Non-Parametric Modeling in Non-Intrusive Load Monitoring

by

Richard Jones

B.S. (Hons) in Physics, University of Manitoba (2018)
B.A. in Psychology/Philosophy, University of Manitoba (2014)

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 2020

Declaration of Committee

Name:	**Richard Jones**
Degree:	**Master of Applied Science**
Thesis title:	**Non-Parametric Modeling in Non-Intrusive Load Monitoring**
Committee:	**Chair:** Ljiljana Trajkovic Professor, Engineering Science
	Stephen Makonin Co-Supervisor Adjunct Professor, Engineering Science
	Ivan Bajić Co-Supervisor Professor, Engineering Science
	Rodney Vaughan Committee Member Professor, Engineering Science
	Daniel Lee Examiner Professor, Engineering Science

Abstract

Non-Intrusive Load Monitoring (NILM) is an approach to the increasingly important task of residential energy analytics. NILM aims to provide appliance-level monitoring using only the available aggregate data, side-stepping the need for expensive and intrusive monitoring equipment. The present work showcases two self-contained, fully unsupervised NILM solutions: the first featuring non-parametric mixture models, and the second featuring non-parametric factorial Hidden Markov Models with explicit duration distributions. In the latter, the use of traditional and novel constraints during inference shows marked improvement in disaggregation accuracy with very little effect on computational cost, relative to the motivating work. A novel denoising filter is developed to aid in change-point detection and clustering accuracy. To constitute a complete unsupervised solution, labels are applied to the inferred components using a Res-Net-based deep learning architecture. Finally, the concept of Bayesian surprise is explored to monitor data-novelty and potentially regularize learning in these and other NILM methods.

Keywords: non-intrusive load monitoring; non-parametric; unsupervised; mixture model; factorial; hidden Markov model

Notes

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

- Steady-state block filter: Section 4.1. Published [1].
- Res-Net-based state combination and labeling: Section 4.3. Submitted and awaiting final decision [2].
- Bayesian Surprise: Sections 2.3.2, 4.5, 5.5. Submitted, accepted for publication [3].

Acknowledgements

My heartfelt thanks to my supervisors Stephen and Ivan. Giving me the freedom to explore what I found interesting was a daunting and much appreciated push in the right direction, both in my studies and in my life as a whole. It allowed me to see value in my instincts and build some trust in my abilities to figure things out. More than anything else, I think these skills will in retrospect prove invaluable.

To my colleagues and friends, Alejandro and Alon: I couldn't have done it without you guys. Often in life I think we search for validation of our choices; something to indicate that we made the right decision. After starting this program, I didn't have to look any further than you guys. Alon's design and patient explanations of the labeling architecture in Section 4.3 is also gratefully acknowledged.

All my thanks and love to my friends, family, and most of all my partner, Leah. Nothing can be said here to express the love and appreciation I have for all that you do, and all that you are. I truly couldn't have done this without you.

Contents

List of Tables

List of Figures

Acronyms

CNNs *Convolutional Neural Networks*. 43

DNN *Deep Neural Network*. 61

DP *Dirichlet Process*. 12

ELBO *Evidence Lower Bound*. 30

EM *Expectation-Maximization*. 8

FFT *Fast-Fourier Transform*. 51

GAN *Generative Adversarial Network*. 51

GEM *Griffiths-Engen-McCloskey*. 12

GMM *Gaussian Mixture Model*. 3

HDP *Hierarchical Dirichlet Process*. 3

HMM *Hidden Markov Model*. 5

HSMM *Hidden Semi-Markov Model*. 3

i.i.d. *independent and identically distributed*. 12

KL *Kullback-Leibler*. 29

LRF *Local Receptive Field*. 44

LSTM *Long Short-Term Memory*. 51

MAP *Maximum A-Posteriori*. 20

MCMC *Markov Chain Monte Carlo*. 14

MF *Mean-Field*. 31

MH *Metropolis-Hastings*. 23

NILM *Non-Intrusive Load Monitoring*. 1

ReLU *Rectified Linear Unit*. 45

RMS *Root-Mean-Square*. 41

RMSE *Root-Mean-Square Error*. 4

RNNs *Recurrent Neural Networks*. 43

SSHMM *Super-State Hidden Markov Model*. 49

VAE *Variational Auto-Encoder*. 51

Chapter 1

Introduction

1.1 The Non-Intrusive Load Monitoring Problem

Non-Intrusive Load Monitoring (NILM) is a field of research focused on developing algorithms that can accurately track constituent electrical loads in a system using only the aggregate signal. This level of information has potential impacts on consumers, distributors, and producers. Studies suggest that even superficial feedback for end-user consumption leads to reduced usage and more energy-conscious consumers [4]. Improved load prediction for both energy producers as well as grid operators can lead to higher efficiency production and optimized participation with transactive energy markets [5]. Greater transparency of consumer usage behaviours and the associated loads in their homes allows a breadth of information to inform load prediction. Utilities can also leverage this information to devise public information campaigns, expansion plans, retrofit programs, etc. Finally, with the increased adoption of electric vehicles and solar combined with in-home energy storage, there is emerging evidence that using these nodes to contribute where necessary to the grid can improve stability and allow for significant penetration of renewables into the grid [6]. Transparency of such grid resources to operators is highly valuable for these ambitious ideas. Although the bulk of NILM research is largely residential in focus, the promise of NILM at the commercial scale brings many obvious benefits to efficiency and profit margins. However, the increased number and complexity of constituent loads in these contexts, as well as the lack of public datasets, suggests that NILM at the commercial scale may not be realized.

Disaggregation of constituent loads in a system is a difficult and complex problem, owing to the fact that electrical loads in a home often significantly overlap in time or are even concurrent with one another [7]. The problem can be stated as solving for each P_i in

$$P_{tot} = \sum_{i \in M} P_i + \epsilon, \tag{1.1}$$

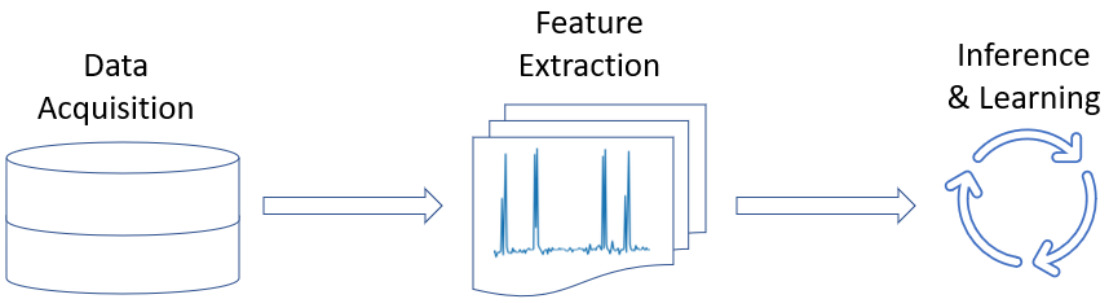

Figure 1.1: General NILM process, adapted from [8].

where each P_i is the power consumption associated with appliance i out of a set of M total appliances, and ϵ is some measurement noise. Given the large, generally unknown, number of appliances in a modern home, estimating every P_i is neither feasible nor necessary. As a result, M is often truncated to a subset of desired, high-demand appliances, M'. The remaining appliances are treated as noise and grouped with the measurement noise:

$$P_{tot} = \sum_{i \in M'} P_i + \tilde{\epsilon}, \tag{1.2}$$

where $\tilde{\epsilon} = \epsilon + \sum_{i \notin M'} P_i$. Many statistical models and algorithms have been developed to solve the problem of estimating each $\hat{P}_i$, some of which will be discussed in Chapter 3. Although present approaches to NILM vary widely, in the general case the process follows Figure 1.1. Data is obtained either online or offline, processed for the relevant features needed for load identification, and used for inference of each $\hat{P}_i$ and for refinement of learned models. Significant results have been achieved in the supervised-learning context, where algorithms are fed labeled data from individual loads in order to learn characterizing features. The hope with these supervised approaches is that with sufficiently representative datasets and regularized learning, generalization to unseen homes will approach feasibility in real use-cases. As it currently stands, the available datasets for NILM research are limited relative to the diversity of appliance types and models. Unsupervised approaches to NILM, which involve no labeled training data, are therefore a viable option. In unsupervised learning, NILM can be broken up into two sub-problems: disaggregation and labelling. These can be and often are addressed simultaneously, such as by setting priors for appliance models in Bayesian settings, and allowing the data to be assigned to the appropriate model and update the relevant parameters. In this way, *pre-labeled* prior information is chosen as general as possible and used to segregate house-specific data while updating the prior information. The assumption inherent in this approach is that averages over models for a given appliance type are sufficiently distinct in a given representation from other appliance types. This claim is difficult to verify in general given the poor representation of appliance

models in existing datasets. Overlap of these appliance-model averages in a given feature space can lead to poor classification accuracy, which is especially troublesome in methods such as [9], the motivating work for this thesis.

An alternative formulation is to avoid pre-labelling, and instead use the house-specific data to disaggregate appliances before applying labels to the resulting disaggregated signals. By doing this, we are implicitly admitting that these model-averaged, "general" priors are either unknown, or known only to a degree of uncertainty that overlap is impossible to avoid. We instead assume that differences between particular models of two different appliance types will be far more distinct on average than the model-averaged differences between those appliance types. We should therefore leverage these house-specific differences to disaggregate appliance signals from one another, rather than relying on weakly informative priors. After disaggregation, we can apply labels using deep learning methods that establish highly non-linear combinations of features to distinguish between appliance types. This motivates the separated disaggregation-labelling approach taken in this work.

1.2 Thesis Preview

The work contained in this thesis is an attempt to solve the NILM problem in an unsupervised way, broken down into what are two self-contained NILM solutions. First, a non-parametric *Gaussian Mixture Model* (GMM) is used to establish house-specific priors over appliance steady-state power values and state durations, leveraging transient behaviour to assist state assignments. An edge-pairing method using various constraints establishes unlabeled disaggregated signals. This is discussed in Sections 4.2 and 5.2. By itself, this method provides promising disaggregation performance when tested on RAE [10] and REFIT [11] homes. Once a labelling method, whether deep learning as in this thesis or otherwise, can be shown to reliably classify the disaggregated traces, this method may prove to be highly useful for a subset of appliances.

The disaggregated components from this mixture model method can then be merged if necessary and labeled using a deep learning approach. The Res-Net-based approach in this thesis is outlined in Sections 4.3 and 5.3. Although further development is required before this approach can reliably label unknown appliances, it shows promise with disaggregated components from an unseen house in the dataset used for training the network. This suggests that using a training set that is regionally similar to the target home could improve labelling reliability.

