the same if we interchange the order of the ground-truth and/or the inferred signals. Now, let us suppose that the aggregate signal is composed of two ground-truths $y_t^{(1)}$ and $y_t^{(2)}$ shown in Figures 4.1 and 4.2 respectively. Later, we perform disaggregation and obtain the inferred signals $\hat{y}_t^{(1)}$ and $\hat{y}_t^{(2)}$ shown in Figure 4.7. Note that the disaggregator was previously denoted as *disaggregator 2*.

Let us denote a new disaggregator (*disaggregator 3*) in which we have the same aggregate signal as in (*disaggregator 2*), composed by the same ground-truth signals. The inferred signals $\hat{y}_t^{(1)}$ and $\hat{y}_t^{(2)}$ are the same as the ones depicted in Figure 4.7. Nevertheless, with the distinction that the inferred signals are permuted (i.e., $\hat{y}_t^{(1)}$ belongs to the dishwasher and $\hat{y}_t^{(2)}$ belongs to the fridge).

Both, *disaggregator 2* and *disaggregator 3* should have the same error metric, regardless of order in which the ground-truth/inferred signals appear in the disaggregator system. Table 4.2 shows the calculation of the proposed error metric for *disaggregator 2* and *disaggregator 3*.

Table 4.2: Proposed metric values for *disaggregator 2* and *disaggregator 3*.

Error Metric	**Disaggregator 2**	**Disaggregator 3**
Proposed	0	0

Test Case 3

Our metric should be robust and penalize scenarios where the appliances are disaggregated in their single state components. Let us suppose that we have the same aggregate as the previous test cases (test case 1 and 2). The aggregate is composed of the two ground-truths $y_t^{(1)}$ and $y_t^{(2)}$ (Figures 4.1 and 4.2). The source separator (now denoted *disaggregator 4*) decomposed the aggregate in three inferred signals: $\hat{y}_t^{(1)}$, $\hat{y}_t^{(2)}$ and $\hat{y}_t^{(3)}$. The first inferred signal $\hat{y}_t^{(1)}$ belongs to the fridge (Figure 4.1). The inferred signals $\hat{y}_t^{(2)}$ and $\hat{y}_t^{(3)}$, are the the first and second operational mode of the dishwasher (Figures 4.3 and 4.4), respectively. Technically speaking, the *disaggregator 4* is not performing poorly. However. it is decomposing one of the appliances contributing to the aggregate into its two operational modes. This should be reflected in a metric. Table 4.3 compares the performance of *disaggregator 2* and *disaggregator 4* with our proposed metric. It can be observed that the *disaggregator 4* has a larger error value. The penalizing weight was set to $\lambda = 1 \times 10^6$. The penalizing parameter λ can be set according to "how much" we want to emphasize cases where multi-state appliances are broken into their single operational modes. Note how clustering algorithms such as the one described in 3.2 or [9, 10, 11] will disaggregate multi-state appliances into a set of single-state appliances and therefore, it is worth addressing these type of scenarios.

Table 4.3: Proposed metric values for *disaggregator 2* and *disaggregator 4*.

Error Metric	**Disaggregator 2**	**Disaggregator 3**
Proposed	0.004	0.006

Test Case 4

A disaggregation metric should be power variant; i.e., if a disaggregator decomposes a signal into a scaled version of the ground-truth signals, a robust error metric should be able to detect such scaling.

Let us suppose that our aggregate signal is composed of two ground-truth signals $y_t^{(1)}$ and $y_t^{(2)}$ (Figures 4.1 and 4.2, respectively). The aggregate signal is decomposed into a scaled version of the ground truth signals. Therefore $\hat{y}_t^{(1)} = s \cdot y_t^{(1)}$ and $\hat{y}_t^{(2)} = s \cdot y_t^{(2)}$, where $s \in \mathbb{R}^+$. For this example, we set $s = 0.8$. We denote this as the *disaggregator 5*. Classification metrics such as F1-score are invariant to cases where the inferred signals are ON/OFF at the same time instants as their corresponding ground-truth signals, yet the

power difference between both can be off by a large variation. The inferred signals $\hat{y}_t^{(1)}$ (scaled version of the fridge signal) and $\hat{y}_t^{(2)}$ (scaled version of the dishwasher signal) are depicted in Figure 4.8. Finally, Table 4.4 presents a summary of a comparison between the performance of the *disaggregator 2* and *disaggregator 5*. We calculated our proposed metric versus F1-score, showing our method is capable of recognizing power variations. On the other hand, the F1-score was not able to detect the power differences between the ground-truth signals and the inferred signals.

Table 4.4: Proposed metric values for *disaggregator 2* and *disaggregator 5*.

Error Metric	**Disaggregator 2**	**Disaggregator 5**
F1-Score (fridge)	100 %	100 %
F1-Score (dishwasher)	100 %	100 %
Proposed	0	0.06

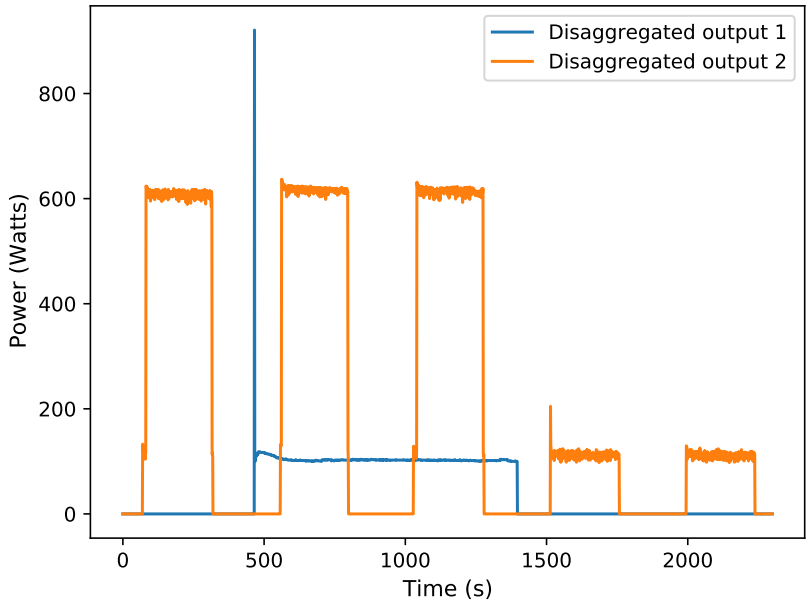

Figure 4.8: An example of scaled disaggregation. The inferred signals are a scaled version of the ground-truth signals.

4.5 Results Using a published NILM Algorithm

We have tested the robustness of our metric for several fundamental cases that could appear in disaggregation methods with architectures such as the one presented in Figure 1.2. The next natural step is to test the proposed metric with a state-of-the-art disaggregator using

more larger and more complex power signals. We chose the method proposed by Jones [9] as to our knowledge it offers good disaggregation performance in terms of RMSE, F1-Score and MAE. The disaggregator is a unsupervised clustering algorithm based on non-parametric GMMs.

We generated a synthetic aggregate composed of 12 hours from: the fridge, the clothes dryer and the dishwasher signals from house 1, block 1 of the RAE dataset. In order to asses our metric's performance, we selected periods were all the appliances were running simultaneously. Figure 4.9 shows the individual ground-truth appliances that contributed to the raw aggregate. Later, the raw aggregate passes through a filtering stage (see Figure 1.2), where we selected the CP filter to perform this task (note that we could also use the CUSUM Kalman filter or the steady-state block filter [2]). Next, the filtered aggregate signal is fed to Jones' disaggregator [9] which produces the inferred signals in Figure 4.10.

Finally, we compute our metric for each of the three windows (see Section 4.3). More specifically, we solve the QP problem from equation (4.13). The computed errors for windows 1, 2 and 3 are summarized in Table 4.6. It can be observed that for about the first 4 days, the disaggregator is able to decompose the fridge, the clothes dryer and the dishwasher accurately. Therefore, the metric for our first window is low compared to the second and third window. It can be seen that the disaggregator is not able to reconstruct the fridge signal for about the final 8 hours. Thus, the metric is larger for the last two windows.

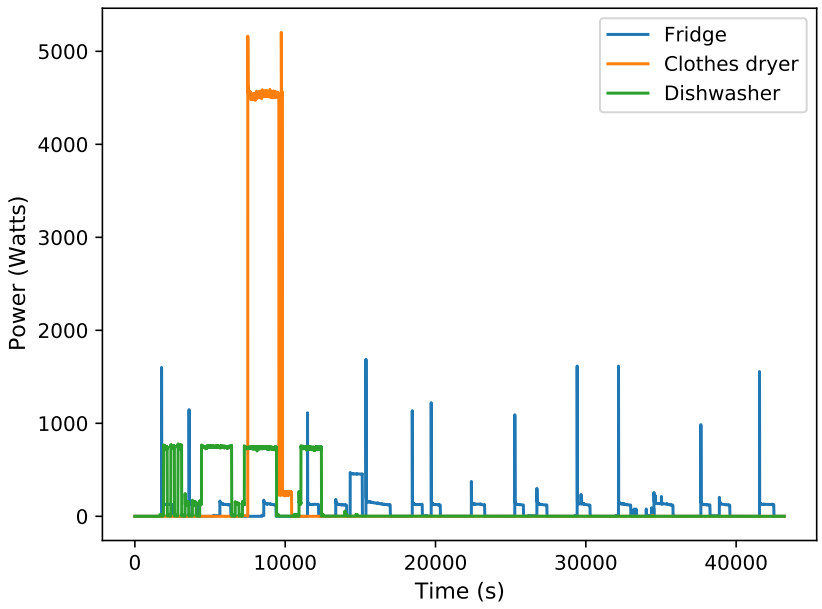

Figure 4.9: Ground-truth appliances that contribute to the raw synthetic aggregate.

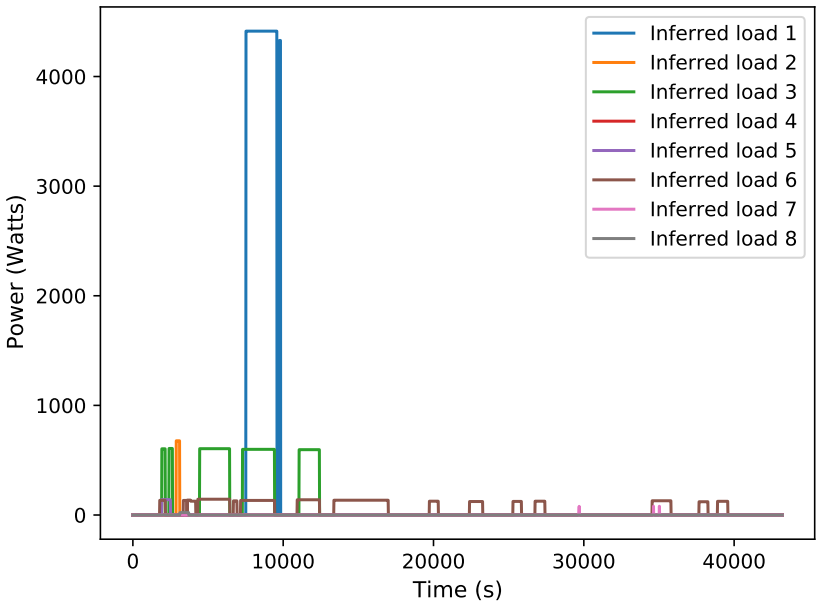

Figure 4.10: Inferred loads after performing disaggregation via non-parametric GMM [9]. The individual appliances contributing to the aggregate can be observed in Figure 4.9

The inferred appliances were manually matched and compared to the binary β_{mn} coefficients (note that for every window, there are $m \times n$ binary coefficients). Table 4.5 shows the manually matched inferred signals with their respective ground-truths.

Table 4.5: Manually matched disaggregated appliances (Figure 4.10) with their corresponding ground-truth appliances (Figure 4.9).

Inferred signals	**Matched ground-truth**
Inferred load 1	Clothes dryer
Inferred load 2	Dishwasher
Inferred load 3	Dishwasher
Inferred load 4 (zero vector)	-
Inferred load 5	Dishwasher
Inferred load 6	Fridge
Inferred load 7	Fridge
Inferred load 8	Dishwasher

Let us denote the matrices $\mathbf{B}_{w_1}$, $\mathbf{B}_{w_2}$, and $\mathbf{B}_{w_3}$ containing the binary coefficients for the first, the second, and the third windows, respectively. This can be done for all of the computed errors for each window. However, we only show the first matrix of binary

coefficients as an example of how these coefficients can be interpreted as the matches between inferred and ground-truth signals. Note that for this particular example, $M = 3$ and $N = 8$.

It can be seen from the matrix in (4.18) that the first three inferred loads (the first three columns. Also, the loads with the largest energies), have a binary assignation of $\mathbf{B} = 1$ in their respective components. The inferred loads 5 and 6 (columns 6 and 7 of the $\mathbf{B}$ matrix) were swapped for the fridge and dishwasher. This is because these two appliances have operational modes with similar power values. The inferred load 7 has equal weights and finally, load 8 was assigned correctly. Manually matching the disaggregated appliances with the ground-truth signals is challenging and due to disaggregator's behaviour, could lead to a poor match between the inferred and ground-truth signals.

Table 4.6: Computed errors using the proposed PAM.