The second solution leverages the learned mixture model distributions over demand and duration to inform the priors for a non-parametric factorial hidden Markov model with explicit state duration distributions, called a Hierarchical Dirichlet Process HDP Hidden Semi-Markov Model HSMM [9]. With pre-labeled components from the previous step, this method can also be considered a complete NILM solution. This method is outlined

in Sections 4.4 and 5.4, and is the main focus of this work. Nearly all major appliances are disaggregated more accurately in terms of F1-score (equation 2.120) with this method when compared to the mixture model method. Furthermore, novel constraints on inference in this complex model result in much fewer iterations necessary to reach convergence in model performance. Relative to unconstrained inference, the imposed constraints result in over 20% higher disaggregation accuracy (equation 2.116) for the same number of iterations. Finally, a method to monitor novel events in the aggregate signal is introduced under the framework of Bayesian surprise [12]. This can be used to maintain flexibility in these and other NILM algorithms when appliances abruptly change in the home, along with potential applications in regularization against overfitting, fault detection, and dataset acquisition and usage. A threshold in terms of postdictive and transitional surprise (discussed in Sections 4.5 and 5.5) is demonstrated by the decay in performance improvements when training with additional same-house data. This threshold shows usefulness in cross-house generalization relative to full training, showing the promise of this method to regularize learning in NILM. Finally, the transitional surprise metric developed in Section 4.5 is shown to be useful in the context of a popular Super-State Hidden Markov Model [13], where model performance improvement follows closely the decay in transitional surprise for increasing dataset size.

All of the above components make use of a simple yet robust signal filter designed to extract steady-state information from aggregate data. The filter removes transient behaviour during appliance activation and measurement noise by imputing the mean signal value between change-points. These change-points are determined by using non-linear demand-dependent and edge-dependent thresholds. The filter is shown to far outperform an existing method both in terms of run-time as well as accuracy of imputed steady-state values. An optimization scheme is developed by regularizing *Root-Mean-Square Error* (RMSE) loss (equation 2.117) with a weighted Hoyer sparsity loss (equation 4.5), and using a dual-annealing stochastic optimization. This approach shows promise for developing an adaptive filter that can be optimized to any particular use-case. This steady-state block filter is discussed in Sections 4.1 and 5.1.

Chapter 2 is devoted to the theoretical details relevant to the above components, including the basics of Hidden Markov Models, their extension into the non-parametric regime, mixture models, stochastic and variational inference, and also a comprehensive introduction to deep learning and the more recent advances in the field relevant to this thesis. Chapter 3 will briefly outline recent developments in NILM research more broadly as well as those of particular relevance to the present work. Chapter 4 will discuss the implementation methods and considerations in each component of the final solution, while Chapter 5 presents their respective performance. Finally, Chapter 6 provides a summary and highlights motivations for further work.

Chapter 2

Background

2.1 Hidden Markov Model Basics

The *Hidden Markov Model* (HMM) is an extension of a *Markov chain*, which is a stochastic model named after Russian mathematician Andrey Markov [14]. Markov chains by definition obey—or are approximated as obeying—the 'Markov property', whereby the prediction of the current state of a system only requires the previous state and any associated observations. These states can be considered random variables $X_1, X_2, \ldots, X_n$, which take on observable values $x_1, x_2, \ldots, x_n \in \mathcal{X}$ (where $\mathcal{X}$ is some measure space), and are said to constitute a (discrete-time) Markov chain if

$$X_t \perp \{X_1, X_2, \ldots, X_{t-2}\} | X_{t-1}, \tag{2.1}$$

where $\perp$ indicates conditional independence. In other words, the joint distribution over the n random variables factors in the following way:

$$p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2)\ldots p(x_n|x_{n-1}), \tag{2.2}$$

and since we can always write

$$p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\ldots p(x_n|x_1, x_2, \ldots, x_{n-1}), \tag{2.3}$$

we can only have equality between the two above expressions if, for all t,

$$p(x_t|x_1, x_2, \ldots, x_{t-1}) = p(x_t|x_{t-1}). \tag{2.4}$$

This requires the statement of conditional orthogonality in equation 2.1. This corresponds to what is sometimes called the *memoryless property*, since the future state of such a system, say at time t, depends only on the state at time $t-1$, and given knowledge of that state, all other past states are independent of the one at t. One simple example of a Markov chain process in the world of finance is shown in Figure 2.1, where market behaviour in the

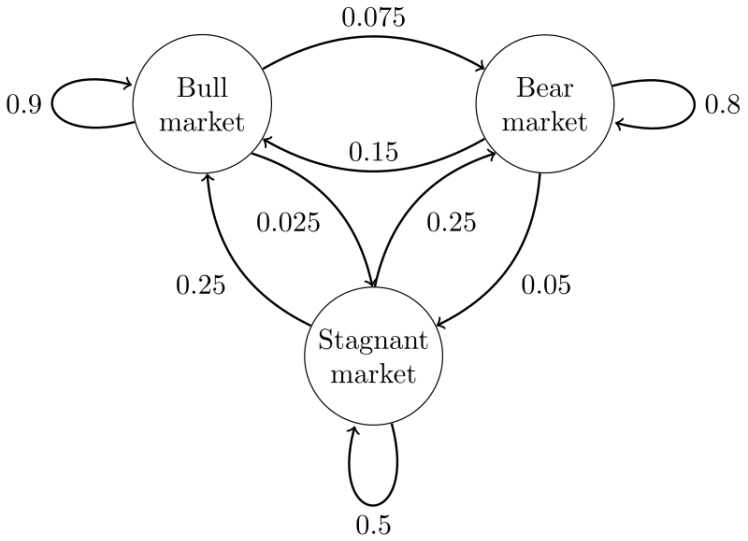

Figure 2.1: Example of a Markov process. Numbers indicate probability of future year behaviour. Image by Gareth Jones, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=26414872

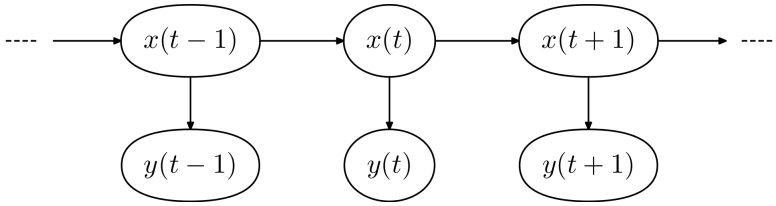

Figure 2.2: Example of a Hidden Markov Model. *(Creative commons license)*

following year (head of arrows) can be predicted with varying probability based only on the current year's market (tail of arrows).

2.1.1 Hidden Markov Models

In HMMs, the internal state of a system is unobservable, and tractable inference must be made based on related observables. In other words, there are observations made on the system that are related to the possible internal states but not uniquely so. In addition, the states are also temporally related to one another but not uniquely so. Let's denote the hidden random variables as $X_1, X_2, \ldots, X_n$, each of which take on values in a discrete set $\{1, 2, \ldots, m\}$. Each hidden state emits some observable random variable $Y_1, Y_2, \ldots, Y_n$, whose realizations give us our data set $D = (y_1, y_2, \ldots, y_n)$. The graphical model for this type of Markov process is shown in Figure 2.2. This figure depicts the memoryless property of Markov chains (see equation 2.1). It also demonstrates a useful property of directional, acyclic graphical models called *d-separation* or *dependence-separation*, which will be referenced several times [15]. The definition of d-separation is as follows:

Let A, B, and C be disjoint subsets of vertices of a graph, G. A path between two vertices is called *blocked* with respect to C if it passes through a vertex in the graph, v, such that either:

1. the directional arrows are head-to-tail or tail-to tail with respect to C and $v \in C$; or
2. the directional arrows are head-to-head and $v \notin C$ and none of the descendants of v are in C (where 'descendants' refers to the directionally subsequent vertices following v).

If all paths between a vertex of A and a vertex of B are blocked with respect to C, then A and B are said to be d-separated by C, which then ensures that $A \perp B|C$.

By examining Figure 2.2, we see that each observable state Y_k is d-separated from all $X_{j<k}$ and $Y_{j<k}$, and so we have the conditional independence

$$Y_k \perp Y_{j<k}, X_{j<k}|X_k,$$

allowing the joint distribution over observations and hidden states in HMMs to be written as

$$p(y_1, \ldots, y_n, x_1, \ldots, x_n) = p(x_1)p(y_1|x_1) \prod_{k=2}^{n} p(x_k|x_{k-1})p(y_k|x_k), \tag{2.5}$$

where by convention the above notation implies $p(y_1|x_1) \equiv p(Y_1 = y_1|X_1 = x_1)$. We can now introduce the simplifying notation to be used hereafter. Since $p(x_k|x_{k-1})$ intuitively represents the probability of the system evolving to state x_k from state x_{k-1}, we call these the *transition probabilities*, and denote the transition from state i to state j as

$$\pi_{i,j} \stackrel{\text{def}}{=} p(x_{k+1} = j|x_k = i), \ \forall \ i, j \in \{1, 2, \ldots, m\} \tag{2.6}$$

All such transitions can then be organized in an $m \times m$ transition matrix, π, with the $(i, j)^{th}$ entry as shown above.

Additionally, since $p(y_1|x_1)$ is the probability of observing y_1 conditioned on the system being in state x_1, we call these the *emission probabilities*, and we can construct a distribution on $\mathcal{Y}$, the measure space for all Y:

$$\vartheta_i(y) \stackrel{\text{def}}{=} p(y|x_k = i) = F(\boldsymbol{\theta}_i) \tag{2.7}$$

where $F(\cdot)$ is some indexed family of distributions (e.g., Gaussian distributions for which $\boldsymbol{\theta}_i$ parameterizes state i). If we define the initial state probability $\pi_0(i) \stackrel{\text{def}}{=} p(x_1 = i)$, this allows us to write equation 2.5 in a more intuitive form:

$$p(y_1, \ldots, y_n, x_1, \ldots, x_n) = \pi_0(i)\vartheta_{x_1}(y_1) \prod_{k=2}^{n} \pi_{x_{k-1},x_k}\vartheta_{x_k}(y_k) \tag{2.8}$$

Given enough training data, distributions over initial state probabilities as well as transition and emission probabilities can be given arbitrary form in a frequentist setting by counting the relevant occurrences. Alternatively, these distributions can be given explicit parametric forms, where the associated parameters can be estimated and fixed by, for example, *Expectation-Maximization* (EM). The most flexible approach, however, is to allow the data to update the beliefs about the parametric form of these distributions in a fully Bayesian way.