Error (window 1)	**Error (window 2)**	**Error (window 3)**
0.009	0.788	0.631

$$\mathbf{B}_{w_1} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{3} & 0.619 & 0.352 & \frac{1}{3} & 0.193 \\ 1 & 0 & 0 & \frac{1}{3} & 0.157 & 0.180 & \frac{1}{3} & 0.312 \\ 0 & 1 & 1 & \frac{1}{3} & 0.224 & 0.467 & \frac{1}{3} & 0.495 \end{bmatrix}. \tag{4.18}$$

4.6 Discussions

The proposed *power assignation metric* is a good alternative to the traditional classification and estimation metrics presented in Section 4.1. Although we presented the metric focusing on NILM, it can be used for other research areas where source separation is the main application.

The fact that the β_{mn} coefficients in our metric are not strictly 1 or 0 is due to the relaxation that was introduced the QP in equation 4.13 in order to allow the `cvxpy` optimizer to work with the problem in a reasonable execution time. Constraining the problem to the condition in (4.11) would ideally provide with better results for the β_{mn} coefficients as the analogy of "strictly matching one disaggregated signal to one ground-truth signal" would be met. This is left to future work. Another potential topic to look after, would be to explore down-sampling the signals in order to speed up the processing time for solving the QP problem.

Finally, we can see how algorithms such as the ones presented in [9, 10, 11, 65, 15] (that use an architecture such as the one presented in Figure 1.2), among others, can get benefited from our proposed metric. The metric not only proves to be robust under fundamental test cases, but also automatically assigns the disaggregated signals with their corresponding

ground-truth signals. Even when the number of disaggregated and ground-truth signals is not equal.

Chapter 5

Conclusions

This final chapter provides with a synthesis of the main results from this thesis and discusses the significance of this work's contributions (Section 5.1). In Section 5.2 we explore the limitations of the work we have presented in Sections 2.2, 3.2 and 4.2.

5.1 Significance

This thesis has explored an attempt to solve NILM (leaving the labelling problem aside). First, we introduced two filters that produce steady-state representations of complex raw aggregate data. The first filter (Subsection 2.2.2) is inspired by change point detection theory and hypothesis testing. On the other hand, the second approach (Subsection 2.2.3) fuses the ideas of the CUSUM algorithm and the Kalman filter. The proposed filters outperform an existing method in terms of running time. The proposed filters' parameters have been optimized based on an objective function defined between the RMSE and the Hoyer sparsity. The local optimal filter parameters were selected using various sparsity loss weights. The first filter and the steady-state block filter [2] present a similar performance in terms of the defined objective function. Despite the fact that the CP filter requires future samples to operate, NILM is usually performed post-consumption. Therefore, this would not prevent the filter to be implemented in such systems. Although the CUSUM Kalman filter did not perform as accurately as the CP filter and the method proposed in [2], due to the its fixed threshold and due to the fact that the Kalman filter does not depend on future sample, it can be a good alternative to the CP filter and the steady-state block filter, especially when real-time processing is required. Although the two proposed filters are used in the NILM context, they can be used for other digital signal processing (DSP) tasks where event detection is a crucial task. This is significant because there are areas such as target tracking and fault detection where detecting events in a fast and accurate manner is essential. Our hope is that the work presented in this thesis has a broader reach than just NILM.

With the aid of the two proposed filters, the aggregate signals have more consistent activations/deactivations. Using the firsts differences from the proposed filters, we developed an unsupervised disaggregator based on a hybrid Knapsack problem and a GMM model. We showed the idea of how a set of activations and deactivations can be modelled using Gaussian distributions. Later, we showed how these activations can be paired with their corresponding deactivations, in order to reconstruct the disaggregated signals as a set of uni-modal appliances. The KP has proven to be useful when appliances simultaneously turn ON/OFF. Moreover, a similarity measure between transients was applied in order to differentiate appliance modes with similar power values. Even though the proposed method is event-based, we leveraged the transient information.

Finally, a novel metric based on QP was developed, which is significant because the metric shows to be robust under several fundamental source separation test cases. The proposed metric achieves to detect faulty scenarios where classification or estimation metrics do not (such as the RMSE and the F1-Score). The *power assignation metric* demonstrated to track power differences between ground-truth signals and disaggregated signals, it operates successfully regardless the mismatch between the number of ground-truth and disaggregated signals, and it is not fragile to permutations. Further, taking advantage of the binary assignation nature of quadratic programming problem, the *power assignation metric* shows promising potential to automatically match each of the disaggregated signals to their corresponding ground-truth signal. The *power assignation metric* can be used not only for NILM but in other source separation applications.

5.2 Limitations

Even though we believe in the significance of our work, and this work contributes to helping solve the NILM problem, there are some limitations that present in it. The CUSUM Kalman filter from Subsection 2.2.3 has the worst performance from the three compared filters. This is mainly due to the CUSUM algorithm. The CUSUM algorithm has a fixed threshold and the largest the value from the threshold, the less events that will get considered events. Although this type of behaviour might seem like an advantage for large appliances such as the clothes dryer and the heat pump, other appliances such as the fridge and the furnace will suffer from this undesired filter behaviour.

The proposed algorithm from Section 3.2 showed promising results in terms of estimated energy for appliances that have cyclical behaviour for certain times in their operational modes (clothes dryer, fridge, and furnace). Nevertheless, the algorithm fails when trying to disaggregate multi-state appliances or more complex aggregate signals. This is mainly due to undesired KP behaviour, i.e., the KP can miss-classify appliances in certain scenarios (see Section 3.5 for more details). Further, pairing ON/OFF edges by simply assigning the next OFF event with the previous ON event is a trivial task. However, it is a poor

representation of how to model the operational modes of the appliances, which could lead to poor disaggregation performance.

Finally, the proposed metric from Section 4.2 has been proved to work under various disaggregation test cases. It showed a robust behaviour and it could potentially be used for automatically matching ground-truths with their respective disaggregated signals due to its binary assignation nature. Further, if the ground-truths contain labels, this could be one way of solving the NILM labelling problem. Nevertheless, we had to relax the problem (see Section 4.3. Therefore, if the disaggregator's performance is poor, the binary assignation or ground-truth/inferred signals matching will also perform poorly due to the relaxation introduced in expression (4.15).

5.3 Future Work

Our work provides with a solid foundation for continuing research in different areas. The behaviour of the CUSUM Kalman filter from Subsection 2.2.3 could potentially be improved in two ways: modifying the threshold from the CUSUM algorithm to be adaptive and not fixed (e.g., the larger the power values, the larger the threshold), or by replacing the CUSUM algorithm with a more robust algorithm for change point detection. There is no clear evidence that suggests how to set the initial values from the process noise covariance and the measurement noise parameters for the CUSUM Kalman filter, yet, optimizing these two extra parameters might lead to better results in terms of the objective function defined for this part of the thesis. This could potentially be achieved with the dual annealing method proposed in [9].

Another item to look at is to incorporate duration information for the algorithm introduced in Section 3.2. Further examination of the edge pairing in the algorithm suggested that a more sophisticated pairing method is needed, such as *first-n pairing* [9]. Moreover, energy conservation considerations have been proved to lead to better results.

The *power assignation metric* in the form introduced in (4.11) could be implemented using a different numerical solver. The metric would be most likely able to perform the automatic matching regardless of the disaggregator's performance. In order to avoid numerical instability, the metric was computed in windows of 4 hours (for 1 Hz data). Another approach that could be looked after is to down-sample the data before computing the error metric. This could be done with the same sampling frequency over all the signals or down-sampling on low-frequency parts of the signal.

References

[1] A. Rodriguez-Silva and S. Makonin, 'Universal non-intrusive load monitoring (unilm) using filter pipelines, probabilistic knapsack, and labelled partition maps', in *2019 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, IEEE, 2019, pp. 1–6 (cit. on pp. iv, 1, 16).

[2] R. Jones, A. Rodriguez-Silva and S. Makonin, 'Increasing the accuracy and speed of universal non-intrusive load monitoring (unilm) using a novel real-time steady-state block filter', in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, IEEE, 2020, pp. 1–5 (cit. on pp. iv, 7, 18, 20, 24, 25, 36, 51, 55).

[3] G. W. Hart, 'Nonintrusive appliance load monitoring', *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992 (cit. on p. 1).

[4] S. Makonin, 'Hue: The hourly usage of energy dataset for buildings in british columbia', Tech. Rep., 2018 (cit. on p. 1).

[5] J. Kolter and M. Johnson, 'REDD: A public data set for energy disaggregation research', in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, vol. 25, 2011, pp. 59–62 (cit. on pp. 1, 34).

[6] S. Makonin, B. Ellert, I. V. Bajić and F. Popowich, 'Electricity, water, and natural gas consumption of a residential house in canada from 2012 to 2014', *Scientific data*, vol. 3, p. 160 037, 2016 (cit. on p. 1).

[7] D. Murray, L. Stankovic and V. Stankovic, 'An electrical load measurements dataset of united kingdom households from a two-year longitudinal study', *Scientific data*, vol. 4, p. 160 122, 2017 (cit. on p. 1).

[8] S. Makonin, Z. Wang and C. Tumpach, 'RAE: The Rainforest Automation energy dataset for smart grid meter data analysis', *data*, vol. 3, no. 1, p. 8, 2018 (cit. on pp. 1, 31, 42–44).

[9] R. Jones, 'Non-parametric modeling in non-intrusive load monitoring', Ph.D. dissertation, Simon Fraser University, Canada, 2020 (cit. on pp. 1–3, 7, 15–18, 20, 26, 33, 37, 45, 49, 51–53, 57).

[10] B. Zhao, L. Stankovic and V. Stankovic, 'On a training-less solution for non-intrusive appliance load monitoring using graph signal processing', *IEEE Access*, vol. 4, pp. 1784–1799, 2016 (cit. on pp. 2, 3, 5, 28, 29, 45, 49, 53).

[11] M. Qureshi, C. Ghiaus and N. Ahmad, 'A blind event-based learning algorithm for non-intrusive load disaggregation', *International Journal of Electrical Power & Energy Systems*, vol. 129, p. 106 834, 2021 (cit. on pp. 2, 45, 49, 53).

[12] R. Jia, Y. Gao and C. J. Spanos, 'A fully unsupervised non-intrusive load monitoring framework', in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, 2015, pp. 872–878 (cit. on pp. 2, 28).

[13] T. Bernard and M. Marx, 'Unsupervised learning algorithm using multiple electrical low and high frequency features for the task of load disaggregation', in *Proceedings of the 3rd international workshop on NILM*, 2016 (cit. on p. 2).

[14] A. Cominola, M. Giuliani, D. Piga, A. Castelletti and A. E. Rizzoli, 'A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring', *Applied energy*, vol. 185, pp. 331–344, 2017 (cit. on p. 2).

[15] G. Jacobs and P. Henneaux, 'Unsupervised learning procedure for nilm applications', in *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, IEEE, 2020, pp. 559–564 (cit. on pp. 2, 53).

[16] S. Makonin, F. Popowich and B. Gill, 'The cognitive power meter: Looking beyond the smart meter', in *2013 26th IEEE Canadian conference on electrical and computer engineering (CCECE)*, IEEE, 2013, pp. 1–5 (cit. on p. 2).

[17] S. Makonin, 'Investigating the switch continuity principle assumed in non-intrusive load monitoring (nilm)', in *2016 IEEE Canadian conference on electrical and computer engineering (CCECE)*, IEEE, 2016, pp. 1–4 (cit. on pp. 2, 30).

[18] R. Bonfigli, S. Squartini, M. Fagiani and F. Piazza, 'Unsupervised algorithms for non-intrusive load monitoring: An up-to-date overview', in *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, IEEE, 2015, pp. 1175–1180 (cit. on pp. 3, 5).

[19] S. Verma, S. Singh and A. Majumdar, 'Multi label restricted boltzmann machine for non-intrusive load monitoring', in *ICASSP 2019-2019 IEEE International Conference*

on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 8345–8349 (cit. on p. 3).

[20] H. Lange, M. Bergés and Z. Kolter, 'Neural variational identification and filtering for stochastic non-linear dynamical systems with application to non-intrusive load monitoring', in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8340–8344 (cit. on p. 3).

[21] A. Harell, S. Makonin and I. V. Bajić, 'Wavenilm: A causal neural network for power disaggregation from the complex power signal', in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8335–8339 (cit. on p. 3).

[22] K. He, L. Stankovic, J. Liao and V. Stankovic, 'Non-intrusive load disaggregation using graph signal processing', *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1739–1747, 2016 (cit. on p. 5).

[23] B. Zhao, K. He, L. Stankovic and V. Stankovic, 'Improving event-based non-intrusive load monitoring using graph signal processing', *IEEE Access*, vol. 6, pp. 53 944–53 959, 2018 (cit. on pp. 5–8, 27).

[24] 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, 2012 (cit. on p. 5).

[25] M. Zeifman and K. Roth, 'Nonintrusive appliance load monitoring: Review and outlook', *IEEE transactions on Consumer Electronics*, vol. 57, no. 1, pp. 76–84, 2011 (cit. on p. 5).

[26] 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*, 2012 (cit. on p. 5).

[27] M. Aiad and P. H. Lee, 'Non-intrusive load disaggregation with adaptive estimations of devices main power effects and two-way interactions', *Energy and Buildings*, vol. 130, pp. 131–139, 2016 (cit. on pp. 5, 6, 8).

[28] M. L. 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 (cit. on p. 5).

[29] F. Englert, P. Lieser, A. Alhamoud, D. Boehnstedt and R. Steinmetz, 'Electricity-metering in a connected world: Virtual sensors for estimating the electricity con-

sumption of iot appliances', in *2015 3rd International Conference on Future Internet of Things and Cloud*, IEEE, 2015, pp. 317–324 (cit. on p. 6).