Bayes' Theorem can be stated for evidence $\mathbf{x}$ and model parameters $\boldsymbol{\theta}$ as follows:

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{2.9}$$

In words, this states that the *posterior* distribution over model parameters given new evidence is the normalized product of the *likelihood* of the evidence under the model and the *prior* probability of the parameters. This allows estimation of the best current value of $\boldsymbol{\theta}$, or the posterior $p(\boldsymbol{\theta}|\mathbf{x})$ can be used as the prior distribution given new observations $\mathbf{x}'$. The normalization constant, called the *marginal likelihood* of the data, is difficult to compute in the general case. However, when the prior and posterior distributions are within the same *parametric family*, the prior is said to be *conjugate* to the likelihood distribution, $p(\mathbf{x}|\boldsymbol{\theta})$. Conjugacy is an important property that allows relatively simple update equations to be derived for conjugate pairs, rather than needing to compute the marginal likelihood for every iteration of Bayes' Theorem. For a description of common conjugate pairs as well as derivations of the update equations for the respective model parameters, see [16].

As an example, a common method to model the emissions in HMMs is with simple Gaussian distributions, $p(y|x_i) = \mathcal{N}(\boldsymbol{\mu_i}, \Sigma_i)$, with mean vector $\boldsymbol{\mu}_i$ and covariance matrix Σ_i. In the general case, both the component means and their associated covariances are unknown, and so the conjugate distribution to the Gaussian likelihood in this case is the normal-inverse-Wishart (or, parameterized slightly differently, the normal-inverse-Chi-squared) joint prior [16]. For each component in the model, this joint prior is described by

$$\begin{aligned} \Sigma &\sim \text{IW}(\nu, \Delta) \\ \boldsymbol{\mu}|\Sigma &\sim \mathcal{N}(\phi, \Sigma/\kappa), \end{aligned} \tag{2.10}$$

where IW is the inverse-Wishart distribution with covariance/scale matrix Δ and degrees of freedom ν. Similarly, the conditionally normal *hyperprior* on the component means is parameterized by a base mean, ϕ, and covariance scaled by another *hyperparameter*, κ. These hyperpriors are shared across components and their associated hyperparameters determine the form for the prior distributions specific to each component.

In the Bayesian framework, the transition matrix needs a prior as well, for which certain conditions must be met:

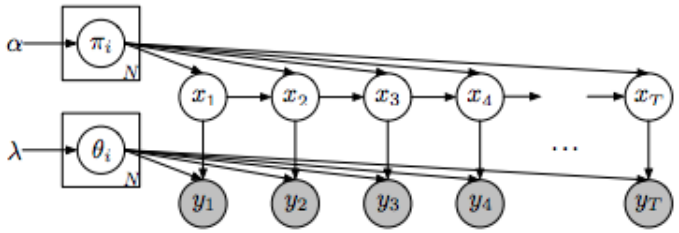

Figure 2.3: Bayesian extension of a traditional HMM. Taken from [9].

1. For each $\pi_{i,j}$, we must have that $0 \leq \pi_{i,j} \leq 1$;
2. For each row $\boldsymbol{\pi}_i$, we must have that $\sum_j \pi_{i,j} = 1$.

For a system with cardinality N (i.e., N components/states), a good candidate for a prior with a support satisfying these constraints is the Dirichlet distribution of dimension N. The Dirichlet distribution is a multivariate generalization of the Beta distribution, and is a method to generate distributions of varying similarity to a source/base distribution. The Dirichlet distribution can be thought of as a distribution over distributions. It is parameterized by a vector of positive real numbers $\boldsymbol{\alpha}$, which determine the concentration along each dimension of the simplex (generalization of a triangle) defined by its support. Useful for later discussions, the probability density function of each transition row is distributed as Dirichlet, given up to a normalization constant by

$$p(\boldsymbol{\pi}_i|\boldsymbol{\alpha}) \propto \prod_{j=1}^{K} \pi_{i,j}^{\alpha_j - 1}. \tag{2.11}$$

Furthermore, the hidden states in the HMM are distributed categorically, meaning that at a given instant the system can evolve to any of the available discrete states, with the probability of transition defined by $\pi_{i,j}$. Consequently, the Dirichlet prior is a natural choice since it is conjugate to a categorical likelihood distribution.

The generative model of the Bayesian HMM with cardinality N can be expressed in block form according to Figure 2.3, or alternatively as the following process:

For $i = 1, 2, \ldots, K$:

$$\begin{aligned} \boldsymbol{\pi}_i &\overset{iid}{\sim} \text{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_N) \\ \boldsymbol{\theta}_i &\overset{iid}{\sim} H \end{aligned} \tag{2.12}$$

For $t = 1, 2, \ldots, T$:

$$\begin{aligned} x_t &\sim \boldsymbol{\pi}_{x_{t-1}} \\ y_t &\sim F(\boldsymbol{\theta}_{x_t}), \end{aligned} \tag{2.13}$$

where $\mathrm{Dir}(\cdot)$ indicates the Dirichlet distribution, the α_i's are the "pseudo-count" prior observations of the i^{th} component. Typically the prior is symmetric such that $\alpha_1 = \alpha_2 = \ldots = \alpha_N = \alpha$, although these can be adjusted to account for prior knowledge, for example appliance activation sparsity. Again, the $\boldsymbol{\theta}_i$'s parameterize the observation distribution F from which observations y_t are drawn, themselves drawn from a base observation distribution H. This conjugate joint prior was the normal-inverse-Wishart distribution in the example above.

Despite the flexibility of Bayesian modelling in HMMs, a major disadvantage of traditional HMMs is the duration distributions they implicitly impose on the hidden states. For a hidden state k with self-transition probability π_{kk}, the duration of state k is by definition distributed as a geometric with mean $1 - \pi_{kk}$. This is simply a consequence of the Markovian assumption—that the current state of the system depends only on the previous one. Unfortunately, this is often an unphysical model for state durations, and so the HSMM was introduced [17]. These more sophisticated models allow explicit distributions on state duration, which removes the strict Markovian assumption of memorylessness.

2.1.2 Hidden Semi-Markov Models

Although several variants exist, the general idea behind HSMMs is to expand the generative process of HMMs to include an explicit duration distribution for each state. Instead of allowing some probability of self-transition as in HMMs (which was exactly what imposed the geometric duration distribution), HSMMs prohibit self-transition by setting all $\pi_{ii} = 0$ by the following transform:

$$\pi_{ij} \leftarrow \bar{\pi}_{ij} \stackrel{\text{def}}{=} \frac{\pi_{ij}}{(1 - \pi_{ii})}(1 - \delta_{i,j}), \tag{2.14}$$

where $\delta_{i,j}$ is the Kronecker delta such that $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ otherwise.

The system then samples a random state duration from the specified distribution when conditioning on a particular hidden state, and remains in that state until the sampled duration expires. In contrast to the traditional Markovian trellis structure discussed in the previous section, the so-called *super-states* of the system, z, are now emitting random length observation sequences, shown in Figure 2.4.[1] These state-specific duration distributions can

[1]Important to note is that these are distinct from super-states of the same name in [13], which is a vector describing the activation state of all appliances in the home. Intended meaning will be clear in the text.

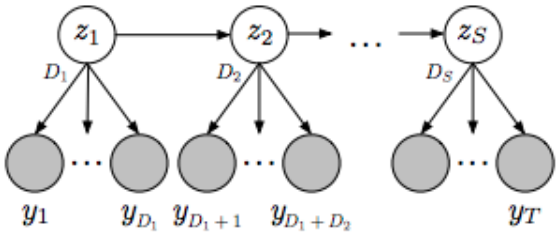

Figure 2.4: Bayesian extension of a traditional HMM. Taken from [9].

be quite complex, for example a weighted sum of Poisson distributions. The improvement in modelling explicit duration distributions predictably comes at computational cost. However, there are several ways to significantly reduce the model complexity, which will be discussed in some detail later.

Once a model is instantiated, we can answer the usual inference questions, such as the duration prior term for a segment s beginning at time $t+1$, conditioned on the hidden state: $p(D_{t+1} = d|x_{t+1} = i)$, or the likelihood term: $p(y_{t+1:t+d}|x_{t+1} = i,\ D_{t+1} = d)$, and many others. The types of questions that can be asked in these models and the inference structures to answer them will be covered in Section 2.3.

Another major disadvantage to the Bayesian HMM and its semi-Markov extension is that the cardinality must be determined *a priori*. Instead, we ideally want to allow for an infinite-dimensional state-space which dynamically expands or contracts as necessary. Section 2.3.2 briefly discusses certain stochastic inference algorithms that allow flexible dimensionality, but an arguably simpler method is to extend our models to the regime of *Bayesian Non-parametrics*.

2.2 Bayesian Non-parametrics

Bayesian non-parametrics is a class of models which are not constrained to finite parameterizations, and allow a dynamic, potentially infinite-dimensional state space that we can extend to HMMs and HSMMs. In this section, some basic structure and properties of Bayesian non-parametrics are explored, followed by an application to Mixture Models to be used later in the context of Bayesian surprise. Finally, we extended this regime to HMMs and HSMMs into HDP-HMMs/HSMMs.

2.2.1 Basics

In the context of HMMs, we know that to accommodate an infinite state space, at the very least the transition matrix must itself be potentially infinite. However, the same previous conditions for the transition matrix must still hold: with probability one we must transition

from a given state to some other state. So our prior on the transition rows must be able to generate an infinite sequence of probabilities summing to one. In the finite case, we saw that a Dirichlet prior on the finite number of states was conjugate to a categorical likelihood of transition. A natural step, then, is to explore an infinite-dimensional analogue of the Dirichlet distribution. Such an object, called the *Dirichlet Process* (DP) was discovered by Thomas Ferguson in the early 1970s [18].

The DP is a stochastic process that generates random probability measures, which follow a Dirichlet distribution for every finite partition of some measurable space [19]. It is uniquely defined by a base measure on the measurable space and a concentration parameter, similar to the finite-dimensional Dirichlet distribution. An intuitive picture for the choice of base distribution is the *stick-breaking process* [20]. In the stick-breaking framework, the infinite sequence of transition probabilities for each row are generated by drawing from a *Griffiths-Engen-McCloskey* (GEM) distribution, described by

$$\begin{aligned} \nu_i|\alpha &\sim \text{Beta}(1,\alpha), \quad i \in \{1,2,\ldots\} \\ \pi_i &= \nu_i \prod_{\ell=1}^{i-1}(1-\nu_\ell), \quad i = \{1,2,\ldots\}, \end{aligned} \tag{2.15}$$

where, when unbolded, π_i is understood to be a scalar quantity.[2] This process can be understood by imagining a unit probability stick being partitioned into sequentially smaller pieces. The proportion of the remaining stick to be broken off is sampled according to a beta distribution parameterized by $(1, \alpha)$.

A draw from the DP (i.e., $G \sim DP(\alpha, G_0)$) is a discrete, infinite random object that can be expressed by

$$G = \sum_{i=1}^{\infty} \pi_i \delta_{\boldsymbol{\theta}_i}, \tag{2.16}$$

where $\boldsymbol{\theta}_i$ is the i^{th} of the countably infinite atoms drawn *independent and identically distributed* (i.i.d.) from a base distribution, G_0, and $\pi_i \delta_{\boldsymbol{\theta}_i}$ is the atom located at $\boldsymbol{\theta}_i$, scaled by weight π_i. That is,

$$\boldsymbol{\theta}_i | G_0 \overset{i.i.d.}{\sim} G_0. \tag{2.17}$$

An important property of the DP is its *preferential attachment* of observations to existing states, encouraging parsimonious state assignments/clusterings. It can be shown that the number of clusters instantiated over N draws from the measure π defined by $\text{DP}(\alpha, G_0)$

[2]This notation is useful to draw an analogy between this constructive process and the transition row objects, which will become clear in Section 2.2.3

scales as $\alpha \log(N)$ as $N \to \infty$ [21]. The predictive distribution on these partitions of the data induced by the DP can be explored using the *Chinese Restaurant Process* analogy. In it, each of the infinite atoms $\boldsymbol{\theta}_i'$ drawn from G_0 is pictured as a customer entering a Chinese restaurant. In this restaurant, each table serves a unique dish θ_k, and each customer either sits at an existing table or starts a new table that was previously unoccupied. It is helpful to introduce an indicator variable z_i which designates the table assignment for atom $\boldsymbol{\theta}_i'$ (i.e. if $z_i = m$, then $\boldsymbol{\theta}_i' = \theta_m$).

Examination of the distribution of the atoms under marginalization of the base measure G_0 in the *Pólya urn* picture (see [19]) shows that the predictive distribution of the indicator variables take the following form:

$$p(z_{N+1} = z | z_1, \ldots, z_N, \alpha) = \frac{\alpha}{N + \alpha} \delta_{z,K+1} + \frac{1}{N + \alpha} \sum_{k=1}^{K} \delta_{z,k}, \tag{2.18}$$

where N_k is the occupation number or cardinality of the table k, and $K + 1$ is a new, previously unoccupied table. In other words, the probability of seating a customer at an existing table is proportional to the number of customers already seated at that table. The proportionality constant—as well as the probability of assigning a customer to a new table—is determined by the same α parameter we saw in the stick-breaking process. In this picture it becomes clear why the α parameter is called the concentration parameter.

This description of the DP lends itself most transparently to mixture models, which allows non-parametric clustering of data into a flexible number of components. The following provides an introduction into parametric mixture models as well as their extension into non-parametrics.

2.2.2 Mixture Models

In a typical parametric GMM with K components, the likelihood is written as

$$p(y | \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_{k=1}^{K} w_k \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k), \tag{2.19}$$

where $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \Sigma_k, w_k\}$ parameterizes component k by its mean vector $\boldsymbol{\mu}_k$, its mixing proportion or weight w_k (where $0 \leq w_k \leq 1$ and $\sum_k w_k = 1$), and its covariance matrix Σ_k.

As for the HMM in the Bayesian context, prior distributions were placed on each component's parameters, which in turn were parameterized by a set of hyperparameters shared across components. Similar to the transition probabilities, the mixing proportions for each component are typically given a Dirichlet conjugate prior with hyperparameter vector $\boldsymbol{\alpha}$:

$$\boldsymbol{w} | \boldsymbol{\alpha} \sim \mathrm{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_K), \tag{2.20}$$

To achieve an unbounded set of mixing components and their respective mixing proportions in non-parametric methods, the infinite sequence of mixing proportions are generated by similarly drawing atoms from a GEM distribution, as in equation 2.17. These atoms then parameterize the associated observations according to an indexed family of distributions $f(\cdot)$:

$$y_i|\boldsymbol{\theta}'_i \sim f(\boldsymbol{\theta}'_i) \tag{2.21}$$

As before, G_0 is typically the joint conjugate prior for the means and covariances that specify the Gaussian components (i.e., the normal-inverse-Wishart distribution, equation 2.10) [22]. In other words, the atoms of the DP parameterize Gaussians centered around the base hyperparameters. The concentration hyperparameter, α, determines the extent to which the atoms cluster around G_0. This construction allows the parameters and weights of the unbounded set of Gaussian components to be determined by the data, often in stochastic samplers such as *Markov Chain Monte Carlo* (MCMC) methods, or using deterministic methods such as variational inference. The inferential framework for such models will be explored in Section 2.3.

2.2.3 Hierarchical Dirichlet Process HMMs

The HDP is a stochastic process in which the base measure (G_0 previously) is itself drawn from a DP prior. This is useful in the situation where data is naturally clustered into distinct groups, but produced by related generative processes. An example given in [19] is a sensor network collecting data from an environment in which the quality of collected data varies with time. In the stick-breaking representation of the HDP-HMM, we begin by producing an unbounded sequence of stick breaks according to a GEM process parameterized by α. This "base-stick" is then used for generating the unbounded number of transition rows, distributed according to a DP, itself parameterized by the base stick and an additional concentration parameter, γ.

The hidden state sequence is sampled according to the usual HMM process, and observations conditioned on a hidden state are drawn from an observation distribution (say a Gaussian). These state-specific observation distributions are parameterized by draws from the corresponding conjugate prior. This can be summarized either in graphical form according to Figure 2.5, or as the following generative process:

$$\beta \sim \text{GEM}(\alpha)$$

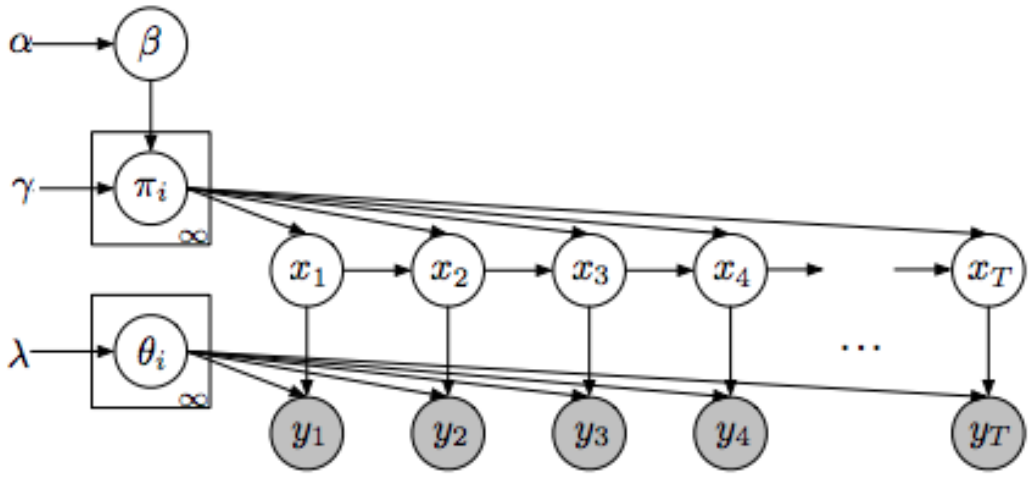

Figure 2.5: HDP-HMM structure. Taken from [9].

For $i = 1, 2, \ldots$:

$$\begin{aligned} \boldsymbol{\pi}_i &\overset{iid}{\sim} \mathrm{DP}(\gamma, \beta) \\ \boldsymbol{\theta}_i &\overset{iid}{\sim} H \end{aligned} \tag{2.22}$$

For $t = 1, 2, \ldots, T$:

$$\begin{aligned} x_t &\sim \boldsymbol{\pi}_{x_{t-1}} \\ y_t &\sim f(\boldsymbol{\theta}_{x_t}), \end{aligned} \tag{2.23}$$

The model is thus hierarchical since the transition objects, $\boldsymbol{\pi}_i$ are generated from a DP that is parameterized by the same discrete measure β. As will be shown in Section 2.3.2, this hierarchical link breaks conjugacy and makes inference in these models difficult. Sampling methods to bypass this difficulty will be discussed. Since we can't explicitly instantiate all of the infinite latent parameters, a so-called *weak limit*, L, is often placed on the number of unique components. This produces a finite approximation to the HDP-HMM, which, in the limit that $L \to \infty$, converges to a true HDP-HMM [19]. Important to note is that this parameter is an upper bound; one can grossly overestimate the necessary components and let the preferential attachment of the model restrict the necessary states. Nevertheless, the weak-limit parameter can also be modified as required by the data in beam sampling proposed by Van Gael et al [23].

2.2.4 Hierarchical Dirichlet Process HSMMs

The extension to semi-Markov models is relatively straight-forward from this point. Drawing from equations 2.22 and 2.23, the HDP-HMM can be modified to include explicit duration distributions to arrive at the graphical model shown in Figure 2.6. This generative process can be described by the following:

$$\beta \sim \text{GEM}(\alpha)$$

For $i = 1, 2, \ldots$

$$\begin{aligned} \boldsymbol{\pi}_i &\overset{iid}{\sim} \text{DP}(\gamma, \beta) \\ (\boldsymbol{\theta}_i, \omega_i) &\overset{iid}{\sim} H \times G \end{aligned} \tag{2.24}$$

For $s = 1, 2, \ldots$

$$\begin{aligned} z_s &\sim \bar{\boldsymbol{\pi}}_{z_{s-1}} \\ D_s &\sim g(\omega_{z_s}) \\ x_{t:(t+D_s-1)} &= z_s \\ y_{t:(t+D_s-1)} &\overset{iid}{\sim} f(\boldsymbol{\theta}_{x_t}) \end{aligned} \tag{2.25}$$

where each t is the sum of previously sampled durations (i.e., $t = \sum_{\bar{s}<s} D_{\bar{s}}$), and where $x_{t:(t+D_s-1)}$ is the indicator for the super-state z_s over the sampled duration D_s (shown in Figure 2.6.

In words, we generate the same base-stick as in the HDP-HMM case, and use it to generate the hierarchical transition rows. The associated observation and duration parameterizations $(\boldsymbol{\theta}_i, \omega_i)$ are drawn *i.i.d.* from their respective conjugate base distributions. For example, if the durations are distributed as Poisson, $\omega_i \sim \text{Gamma}(\cdot)$. As before, the super-state sequence is sampled according to semi-Markov dynamics with $\bar{\pi}_{ij} = \frac{\pi_{ij}}{(1-\pi_{ii})}(1 - \delta_{i,j})$ (i.e., self-transitions are forbidden). A duration D_s is sampled from the observation distribution (Poisson in the example above) parameterized by the ω corresponding to state z_s. The label sequence for the duration sampled is set to the sampled super-state (i.e., $x_{t=s:s+D_s} = z_s$), and finally the observations for that duration are drawn *i.i.d.* from the observation distribution parameterized by the corresponding $\boldsymbol{\theta}$ draw. Important to note is that the HDP-HMM case can be recovered by explicitly placing geometric distributions on the state durations.