[30] M. Weiss, A. Helfenstein, F. Mattern and T. Staake, 'Leveraging smart meter data to recognize home appliances', in *2012 IEEE International Conference on Pervasive Computing and Communications*, IEEE, 2012, pp. 190–197 (cit. on pp. 6, 8).

[31] L. Yin, R. Yang, M. Gabbouj and Y. Neuvo, 'Weighted median filters: A tutorial', *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 3, pp. 157–192, 1996 (cit. on p. 6).

[32] W. R. Schucany, 'Kernel smoothers: An overview of curve estimators for the first graduate course in nonparametric statistics', *Statistical Science*, pp. 663–675, 2004 (cit. on p. 6).

[33] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006 (cit. on p. 6).

[34] D. Egarter and W. Elmenreich, 'Autonomous load disaggregation approach based on active power measurements', in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, IEEE, 2015, pp. 293–298 (cit. on pp. 6, 8).

[35] P. Perona and J. Malik, 'Scale-space and edge detection using anisotropic diffusion', *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 7, pp. 629–639, 1990 (cit. on p. 6).

[36] D. Barash, 'Bilateral filtering and anisotropic difusion: Towards a unified viewpoint', in *International Conference on Scale-Space Theories in Computer Vision*, Springer, 2001, pp. 273–280 (cit. on pp. 6–8).

[37] E. S. Gastal and M. M. Oliveira, 'Domain transform for edge-aware image and video processing', in *ACM Transactions on Graphics (ToG)*, ACM, vol. 30, 2011, p. 69 (cit. on pp. 6–8).

[38] K. Basu, A. Hably, V. Debusschere, S. Bacha, G. J. Driven and A. Ovalle, 'A comparative study of low sampling non intrusive load dis-aggregation', in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2016, pp. 5137–5142 (cit. on p. 8).

[39] C. Tomasi and R. Manduchi, 'Bilateral filtering for gray and color images.', in *Iccv*, vol. 98, 1998, p. 2 (cit. on p. 8).

[40] J. A. Rice, *Mathematical Statistics and Data Analysis*. Cengage Learning, 2006 (cit. on p. 9).

[41] N. Kovvali, M. Banavar and A. Spanias, *An introduction to kalman filtering with matlab examples*, 2. Morgan & Claypool Publishers, 2013, vol. 6, pp. 1–81 (cit. on pp. 15, 65, 66).

[42] F. Gustafsson and F. Gustafsson, *Adaptive filtering and change detection*. Citeseer, 2000, vol. 1 (cit. on pp. 15, 65, 67, 68).

[43] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2014 (cit. on pp. 15, 26, 65, 69, 71).

[44] P. O. Hoyer, 'Non-negative matrix factorization with sparseness constraints.', *Journal of machine learning research*, vol. 5, no. 9, 2004 (cit. on p. 16).

[45] D. Egarter, A. Sobe and W. Elmenreich, 'Evolving non-intrusive load monitoring', in *European Conference on the Applications of Evolutionary Computation*, Springer, 2013, pp. 182–191 (cit. on pp. 26, 28, 29, 73).

[46] M. Z. A. Bhotto, S. Makonin and I. V. Bajić, 'Load disaggregation based on aided linear integer programming', *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 7, pp. 792–796, 2016 (cit. on p. 26).

[47] M. Figueiredo, B. Ribeiro and A. de Almeida, 'Electrical signal source separation via nonnegative tensor factorization using on site measurements in a smart home', *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 2, pp. 364–373, 2013 (cit. on p. 26).

[48] M. Wytock and J. Z. Kolter, 'Contextually supervised source separation with application to energy disaggregation', in *Twenty-eighth AAAI conference on artificial intelligence*, 2014 (cit. on p. 27).

[49] H. Gonçalves, A. Ocneanu, M. Bergés and R. Fan, 'Unsupervised disaggregation of appliances using aggregated consumption data', in *The 1st KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, 2011 (cit. on p. 27).

[50] H. Kim, M. Marwah, M. Arlitt, G. Lyon and J. Han, 'Unsupervised disaggregation of low frequency power measurements', in *Proceedings of the 2011 SIAM international conference on data mining*, SIAM, 2011, pp. 747–758 (cit. on p. 27).

[51] S. Pattem, 'Unsupervised disaggregation for non-intrusive load monitoring', in *2012 11th International Conference on Machine Learning and Applications*, IEEE, vol. 2, 2012, pp. 515–520 (cit. on p. 27).

[52] O. Parson, S. Ghosh, M. Weal and A. Rogers, 'An unsupervised training method for non-intrusive appliance load monitoring', *Artificial Intelligence*, vol. 217, pp. 1–19, 2014 (cit. on p. 27).

[53] D. Murray, L. Stankovic, V. Stankovic, S. Lulic and S. Sladojevic, 'Transferability of neural network approaches for low-rate energy disaggregation', in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8330–8334 (cit. on p. 27).

[54] D. Egarter, V. P. Bhuvana and W. Elmenreich, 'Paldi: Online load disaggregation via particle filtering', *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 2, pp. 467–477, 2014 (cit. on p. 27).

[55] J. Liao, G. Elafoudi, L. Stankovic and V. Stankovic, 'Non-intrusive appliance load monitoring using low-resolution smart meter data', in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, 2014, pp. 535–540 (cit. on p. 27).

[56] R. Labbe, 'Kalman and bayesian filters in python', *Chap*, vol. 7, no. 246, p. 4, 2014 (cit. on pp. 30, 65).

[57] G. J. McLachlan, 'Mahalanobis distance', *Resonance*, vol. 4, no. 6, pp. 20–26, 1999 (cit. on p. 30).

[58] *Numpy.correlate*. [Online]. Available: `https://numpy.org/doc/stable/reference/generated/numpy.correlate.html` (cit. on p. 31).

[59] *Template matching*, `https://docs.opencv.org/4.5.2/d4/dc6/tutorial_py_template_matching.html`, Accessed: 2021-08-17 (cit. on p. 31).

[60] S. Makonin and F. Popowich, 'Nonintrusive load monitoring (nilm) performance evaluation', *Energy Efficiency*, vol. 8, no. 4, pp. 809–814, 2015 (cit. on pp. 34, 38, 39).

[61] S. Boyd, S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004 (cit. on pp. 38, 41).

[62] L. Pereira and N. Nunes, 'Performance evaluation in non-intrusive load monitoring: Datasets, metrics, and tools—a review', *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 8, no. 6, e1265, 2018 (cit. on p. 38).

[63] S. Diamond and S. Boyd, 'CVXPY: A Python-embedded modeling language for convex optimization', *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016 (cit. on p. 42).

[64] A. Agrawal, R. Verschueren, S. Diamond and S. Boyd, 'A rewriting system for convex optimization problems', *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018 (cit. on p. 42).