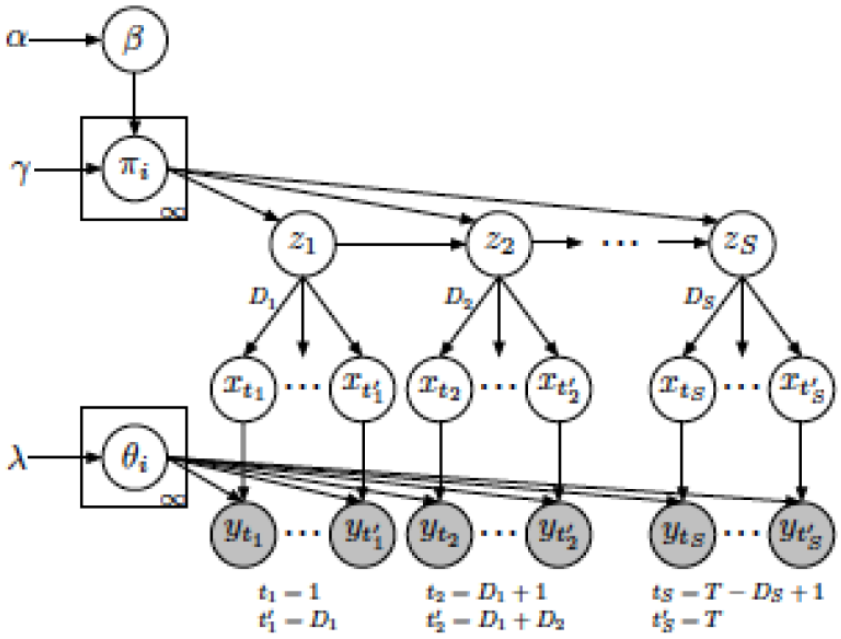

Figure 2.6: HDP-HSMM structure. Taken from [9].

2.2.5 Factorial Hierarchical Dirichlet Process HSMMs

A final extension of these ideas is to apply the non-parametric generative model shown above to factorial structures, which will be used in Chapter 4. Assuming an independent noise source w, factorial models compose an overall chain of observables $\bar{y}$ by a sum of independent Markov chains:

$$\bar{y}_t = \sum_{k=1}^{K} y_t^{(k)} + w_t, \tag{2.26}$$

where

$$y^{(k)} \sim \text{HDP-HSMM}(\alpha_k, \gamma_k, H_k, G_k). \tag{2.27}$$

That is, each Markov chain contributing to the overall observable, $\bar{y}_t$ is distributed according to the generative model described for the HDP-HSMM previously. In this framework, each chain has its own set of hyperparameters, including base observation and duration distributions, as well as concentration parameters for generating the hierarchical states. This highly flexible model is intuitively a reasonable match for the type of processes involved in home power signals, where each appliance contributes to the overall aggregate with (approximately) independent dynamics. Assuming priors that are sufficiently close to the actual appliance dynamics (and separate from the other contributing appliances), this method constitutes a potentially unsupervised NILM solution. However, as will be

discussed at some length, it seems highly unlikely to be able to provide these specific, yet general, appliance priors for labelling the returned states. Before this discussion, however, an important component to the underlying success in these models lies in inference methods. The following section outlines several methods for inference as well as their application in these complex generative models.

2.3 Inference

2.3.1 Exact Inference

The usual goal in the Bayesian framework is to use observed data to form a likelihood given prior assumptions, so as to update the posterior distribution and estimate model parameters. However, the normalizing constant for the posterior distribution, called the marginal likelihood ($p(\mathbf{x}) = \int_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$) is NP-hard to compute exactly (i.e., is computationally intractable) for arbitrary graphical models [24]. The distribution over model parameters $\{\boldsymbol{\theta}\}$ is often high-dimensional and unavailable in closed form. Simple models for which exact inference is possible involve highly strict assumptions of independence (e.g., in standard HMMs, the Markov property and finite cardinality). However, with the possibility of exact inference comes significant restriction on the expressiveness of the model to map onto actual patterns in the data. If we move to more complex generative models, such as a model that is non-parametric, we face the intractability in the marginal likelihood. In these cases we are forced to appeal to approximate inference methods. Nevertheless, there are important parallels between approximate inference in complex models such as the factorial HDP-HSMM and exact methods for simpler models. Two examples of exact inference algorithms to be explored here are the *Forward-Backward algorithm* [25] and the *Viterbi algorithm* [26].

Forward-Backward, Viterbi Algorithms

Returning to the simple HMM discussed initially, we assume the transition and emission matrices are determined, along with the initial distribution, $\pi_0(i)$. Given these, important questions regarding the hidden state sequence can be addressed. For example, consider the posterior probability of state x_k given all available evidence $y_1, y_2, \ldots, y_n$, $n > k$:

$$p(x_k|y_{1:n}), \tag{2.28}$$

where the notation $1:n$ indicates $y_1, y_2, \ldots, y_n$. Computation of this distribution is achieved using the forward-backward algorithm, comprised of two parts unsurprisingly called the 'forward' part and the 'backward' part. The respective goals of each are to compute

$$p(x_k, y_{1:k}), \tag{2.29}$$

and

$$p(y_{k+1:n}|x_k). \tag{2.30}$$

To see this, we note that by the usual rules of conditional probability, equation 2.28 can be written

$$p(x_k|y_{1:n}) \propto p(x_k, y_{1:n}) = p(y_{k+1:n}|x_k, y_{1:k})p(x_k, y_{1:k}) \tag{2.31}$$

But examining Figure 2.2 shows that $y_{1:k}$ is d-separated from $y_{k+1:n}$ given x_k, since x_k blocks the only path from $y_{1:k}$ to $y_{k+1:n}$. Therefore, we can rewrite equation 2.31 to exclude this conditional dependence:

$$p(x_k|y_{1:n}) \propto p(x_k, y_{1:n}) = p(y_{k+1:n}|x_k)p(x_k, y_{1:k}), \tag{2.32}$$

which is the product of forward part and backward part as claimed. Once $p(x_k, y_{1:n})$ is determined, we can recover the desired conditional distribution by using Bayes' Theorem (i.e., dividing by $\sum_k p(x_k)$).

As an example of how one would go about finding the quantities in equations 2.29 and 2.30, the following will lay out the recursion relation used to find the forward part, and a similar argument can be made for the backward part (see [27]). To reiterate, the goal of the forward algorithm is to compute the joint probability $p(x_k, y_{1:k})$. We can construct a useful recursion to take advantage of dynamic programming by marginalizing over x_{k-1} and using the Markov properties:

$$\begin{aligned} p(x_k, y_{1:k}) &= \sum_{x_{k-1}=1}^{m} p(x_k, x_{k-1}, y_{1:k}) \\ &= \sum_{x_{k-1}=1}^{m} p(y_k|x_k, x_{k-1}, y_{1:k-1})p(x_k|x_{k-1}, y_{1:k-1})p(x_{k-1}, y_{1:k-1}) \end{aligned} \tag{2.33}$$

By d-separation, y_k is conditionally independent of x_{k-1} and $y_{1:k-1}$ given x_k. Additionally, x_k is conditionally independent of $y_{1:k-1}$ given x_{k-1}. So the equation 2.33 simplifies to

$$p(x_k, y_{1:k}) = \sum_{x_{k-1}=1}^{m} p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}, y_{1:k-1}), \tag{2.34}$$

which is exactly

$$p(x_k, y_{1:k}) = \sum_{x_{k-1}=1}^{m} \vartheta_{x_k}(y_k)\pi_{x_{k-1},x_k}p(x_{k-1}, y_{1:k-1}). \tag{2.35}$$

If we define $p(x_k, y_{1:k}) \stackrel{\text{def}}{=} \alpha_{x_k,k}$, we can write

$$\alpha_{x_k,k} = \sum_{x_{k-1}=1}^{m} \vartheta_{x_k}(y_k)\pi_{x_{k-1},x_k}\alpha_{x_{k-1},k-1}. \tag{2.36}$$

This result allows us to find the probability of getting the observation sequence from $t = 1 : k$, and ending up in state x_k. In other words, at each time step we calculate the total probability of arriving at state x_k from every possible sequence of hidden states, involving a time complexity of $\mathcal{O}(nm^2)$ [28].

If instead we decided to only keep track of the maximizing sequence of hidden states ending at each value of x_k, we reduce the number of sequences to m, since each new x_k will be maximally likely to belong to one particular sequence. From there, we can determine the maximally probable sequence of hidden states that can explain the observed data. In other words, we can find the *Maximum A-Posteriori* (MAP) sequence of hidden states, i.e., $\text{argmax}_x\ p(x|y)$. This method is known as the Viterbi algorithm [26].

We can approach this problem in a similar way as the forward algorithm by setting up a recursion relation. First, since the joint and conditional are proportional given Bayes' Theorem, we can equally well write

$$\underset{x}{\text{argmax}}\, p(x|y) = \underset{x}{\text{argmax}}\, p(x, y) \tag{2.37}$$

If we choose some time step k arbitrarily, we can find a recursion for the joint distribution $p(x_{1:k}, y_{1:k})$ by using the Markov properties. Following the trellis diagram in Figure 2.2, we can segment the joint probability into two sections: $1 : k-1$ and k by itself:

$$p(x_{1:k}, y_{1:k}) = p(y_k|x_k)p(x_k|x_{k-1})p(x_{1:k-1}, y_{1:k-1}). \tag{2.38}$$

Now, since we want to find the maximally probable sequence of states, we can maximize both sides of equation 2.38 over x_k from 1 to $k-1$:

$$\max_{x_{1:k-1}} p(x_{1:k}, y_{1:k}) = \max_{x_{1:k-1}} p(y_k|x_k)p(x_k|x_{k-1})p(x_{1:k-1}, y_{1:k-1}). \tag{2.39}$$

Now the trick: since $p(y_k|x_k)$ and $p(x_k|x_{k-1})$ are both independent of $x_{1:k-2}$ given the Markov property, we can split the maximization to be over x_{k-1} and $x_{1:k-2}$[3]:

$$\max_{x_{1:k-1}} p(x_{1:k}, y_{1:k}) = \max_{x_{k-1}} \left[p(y_k|x_k)p(x_k|x_{k-1}) \max_{x_{1:k-2}} p(x_{1:k-1}, y_{1:k-1}) \right], \tag{2.40}$$

[3]This holds when both functions ($p(y_k|x_k)p(x_k|x_{k-1})$ and $p(x_{1:k-1}, y_{1:k-1})$) are non-negative for all possible arguments, as distributions of course must be.

which now involves a recursion relation in the $p(x_{1:k}, y_{1:k})$'s. Also note the familiar quantities which were assumed to be known: $p(y_k|x_k) = \vartheta_{x_k}(y_k)$, and $p(x_k|x_{k-1}) = \pi_{x_{k-1},x_k}$. Furthermore, let's define

$$\max_{x_{1:k-1}} p(x_{1:k}, y_{1:k}) \stackrel{\text{def}}{=} \mu_k(x_k), \tag{2.41}$$

allowing equation 2.40 to be written as

$$\mu_k(x_k) = \max_{x_{k-1}} \Big[\vartheta_{x_k}(y_k)\pi_{x_{k-1},x_k}\mu_{k-1}(x_{k-1})\Big], \tag{2.42}$$

and finally, since $\vartheta_{x_k}(y_k)$ is independent of x_{k-1}, we write

$$\mu_k(x_k) = \vartheta_{x_k}(y_k) \max_{x_{k-1}} \Big[\pi_{x_{k-1},x_k}\mu_{k-1}(x_{k-1})\Big]. \tag{2.43}$$

The left-hand side expresses the value of the joint probability of a state chain ending up in state x_k and emitting observation y_k, when maximizing over all possible state chains up to $k-1$. For each possible final state, say for $x_k = j$, the probability of being in the hidden state j at time t is the maximal product of the previous states in the sequence with their associated transition and emission probabilities, which we could write as $\mu_k(x_k = j)$. The Viterbi algorithm is expressed in Algorithm 1.

Algorithm 1 The HMM Viterbi Algorithm

1: Start by initializing $\mu_0(x_0 = i) = \pi_0(i)\vartheta_i(y_0)$ for each state i, $\phi_0 = 0$
2: run forward in time t:
3: **for** each possible state j: **do**
4: $\quad \mu_t(x_t = j) = \max_i \left(\mu_{t-1}(x_{t-1} = i)\pi_{i,j}\right)\vartheta_j(y_t)$
5: $\quad \phi_t(x_t = j) = \text{argmax}_i \left(\mu_{t-1}(x_{t-1} = i)\pi_{i,j}\right)\vartheta_j(y_t)$
6: **end for**
7: from $t = T$, set $q_T^* = \text{argmax}_i(\mu_T(x_T = i))$, then run backward in time t:
8: compute $q_{t-1}^* = \phi_{q_t^*,t}$

Whereas the forward algorithm determines the total probability via any sequence of hidden states to end up in a particular one at a given time, the Viterbi algorithm determines the most probable sequence of hidden states for a given sequence of observations. It walks forward in time computing the probability (the μ terms) and states (the ϕ terms) associated with a given sequence, then finds the maximally probable final state given the final observation. Finally, it walks backward in time, and selects those states leading up to the most probable final state as the most likely sequence of states (the $q^*_{t=1:T}$ terms).

As mentioned, equation 2.32 recovers the desired posterior only after normalizing by the marginal likelihood. This is often intractable and so the true posterior is inaccessible. In these situations, we must appeal to approximate inference methods.

2.3.2 Approximate Inference

In approximate inference, there are two main historical approaches. The first involves stochastic, sampling-based methods. These methods construct solutions to inferential tasks based on sampling a large number of times from a proposal distribution q and evaluating those samples relative to the unnormalized desired distribution [29]. These *Monte Carlo* sampling methods approximate target expectations by

$$\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{2.44}$$

where the x_i are samples drawn from p. By the strong law of large numbers, we know that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \mathbb{E}_{x\sim p}[f(x)], \tag{2.45}$$

meaning we can approximate the true expectation to arbitrary precision given enough *i.i.d.* samples from p.

There are several different types of samplers under the umbrella of Monte Carlo methods, including rejection and importance sampling, for example. We will restrict our discussion to a particularly popular and useful class of algorithms known as MCMC methods. In MCMC, iterative sampling is performed on a specially constructed ergodic Markov chain whose stationary distribution is the (arbitrarily complex) desired distribution [19]. Ergodicity in a Markov chain is jointly implied by

1. Aperiodicity: For every state in the system, there is no period τ for which a return to the state must occur.

2. Harris recurrence [30]: Every state is visited by the system an unbounded number of times as $t \to \infty$.

The Markov chain is traversed in a random walk determined by the *transition kernel*, which determines the dynamics of the sample sequence. Under the *detailed balance* condition, the Markov chain will have the desired distribution p as its stationary distribution. For a transition kernel $\kappa(\cdot|\cdot)$, detailed balance is defined by

$$\kappa(y|x)p(x) = \kappa(x|y)p(y). \tag{2.46}$$

An equivalent statement is that the Markov chain is *reversible* with respect to p [19]. Despite guarantees of convergence to the target distribution under conditions of ergodicity and reversibility, poor initializations or poorly constructed proposal distributions can result in significant *burn-in time* before the chain reliably produces samples from the target distribution [29]. Poor choice of proposal distribution can also result in only small regions of

the state space to be explored, a phenomenon known as poor *chain mixing*. Finally, determining the rate of convergence is difficult, often leaving only heuristic estimates or loose bounds [29].

Sampling Algorithms

Metropolis-Hastings Sampler One highly popular MCMC sampler is the *Metropolis-Hastings* (MH) algorithm [31], shown for the simplest case in Algorithm 2.

Algorithm 2 METROPOLIS-HASTINGS SAMPLER

1: Initialize $x^{(0)}$
2: **for** $t = 1, 2, \ldots,$ **do**
3: Sample $x' \sim q(x'|x^{(t-1)})$
4: Calculate the acceptance probability:

$$\rho(x'|x^{(t-1)}) = \min\left\{\frac{p(x')q(x^{(t-1)}|x')}{p(x^{(t-1)})q(x'|x^{(t-1)})}, 1\right\} \tag{2.47}$$

5: Sample

$$x^{(t)} \sim \rho(x'|x^{(t-1)})\delta_{x'} + \Big(1 - \rho(x'|x^{(t-1)})\Big)\delta_{x^{(t-1)}}, \tag{2.48}$$

where δ_x is a Dirac mass at x.
6: **end for**

In words, at each iteration the Metropolis-Hastings algorithm samples from the proposal distribution and accepts that sample with certain probability. This probability is proportional to the ratio of the probability of the proposal sample under the (unnormalized) target distribution relative to the previously accepted sample. The proportionality is determined by the factor $q(x^{(t-1)}|x')/q(x'|x^{(t-1)})$, which accounts for non-symmetric proposal distributions, where the probability of transition from $x^{(t-1)}$ to x' may differ from the reverse. In other words, it may be much more likely under the proposal distribution to transition from x_i to x_j than from x_j to x_i. This weighs the $j \to i$ transition less heavily in ρ, even for equal probabilities in the target distribution. In this way, the algorithm gradually reinforces samples which are highly likely under the target distribution, allowing the use of samples from the converged Markov chain to approximate samples from the desired posterior distribution. As mentioned, the marginal likelihood term is typically intractable, and so these methods sidestep the need to compute it exactly.

Again, implicitly assumed in the MH algorithm is that the target distribution up to a normalization factor can be evaluated easily. If the joint target distribution can be factored into conditional distributions from which sampling is easy (e.g., mixture models), the Gibbs sampler can instead be used.

Gibbs Sampler The Gibbs sampler is in some sense a simplification of the MH algorithm in that the acceptance probability ρ is always 1. That is, every sampling step modifies the value of a given variable; there is no proposal distribution necessary. In other ways it is more complex, in that the conditional distributions for the desired variables need to be explicitly defined. In Gibbs sampling, samples from the joint target distribution of $x_1, x_2, \ldots, x_n$ are generated by sampling from the associated conditional densities. Conditional independence given d-separation in directed graphical models is a useful property to simplify this sampling process. The graph can be broken into conditionally independent sub-graphs defined for a given node by its *Markov blanket* (the set of nodes d-separating the given node from the rest of the graphical model). Generative models like HMMs and their more sophisticated variants are sparse in their connections given the memoryless property. These implied independencies can be leveraged for efficient sampling following Algorithm 3, as seen in [19].

Algorithm 3 GIBBS SAMPLER

Given a previous sample $\boldsymbol{x}^{(t-1)} = (x_1^{(t-1)}, \ldots x_n^{(t-1)})$, generate:
$x_1^{(t)} \sim p_1(x_1 | x_2^{(t-1)}, \ldots, x_n^{(t-1)})$
$x_2^{(t)} \sim p_2(x_2 | x_1^{(t)}, x_3^{(t-1)} \ldots, x_n^{(t-1)})$
$\vdots$
$x_n^{(t)} \sim p_n(x_n | x_1^{(t)}, \ldots, x_{n-1}^{(t)})$.

As shown, the Gibbs algorithm samples a variable assignment from the conditional density for that variable, assuming the values of the other variables within its Markov blanket. The sampling sequence can be reversed or randomized, often improving the rate of convergence to the stationary target distribution [32]. The equality of the stationary distribution of the Markov chain defined by Algorithm 3 and the desired target distribution is guaranteed under the condition of ergodicity [19]. A sufficient condition for ergodicity is Harris recurrence, which in this sampler is verified given absolute continuity of the transition kernel with respect to the dominating measure (see [33] for details).

Many variants of the basic Gibbs sampler exist, including auxiliary sampling, blocked sampling, collapsed sampling, beam sampling, and more. Several of these will be relevant to inference in the factorial HDP-HSMM framework, but the general structure remains: knowing the (unnormalized) conditional distributions over individual components of the model allows sequential resampling of each component.

Approximate Inference in HDP-HSMMs In the HDP-HSMM, the ultimate goal for a given time t is to produce samples from the true posterior

$$p((\boldsymbol{x}_t), \{\boldsymbol{\theta}_i\}, \{\boldsymbol{\pi}_i\}, \{\omega_i\} | (y_t), H, G, \alpha, \gamma) \tag{2.49}$$

Each of the components of the posterior have known conditional distributions. Given the choice of conjugate pairs, sampling $\{\boldsymbol{\theta}_i\}$ and $\{\omega_i\}$ follow the appropriate update equations that can be found elsewhere (again, see [16]). Sampling the conditional distribution of the state sequence associated with $(\boldsymbol{x}_t)|\{\boldsymbol{\theta}_i\}, \{\boldsymbol{\pi}_i\}, \{\omega_i\}, (y_t)$ can be developed in a manner similar to the forward-backward algorithm. Conditioning on $\{\theta_i\}, \{\pi_i\}, \{\omega_i\}, (y_t)$, a finite number of components will have been instantiated such that we can treat the sequence as finite dimensional.

Similarly to the forward-backward algorithm, we define in the HSMM case the equivalent backward message $\beta_t(x, d) \stackrel{\text{def}}{=} p(y_{t+1:T}|x, d, F_t = 1)$, where F_t is a change-point indicator such that a new segment begins at time $t + 1$. This quantity is the probability of future evidence (observations $y_{t+1:T}$) assuming the system terminates in state x at time t after persisting for a duration d.