[65] H. Shao, M. Marwah and N. Ramakrishnan, 'A temporal motif mining approach to unsupervised energy disaggregation: Applications to residential and commercial buildings', in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, 2013 (cit. on p. 53).

[66] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011 (cit. on pp. 65, 66, 68).

[67] P. S. Diniz *et al.*, *Adaptive filtering*. Springer, 1997, vol. 4 (cit. on p. 65).

[68] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738 (cit. on pp. 69, 71, 72).

[69] *Python machine learning tutorial*. [Online]. Available: `https://www.python-course.eu/expectation_maximization_and_gaussian_mixture_models.php` (cit. on pp. 71, 72).

[70] H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack problems*. Springer, Berlin, 2003 (cit. on pp. 73, 74).

[71] P. Toth, 'Dynamic programming algorithms for the zero-one knapsack problem', *Computing*, vol. 25, no. 1, pp. 29–45, 1980 (cit. on p. 74).

Appendix A

Additional Methods and Algorithms

This appendix contains a more formal description of some of the methods/algorithms that were used through Chapters 2, 3, and 4. We provide with a short summary of: the Kalman filter, the CUSUM algorithm, the knapsack problem, and Gaussian mixture models.

A.1 The Kalman Filter

The Kalman filter, a recursive estimator, is widely employed in numerous applications, ranging from communication systems, tracking moving targets, guiding and navigation, biomedical signal processing, and change detection. [41, 42, 43]. The Kalman filter belongs to the Bayesian filter family [41], where the states of a dynamical system are sequentially estimated. The system state evolution and measurement processes are both assumed to be Gaussian and linear [41, 66, 67]. The process can be seen as a predict-correct behaviour, where the Kalman filter computes a prediction of the state at the next time step. Later, an error term based on a measurement and the prediction is calculated and corrected. This procedure is repeated iteratively.

On principle, there is a time-varying process that generates a set of outputs corrupted with noise. The noise is generated from two sources: process noise, and the measurement noise. Both noise sources are assumed to be Gaussian with zero mean and independent of each other [41, 56, 66]. Let us consider a state space model in the following form

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{v}_t, \tag{A.1}$$

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{w}_t, \tag{A.2}$$

where $\mathbf{x}_t$ is the $D \times 1$ state vector at time step $t \in \mathbb{Z}^+$, $\mathbf{y}_t$ is the $M \times 1$ measurement vector, $\mathbf{F}_t$ is the $D \times D$ state-transition matrix, $\mathbf{v}_t$ is a $D \times 1$ Gaussian random state noise vector with zero mean and covariance matrix $\mathbf{Q}_t$, $\mathbf{H}_t$ is the $M \times D$ measurement matrix, and $\mathbf{w}_t$ is a $M \times 1$ Gaussian random measurement noise vector with zero mean and covariance matrix $\mathbf{R}_t$. The conditional probability density functions $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ of the state $\mathbf{x}_t$ and measurement vectors $\mathbf{y}_t$, respectively, are expressed in terms of Gaussian probability density functions

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \equiv \mathcal{N}(\mathbf{x}_t \mid \mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{Q}_t), \tag{A.3}$$

$$p(\mathbf{y}_t|\mathbf{x}_t) \equiv \mathcal{N}(\mathbf{y}_t \mid \mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t). \tag{A.4}$$

We assume the initial state distribution $p(\mathbf{x}_0)$ is Gaussian. The posterior probability density function $p(\mathbf{x}_t \mid \mathbf{Y}_t)$ is also Gaussian [66, 41]:

$$p(\mathbf{x}_t|\mathbf{Y}_t) \equiv \mathcal{N}(\mathbf{x}_t \mid \mathbf{m}_{t\,|\,t}, \mathbf{P}_{t\,|\,t}), \tag{A.5}$$

where $\mathbf{Y}_t$ denotes the set of measurements, $\mathbf{Y}_t = \{\mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_t\}$, $\mathbf{m}_{t|t}$ and $\mathbf{P}_{t|t}$ refer the Gaussian posterior mean and covariance at time step t. Moreover, the subscript notation "$t \mid t$" is used to indicate association with the Gaussian probability density function of the state vector $\mathbf{x}_t$ at time step t calculated using the measurements up to time step t. Hence, the Kalman filter boils down to recursively compute the mean $\mathbf{m}_{t\,|\,t}$ and the covariance matrix $\mathbf{P}_{t\,|\,t}$ of the Gaussian posterior probability density function $p(\mathbf{x}_t \mid \mathbf{Y}_t)$ at each time step t:

$$\{\mathbf{m}_{0\,|\,0},\mathbf{P}_{0\,|\,0}\} \xrightarrow{\text{predict}} \{\mathbf{m}_{1\,|\,0},\mathbf{P}_{1\,|\,0}\} \xrightarrow{\text{update}} \{\mathbf{m}_{1\,|\,1},\mathbf{P}_{1\,|\,1}\} \xrightarrow{\text{predict}} \dots\dots \xrightarrow{\text{update}} \{\mathbf{m}_{t\,|\,t},\mathbf{P}_{t\,|\,t}\} \xrightarrow{\text{predict}} \dots \tag{A.6}$$

The Kalman filter equations can be divided in two parts: the prediction and the update. The prediction equations are given by

$$\mathbf{m}_{t\,|\,t-1} = \mathbf{F}_t\mathbf{m}_{t-1\,|\,t-1}, \tag{A.7}$$

$$\mathbf{P}_{t\,|\,t-1} = \mathbf{F}_t\mathbf{P}_{t-1\,|\,t-1}\mathbf{F}_t^\intercal + \mathbf{Q}_t, \tag{A.8}$$

where the expression (A.7) calculates the predicted next step of the system. Further, equation (A.8) computes the covariance matrix by incorporating the process noise variance $\mathbf{Q}_t$.

The update equations can be expressed as

$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t\,|\,t-1} \mathbf{H}_t^\intercal + \mathbf{R}_t, \tag{A.9}$$

$$\mathbf{K}_t = \mathbf{P}_{t\,|\,t-1} \mathbf{H}_t^\intercal \mathbf{S}_t^{-1}, \tag{A.10}$$

$$\mathbf{m}_{t\,|\,t} = \mathbf{m}_{t\,|\,t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t \mathbf{m}_{t\,|\,t-1}), \tag{A.11}$$

$$\mathbf{P}_{t\,|\,t} = \mathbf{P}_{t\,|\,t-1} - \mathbf{K}_t \mathbf{H}_t \mathbf{P}_{t\,|\,t-1}. \tag{A.12}$$

The expression in (A.9) computes the system uncertainty. The Kalman gain (A.10) is a real number between 0 and 1 and it is in charge of partially choosing between the prediction and the measurement. I.e., it chooses either to "believe more in the prediction or in the measurement". Equations (A.11) and (A.12) calculate the state update and the covariance update, respectively.