This can be rewritten recursively as

$$\beta_t(x, d) = \sum_{x'} \sum_{d'} \beta_{t+d'}(x', d')p(y_{t+1:t+d'}|x', d')p(x', d'|x, d), \tag{2.50}$$

in other words, we calculate the probability of observations in the block $t + 1 : t + d'$ multiplied by the probability of transitioning into state x' from state x, and the previous message $\beta_{t+d'}$, marginalizing over all possible duration draws d' and state transitions x'.

A simplifying, standard assumption is that the current sampled duration depends only on the current hidden state, and also that the current hidden state depends only on the previous one. That is,

$$p(x', d'|x, d) = p(x'|x)p(d'|x')$$

This simplifies equation 2.50, leaving the right hand side independent of d, giving

$$\beta_t(x) = \sum_{x'} \sum_{d'} \beta_{t+d'}(x')p(y_{t+1:t+d'}|x', d')p(x'|x)p(d'|x'), \tag{2.51}$$

which, separating the sumation indices, can be written as

$$\beta_t(x) = \sum_{x'} p(x'|x) \sum_{d'} p(d'|x')\beta_{t+d'}(x')p(y_{t+1:t+d'}|x', d'). \tag{2.52}$$

By definition, we set $\beta_T(x) = 1$. Now we rewrite the above introducing a new term:

$$\beta_t(x) = \sum_{x'} p(x'|x)\beta^*(x'), \tag{2.53}$$

where $\beta^*(x') \stackrel{\text{def}}{=} p(y_{t+1:T}|x', F_t = 1) = \sum_{d'} p(d'|x')\beta_{t+d'}(x')p(y_{t+1:t+d'}|x', d')$.[4]

For inference of the state sequence, we first want to compute $p(x_1 = i|y_{1:T})$, using Bayes' theorem, this can be written (suppressing the notation indicating conditioning on other model parameters):

$$p(x_1 = i|y_{1:T}) \propto p(x_1 = i)p(y_{1:T}|x_1 = i, F_0 = 1) \tag{2.54}$$

$$= p(x_1 = i)\beta_0^*(i). \tag{2.55}$$

This allows the first state (say $x_1 = \bar{x}_1$) to be drawn according to this distribution as per the usual Gibbs step. Once the state has been selected, we still need to sample the posterior of the conditional duration distribution; i.e., we need

$$p(D_1 = d|y_{1:T}, x_1 = \bar{x}_1, F_0 = 1) = \frac{p(D_1 = d, y_{1:T}|x_1 = \bar{x}_1, F_0 = 1)}{p(y_{1:T}|x_1 = \bar{x}_1, F_0 = 1)}. \tag{2.56}$$

Here, we can use Bayes' Theorem to split the observations in the numerator of equation 2.56 up to the sampled duration d (suppressing the conditioned change-point indicator $F_0 = 1$):

$$p(D_1 = d|\ldots) = \frac{p(D_1 = d|x_1 = \bar{x}_1)p(y_{1:d}|D_1 = d, x_1 = \bar{x}_1)p(y_{d+1:T}|D_1 = d, x_1 = \bar{x}_1)}{p(y_{1:T}|x_1 = \bar{x}_1)}, \tag{2.57}$$

and using the definition of β and β^*, we can write

$$p(D_1 = d|\ldots) = \frac{p(D_1 = d|x_1 = \bar{x}_1)p(y_{1:d}|D_1 = d, x_1 = \bar{x}_1)\beta_d(\bar{x}_1)}{\beta_0^*(\bar{x}_1)}. \tag{2.58}$$

Given the pre-computed backwards messages β and β^*, a duration d_1 can be sampled from the conditional posterior given in equation 2.58. Following this, the next state distribution can be computed according to:

$$p(x_{d_1+1} = i|x_1 = \bar{x}_1, y_{(d_1+1):T}) \propto p(x_{d_1+1} = i|x_1 = \bar{x}_1)p(y_{(d_1+1):T}|x_{d_1+1} = i, x_1 = \bar{x}_1), \tag{2.59}$$

but given d-separation,

$$p(y_{(d_1+1):T}|x_{d_1+1} = i, x_1 = \bar{x}_1) = p(y_{(d_1+1):T}|x_{d_1+1} = i) = \beta_{d_1}^*(i), \tag{2.60}$$

[4] According to [34], an additional "censoring" term is usually required in this expression to account for segments that exceed the observation length T. This term relates to the survival function of the states' duration distribution (i.e. $1 - \text{CDF}(\cdot)$). It has been omitted here for simplification of what follows.

and so we get

$$p(x_{d_1+1} = i | x_1 = \bar{x}_1, y_{(d_1+1):T}) \propto p(x_{d_1+1} = i | x_1 = \bar{x}_1) \beta^*_{d_1}(i), \tag{2.61}$$

the distribution from which x_{d_1+1} is drawn. This process continues to $t = T$, giving conditional posterior samples of the state sequence and the associated durations.

As mentioned in Section 2.1.2, several efficiencies can be leveraged for this inference process, the most significant of which include truncation and change-point selection. It is often the case that the length of observations T exceeds the maximum expected state duration d_{max}. The support of the duration distributions can be truncated and the asymptotic complexity of this message passing scheme can be improved for state cardinality N from $\mathcal{O}(T^2N + TN^2)$ to $\mathcal{O}(Td_{max}N + TN^2)$ [9]. The support of the duration distributions can further be reduced by considering only durations $d \in \mathcal{D}$, where $\mathcal{D}$ is the times t for which a change-point occurs. All other times can be ignored in message passing. For a number of change-points $T_c = |\mathcal{D}|$, this reduces the base asymptotic complexity from $\mathcal{O}(T^2N + TN^2)$ to $\mathcal{O}(T_c^2N + T_cN^2)$ [9]. Luckily, it is often the case that $T_c << T$, greatly improving inference cost. Changepoints can be detected within the data by a variety of methods: Kalman filters, simple thresholds on signal differences or cumulative differences, functions of empirical variance, and many more. This provides an additional opportunity to make use of the filter described in Chapter 4, which will be shown to be fast and accurate in detecting change-points.

To summarize, conjugate resampling of observation and duration distributions is a relatively simple problem with well-established update equations, and we have now determined how to resample state sequences as well as state durations with message passing. The final component in the Gibbs sampling chain of the true posterior in equation 2.49 is the unbounded transition rows, $\boldsymbol{\pi}_i$. As discussed in Section 2.2.3, having been produced through a DP parameterized by the same base-stick, the transition rows are hierarchically linked. It was claimed that this loss of mutual independence destroys conjugacy in the case where self-transitions are forbidden (i.e. HSMMs). This loss of conjugacy makes resampling difficult. To show this is the case, let's assume a particular state (say, state 1) as part of a resampled state sequence $\boldsymbol{x}_t$, a concentration parameter γ, and a hierarchical base stick β. The posterior for the transition probabilities for state 1 can be written using Bayes' Theorem as:

$$p(\boldsymbol{\pi}_1 | (\boldsymbol{x}_t), \beta) \propto p(\boldsymbol{\pi}_1 | \beta) p((\boldsymbol{x}_t) | \boldsymbol{\pi}_1), \tag{2.62}$$

which, expanding $p(\boldsymbol{\pi}_1|\beta)$ as Dirichlet in the weak-limit of L maximum states, becomes

$$p(\boldsymbol{\pi}_1 | (\boldsymbol{x}_t), \beta) \propto \pi_{11}^{\gamma\beta_1 - 1} \dots \pi_{iL}^{\gamma\beta_L - 1} \left(\frac{\pi_{12}}{1 - \pi_{11}} \right)^{n_{12}} \dots \left(\frac{\pi_{1L}}{1 - \pi_{11}} \right)^{n_{1L}}. \tag{2.63}$$

Here, n_{ij} is the number of transitions from state i to state j within the state sequence $(\boldsymbol{x}_t)$. This expression arises since the posterior of $\boldsymbol{\pi}_1$ in the weak-limit HDP-HMM case is distributed as $\text{Dir}(\gamma\beta_1, \ldots, \gamma\beta_L)$, but in the semi-Markov case we have $\bar{\pi}_{ij} = \frac{\pi_{ij}}{(1-\pi_{ii})}(1-\delta_{ij})$, resulting in the trailing terms above where $i \neq j$. The expression presented in equation 2.63 cannot be cast in terms of $\text{Dir}(\cdot)$, and so conjugacy is lost in the HDP-HSMM.

In order to be able to recast this expression in a conjugate way, we need to include a term proportional to $(1-\pi_{11})^{n_1}$, where $n_j = \sum_{i;i\neq j} n_{ji}$ (i.e., the total number of transitions out of state j.) This sort of consideration is the premise of *auxiliary sampling*; we augment the generative process to include an extra variable, even if the auxiliary variable has no physical interpretation. This variable's properties in the expansion serve to clean up the posterior and regain conjugacy with the prior. A distribution that behaves in the desired way is a geometrically distributed random variable, which we'll call ρ, supported on $\{0, 1, \ldots\}$ with success probability $(1-\pi_{11})$. In other words, for each transition out of state 1, we sample $\rho_i|\pi_{11} \overset{iid}{\sim} \text{Geom}(1-\pi_{11})$. This process is shown graphically in Figure 2.7.

The new posterior is then

$$\begin{aligned} p(\boldsymbol{\pi}_1|(\boldsymbol{x}_t), \beta, \{\rho_i\}) &\propto p(\boldsymbol{\pi}_1|\beta)p((\boldsymbol{x}_t)|\boldsymbol{\pi}_1)p(\{\rho_i\}|\{\pi_{1i}\}) \\ \propto \pi_{11}^{\gamma\beta_1-1}\ldots\pi_{iL}^{\gamma\beta_L-1}&\left(\frac{\pi_{12}}{1-\pi_{11}}\right)^{n_{12}}\ldots\left(\frac{\pi_{1L}}{1-\pi_{11}}\right)^{n_{1L}}\left(\prod_{i=1}^{n}\pi_{11}^{\rho_i}(1-\pi_{11})\right). \end{aligned} \tag{2.64}$$

Expanding the product, we get

$$\begin{aligned} p(\boldsymbol{\pi}_1|(\boldsymbol{x}_t), \beta, \{\rho_i\}) &\propto \pi_{11}^{\gamma\beta_1-1}\ldots\pi_{iL}^{\gamma\beta_L-1}\pi_{12}^{n_{12}}\ldots\pi_{1L}^{n_{1L}}\left(\prod_{i=1}^{n}\pi_{11}^{\rho_i}\right) \\ &= \pi_{11}^{\gamma\beta_1+\sum_i\rho_i-1}\pi_{i2}^{\gamma\beta_2+n_2-1}\ldots\pi_{iL}^{\gamma\beta_L+n_L-1} \\ &\propto \text{Dir}\left(\gamma\beta_1+\sum_i\rho_i, \gamma\beta_2+n_2, \ldots, \gamma\beta_L+n_L\right). \end{aligned} \tag{2.65}$$

Thus, the posterior can once again be expressed as Dirichlet, allowing conjugate resampling according to the usual update equations for these distributions. With this, all components of the HDP-HSMM have their conditional distributions expressed up to a normalization constant, allowing these components to be resampled sequentially according to the Gibbs sampler in Algorithm 3. In the factorial setting, each sub-chain is generated by an HDP-HSMM, and so this inference framework transfers easily to factorial models.