A.2 Cumulative Sum Algorithm

Before talking about the cumulative sum algorithm, we would like to point out that although the previous section was written for the multi-dimensional Kalman filter and we will reuse some of this theory in the next sections, from this point, we will use the one-dimensional version or the Kalman filter as this is the version which was used throughout this thesis.

Change detection has the goal of determining a change in the mean model. It can be used for detecting abrupt changes in a signal [42]. This is the case of the cumulative sum (CUSUM). The idea is that a signal denoted x_t undergoes an rapid change at time $t = k$, the change is detected and the procedure starts all over again until another change is detected. The parameter time-variations can be modelled using:

$$.x_t = x_{t-1} + v_t \tag{A.13}$$

Here, v_t is the Gaussian random noise variable with variance q_t. It can be observed that this is a special case of the expression in (A.1). The state transition matrix is assumed to be $\mathbf{F}_t = 1$. The core idea is to have a detection method that "fires" an alarm when the signal x_t exceeds a certain threshold h. A convenient method for calculating change in the mean problem is by computing the residual between the measurement y_t and the output of the Kalman filter which we will denote $\hat{x}_{t-1}$. This approach should be robust to variance changes [42]. The residual can be expressed by

$$s_t = y_t - \hat{x}_{t-1}. \tag{A.14}$$

Later, the variable g_t is introduced with the purpose of adding up the inputs s_t until the additions exceed the threshold h. In order to prevent false alarms, one could subtract a small drift term d at each time instant. Moreover, in order to prevent a negative drift, the time to detect a change can be increased. The variable g_t is set equal to 0 each time $g_t < 0$. The CUSUM algorithm can be described as follows

$$g_t = g_{t-1} + s_t - d, \tag{A.15}$$

$$g_t = 0, \text{ and } \hat{k} = t \text{ if } g_t < 0, \tag{A.16}$$

$$g_t = 0, \text{ and } t_a = t \text{ and alarm if } g_t > h > 0, \tag{A.17}$$

where $\hat{k}$ is the estimated change time and t_a is the alarm time.

A.3 The CUSUM Kalman Change Detector

The CUSUM algorithm, with the aid of the Kalman filter, can be used to develop a robust change detector due to the fact that the Kalman filter is the optimal estimator in the least square criterion [66, 42]. Hence, the Kalman filter is in charge of keeping track of the signal variations and the CUSUM algorithm will set an alarm when it observes that the signal has exceeded certain threshold. The combination of these two concepts lead to the CUSUM Kalman filter change detector which is summarised in the following algorithm [1]

[1] It is worth noting that the Algorithm 2 uses a version of the one dimensional Kalman filter. Therefore, all the supporting theory in Section A.1 reduces to scalar variables (i,e,. the vectors and matrices become scalars).

Algorithm 2 CUSUM Kalman Filter Change Detector

for $t = 1, 2, \ldots, T$ **do**
    Initialize the mean and variance of the Gaussian state distribution $\mathcal{N}(x_0|m_{0\,|\,0}, P_{0\,|\,0})$ at time step $t = 0$.
    Compute the prediction state
    ◇ Predict state as $m_{t-1\,|\,t-1} = m_{t-1\,|\,t-1}$
    ◇ Predict the state variance using equation (A.8)
    Compute the estimate $\hat{x}_t = m_{t\,|\,t}$ of the state x_t using the update step
    ◇ Calculate the system uncertainty using equation (A.9)
    ◇ Calculate the Kalman gain using equation (A.10)
    ◇ Update the state using equation (A.11)
    ◇ Update the variance using equation (A.12)
    Calculate the residual $s_t = y_t - \hat{x}_{t-1}$ using equations (A.14) and (A.11)
    Compute g_t using equation (A.15)
    Evaluate if an abrupt change in x_t occurred
    if $g_t > h$ and $g_t > 0$ **then**
        Alarm detected at t_a, record t_a
        $g_t = 0$
    else if $g_t < 0$ **then**
        $g_t = 0$
    end if
end for

A.4 Gaussian Mixture Models

Modelling real datasets with the use of one Gaussian distribution can be a poor representation of the data and suffer from limitations. To work around this problem, a linear superposition of two or more Gaussian distributions could characterise the data more accurately [43, 68]. These linear superposition of Gaussian distributions are denoted mixture distributions or mixture Gaussian model (GMM). The main idea is that one could approximate complex densities by having a sufficient number of Gaussian distributions, the means, covariances, and coefficients in the linear combination of Gaussians have to be adjusted, of course.

Let us suppose that we have a dataset $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the dataset $\mathbf{x}$ consists of N observations and a D dimension. The goal is to partition the data in K different clusters. We can achieve our goal by using Gaussian distributions [68].

A superposition of K Gaussian densities is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{A.18}$$

Each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma_k})$ is called a component of the GMM. Each of the k components has its own mean $\boldsymbol{\mu}_k$ and covariance $\mathbf{\Sigma}_k$. Further, the parameters π_k in (A.18) are known as the mixing coefficients. Integrating the expression in (A.18) by both sides with respect to $p(\mathbf{x})$, the following holds

$$\sum_{k=1}^{K} \pi_k = 1. \tag{A.19}$$

Moreover, considering that $p(\mathbf{x}) \geq 0$ and $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \geq 0$, implies that $\pi_k \geq 0$ for all k. Combining this with equation (A.19), the following condition can be obtained

$$0 \leq \pi_k \leq 1. \tag{A.20}$$

The marginal density can be computed via

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k), \tag{A.21}$$

which is equivalent to the expression in (A.18), where $p(k) = \pi_k$ and can be viewed as the prior probability of picking the k-th component, and $p(\mathbf{x}|k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ is the probability of $\mathbf{x}$ conditioned on k or broadly speaking, the probability that the data $\mathbf{x}$ belongs to the cluster k.

Let us introduce a K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representations, where a particular element z_k is equal to 1 and all other elements equal 0. Therefore, $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. Moreover, the are K possible states for the vector $\mathbf{z}$ according to which element is nonzero.

We can define the probability of $\mathbf{z}$ as

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{A.22}$$

The conditional distribution of $\mathbf{x}$ given a particular value of $\mathbf{z}$ can be written as

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k). \tag{A.23}$$

We can rewrite equation (A.23) as

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{A.24}$$

Thus, the joint probability distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$. The marginal distribution of $\mathbf{x}$ can be obtained by summing the joint distribution over all possible values of $\mathbf{z}$. Which can be calculated as follows

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{A.25}$$

Therefore, the expression (A.25) shows the marginal distribution of $\mathbf{x}$ is a Gaussian mixture. Moreover, if we have more than one observation of $\mathbf{x}$ in the form $\mathbf{x}_1, \ldots, \mathbf{x}_N$, for every data point $\mathbf{x}_n$ there is a corresponding latent variable $\mathbf{z}_n$. The fact that we work with the joint probability $p(\mathbf{x}, \mathbf{z})$ rather than working with the marginal probability $p(\mathbf{x})$, will come very useful when we try to compute the expectation-maximization algorithm (EM) [68, 43].

The conditional probability of $p(\mathbf{z})$ given $p(\mathbf{x})$. The quantity $\gamma(z_k)$ denotes $p(z_k = 1|\mathbf{x})$. This can be computed via

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{A.26}$$

Suppose that there are data observations of the following form $\{\mathbf{x}_1 \ldots \mathbf{x}_N\}$. The goal is to use a mixture of Gaussians to model the data. The data can be represented as a $N \times D$ matrix $\mathbf{X}$, where the n-th row of the matrix $\mathbf{X}$ is given by $\mathbf{x}_n^\intercal$. The hidden or latent variables can be expressed by an $N \times K$ matrix $\mathbf{Z}$ with rows $\mathbf{z}_n^\intercal$. if we assuming the data points are drawn independently from the distribution. The GMM for this (i.i.d.) dataset can be expressed with the log of the likelihood function from the equation in (A.25) given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \tag{A.27}$$

Maximizing the expression in (A.27) in the maximum likelihood framework applied to GMMs is not a trivial task as there is the presence of singularities [68]. Moreover, the summation over k appearing inside the logarithm. Thus the logarithm does not act directly on the Gaussian. A closed form solution will cannot be obtained by setting the derivatives equal to zero. Nevertheless, we can work around this problem by making use of the EM algorithm, which allows us to find maximum likelihood solutions for models with hidden variables [68, 43, 69].

Given a GMM, where the goal is to maximize the likelihood function with respect to the parameters (the means and the covariances of the components and the mixing coefficients), the EM algorithm can be expressed as follows:

Algorithm 3 The EM for Gaussian Mixtures

1: Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k. Evaluate the initial value of the log likelihood.

2: **Expectation step (E step).** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{A.28}$$

3: **Maximization step (M step)**. Re-compute the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{A.29}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\intercal} \tag{A.30}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{A.31}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{A.32}$$

4: Evaluate the log likelihood and check if the parameters or the log likelihood have converged. If not, return to step 2.

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \tag{A.33}$$

In simple words, the expectation step (E-step) calculates for each data-point $\mathbf{x}_n$, the probability $\gamma(z_{nk})$, which can be roughly interpreted as the fraction of the probability that $\mathbf{x}_n$ belongs to the cluster k over the probability of $\mathbf{x}_n$ over all clusters. On the other hand, the maximization step (M-step) updates the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k. Where $\gamma(z_{nk})$ can be loosely interpreted as the fraction of points allocated per cluster [68, 69].

A.5 The Knapsack Problem

The knapsack problem (KP), an optimization problem, consists of the analogy of having a knapsack or backpack with a with a limited weight capacity c. Furthermore, there is a set of N items that one can choose to pack or not in the knapsack $x_i \in \{0, 1\}$. Each of the

items have an associated weight w_i and profit p_i. The goal is to choose which item/s to pack by maximizing the profit without exceeding the backpack capacity [70].

Mathematically, the KP can be formulated as the following linear integer programming formulation:

$$\text{maximize} \sum_{i=1}^{n} p_i x_i \tag{A.34}$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq c, \tag{A.35}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n. \tag{A.36}$$

The goal is to find the binary coefficients x_i that maximize the objective function in (A.34), given the constraints in (A.35) and (A.36). We will denote the *optimal solution vector* by $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ and the *optimal solution* by z^* The most intuitive method to solve the problem would be to consider profit the weight ratio and try to pack the items with highest efficiency into the knapsack. This is referred to the greedy method or greedy algorithm [45, 70]. The main issue with the greedy algorithm is that it does not guarantee to provide with the optimal solution.

A.5.1 Solving KP with Dynamic Programming

As it was previously mentioned, the greedy method for solving the KP problem may not provide the optimal solution. Moreover, the solution can be far away from the optimal solution [70]. Therefore, a tool which allow us to solve the KP with a more suitable solution is needed. Dynamic programming is a general approach and it is broadly used in many areas of operations research. The reason why dynamic programming can be applied to the KP is due to the optimal solution consists of a combination of optimal solutions to sub-problems.

Let us consider the optimal solution for the KP. It is clear that removing any item r from the optimal knapsack packing, the remaining solution set must be an optimal solution to the sub-problem with capacity $c - w_r$ and item set $N - \{r\}$. Having this property, which is called an *optimal substructure*, we can do the following: Let us assume that the optimal solution for the KP has already been calculated for a subset of items and all capacities up to c. Later, one item is added to the subset and check if the optimal solution has to change when the subset is larger. Using the solutions with smaller capacity is now the method to inspect, if the optimal solution has to be modified. The computation of the possible change of the optimal solution needs to be performed again for all the capacities. The procedure of

adding an item to the knapsack is repeated over and over until all the items were considered. Hence, the overall optimal solution is found [70, 71].

We can describe the following sub-problem which consists of the item set $\{1, 2, \ldots, i\}$ and a knapsack capacity $d \leq c$. For $i = 1, 2, \ldots n$ and $d = 0, 1, \ldots, c$ let us define $(\mathrm{KP}_i(d))$ as

$$(\mathrm{KP}_i(d)) \qquad \text{maximize} \sum_{l=1}^{i} p_l x_l \tag{A.37}$$

$$\text{subject to} \sum_{l=1}^{i} w_l x_l \leq d, \tag{A.38}$$

$$x_l \in \{0, 1\}, \quad l = 1, \ldots, i, \tag{A.39}$$

with optimal solution value $z_i(d)$.

If we know $z_{i-1}(d)$ for all capacity values $d = 0, 1, \ldots c$, then the additional item i can be considered and calculate the corresponding solution $z_i(d)$ via the recursive formula

$$z_i(d) = \begin{cases} z_{i-1}(d) & \text{if } d < w_i, \\ \max\{z_{i-1}(d), z_{i-1}(d - w_i + p_i)\} & \text{if } d \geq w_i. \end{cases} \tag{A.40}$$

The dynamic programming for solving the KP is shown in Algorithm 4.

Algorithm 4 The Dynamic Programming Algorithm for KP

```
1: for d = 1, 2, ..., c do
2:     Initialize z_0, z_0(d) = 0
3: end for
4: for i = 1, 2, ..., n do
5:     for d = 0, 1, ..., w_{i-1} do
6:         z_i(d) = z_{i-1}(d)
7:     end for
8:     for d = w_i, w_{i+1}, ..., c do
9:         if z_{i-1}(d - w_i) + p_i then
10:            z_i(d) = z_{i-1}(d - w_i) + p_i
11:        else z_i(d) = z_{i-1}(d)
12:        end if
13:    end for
14: end for
15: z* = z_n(c)
```