Stochastic samplers are not the only methods for approximate inference, however. And despite their enormous historical success and flexibility, these methods are not without their drawbacks. As mentioned, properties relevant to the Gibbs chain's stationary distribution, along with the certainty and rate of convergence, are difficult to verify in general. More

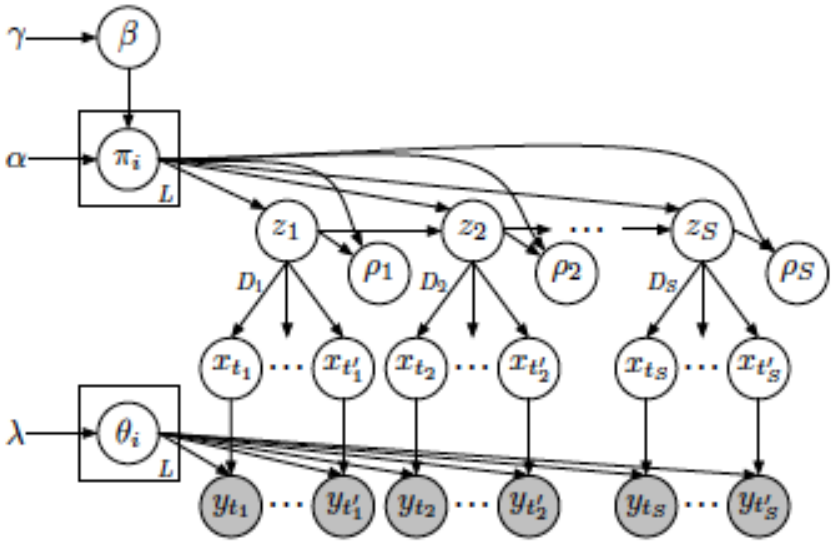

Figure 2.7: HDP-HSMM structure with auxiliary variables (ρ) to retain conjugacy. Taken from [9].

sophisticated methods such as hybrid Gibbs-MH sampling or reversible jump MCMC are significantly more challenging to verify [29]. Furthermore, such sampling chains can become locked in modes of high probability, where transitional probabilities in the Markov kernel become spread out over a large number of states. Alternately, probabilities can concentrate to a small subset of the state space. In either case, convergence to the true posterior can take a prohibitive amount of time [29].

In contrast to sampling methods, another approach to approximate inference is to squeeze the value of the intractable posterior by establishing a proposal distribution that bounds it, which is by construction amenable to optimization. This approach is called variational inference.

Variational Inference

If the true posterior is inaccessible under our model, our goal is then to try to find a different distribution that *looks* like the true posterior, but is structured such that we can deal with it tractably. The extent to which two distributions look similar is a somewhat ambiguous notion, but one commonly used measure is called the *Kullback-Leibler* (KL) divergence, which is defined for two distributions q and p as:

$$KL(q||p) = \mathbb{E}_q \log \frac{q}{p}. \tag{2.66}$$

This measure is called a divergence rather than a distance because it is not symmetric with respect to exchange of p and q, but has the properties that $KL(q||p) \geq 0$ and $KL(q||p) = 0$ if and only if $q = p$ [24].

Suppose p is the posterior distribution we wish to approximate with q (paramerterized by *variational parameters*, $\boldsymbol{\nu}$) in order to do approximate inference. Then, expanding the log and using Bayes' Theorem, we can write

$$\begin{aligned} KL(q(\mathbf{z};\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{y})) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log q(\mathbf{z};\boldsymbol{\nu}) - \log p(\mathbf{z}|\mathbf{y})] \\ &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log q(\mathbf{z};\boldsymbol{\nu}) - \log p(\mathbf{z},\mathbf{y})] + \log p(\mathbf{y}), \end{aligned} \tag{2.67}$$

where $\mathbf{y}$ are some observations, and $\mathbf{z}$ are some latent variables. This shows the dependence on the marginal likelihood and therefore the general intractability of computing the KL divergence directly. The tractable part of the above is called the *Evidence Lower Bound* (ELBO), defined as

$$\mathcal{L}(q) = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log p(\mathbf{z},\mathbf{y}) - \log q(\mathbf{z};\boldsymbol{\nu})], \tag{2.68}$$

which, again using Bayes' Theorem, can be written as

$$\begin{aligned} \mathcal{L}(q) &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log p(\mathbf{z}) + \log p(\mathbf{y}|\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu})] \\ &= \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log p(\mathbf{y}|\mathbf{z})] - KL(q(\mathbf{z};\boldsymbol{\nu})||p(\mathbf{z})). \end{aligned} \tag{2.69}$$

As an objective function, maximizing the ELBO encourages q to place its probability mass on the expected likelihood, $\mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log p(\mathbf{y}|\mathbf{z})]$, while being regularized by the second term to keep the mass relatively close to the prior, $p(\mathbf{z})$. So why is it called the evidence lower bound? Recall that $KL(\cdot||\cdot) \geq 0$. So from equations 2.67 and 2.68, we see that

$$\log p(\mathbf{y}) = KL(q(\mathbf{z};\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{y})) + \mathcal{L}(q), \tag{2.70}$$

and so

$$\log p(\mathbf{y}) \geq \mathcal{L}(q). \tag{2.71}$$

This establishes $\mathcal{L}(q)$ as the lower bound on the log likelihood of the data $\log p(\mathbf{y})$. Furthermore, since $\log p(\mathbf{y})$ is fixed, maximizing $\mathcal{L}(q)$ is equivalent to minimizing the KL divergence $KL(q(\mathbf{z};\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{y}))$, which itself is intractable. The caveat, however, is that the ELBO is in general not convex in the variational parameters, and so unlike MCMC methods, global optimization is typically not possible [35].

As in other methods of inference, the first task is to select a $p(y, z)$ (and therefore a $p(y|z)$ given a chosen prior $p(z)$) that is flexible enough to represent the underlying structure and dependencies in the data. In variational inference, we also need to select a variational family $q(z;\nu)$ whose posterior density $q(z|y;\nu)$ is able to adequately approximate the true posterior.

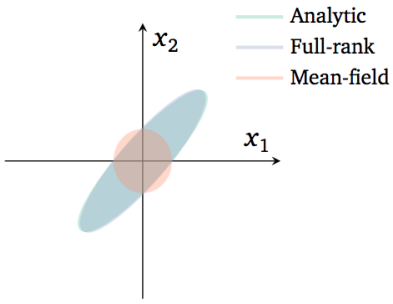

Figure 2.8: Example of a posterior distribution with nonzero covariance, exposing the weakness of expression in the MF approximation relative to a full-rank approximation, from [36].

The simplest possible place to start would be choosing a fully factorized distribution such that the latent variables are mutually independent under q:

$$q(\mathbf{z}; \boldsymbol{\phi}) = \prod_{j=1}^{m} q_j(z_j; \phi_j). \tag{2.72}$$

This is referred to as the *Mean-Field (MF) variational family*, where each latent variable is assigned its own variational factor, ϕ_j, which are optimized in the maximization of $\mathcal{L}(q)$. This approximating family can describe, for example, a mixture of Gaussians, where the mean-field factors correspond to the mixture components. However, by construction the covariance between components is ignored given the assumption of mutual independence of the factors. Figure 2.8 shows an example of the limitations of the mean-field approximation.

Despite the lack of expressiveness, the fully factorized assumption is often a good place to start, and allows simple optimization of the ELBO, since each factor can be optimized independently of the others. This is referred to as coordinate ascent, and was traditionally how variational inference was done. Convergence to a local optimum is guaranteed for a large class of commonly used models, called conditionally conjugate models [35].

As a brief case-study into the usefulness of variational inference, the following section explores inference in nonparametric mixture models. This inference framework is used in Chapter 5, where it is used to establish data-driven priors for the factorial HDP-HSMM, as well as to monitor incoming data for novel events using a concept known as Bayesian surprise.

Variational Inference in Non-parametric Mixture Models Extending the discussion in Section 2.2.2, we explore variational inference in non-parametric mixture models,

introduced first by David Blei and others [37]. The mean-field variational approximation for these cases is proposed to take the following form:[5]

$$q(\nu, \theta, \mathbf{z}) = \prod_{k=1}^{K-1} q_{\gamma_k}(\nu_k) \prod_{k=1}^{K} q_{\tau_k}(\theta_k) \prod_{n=1}^{N} q_{\phi_n}(z_n) \tag{2.73}$$

Here, $\{\gamma, \tau, \phi\}$ are the variational parameters subject to coordinate ascent optimization. q_γ are beta distributions parameterized by the individual stick lengths, ν_k. q_τ are in our case Gaussians parameterized by $\theta_k = \{\mu_k, \Sigma_k\}$, although extension to general exponential families is possible. q_{ϕ_n} are multinomial, parameterized by indicator variables z_n, which denote the component to which the observation x_n is assigned. To speed up inference, a truncation on the maximum number of possible states is imposed on the variational approximation, similar to truncation in the weak-limit introduced for HDP-HSMMs. This value, K, is itself a variational parameter which can be fixed or optimized with respect to the ELBO. Important later for Bayesian surprise, the resulting posterior predictive distribution under this approximation can be neatly factored as expectations with respect to the variational distribution, q:

$$p(x_{N+1}|x_1, \ldots, x_N, \alpha, G_0) \approx \sum_{k=1}^{K} \mathbb{E}_q[\pi_k] \mathbb{E}_q[p(x_{N+1}|\theta_k)], \tag{2.74}$$

where the DP prior parameters α and G_0 were introduced in Section 2.2.1.

As discussed in Section 1.1, the separation of NILM into disaggregation and labelling allows flexible methods like the factorial HDP-HSMM to model home appliances with data-driven priors. This is in contrast to trying to specify priors that are general enough to translate disaggregation performance from household to household, while being specific enough to translate labelling performance within a given home. Using data-driven priors compensates for these intrinsic weaknesses. Although the labelling problem persists, it can now be outsourced to methods with established success in combining abstract features in a highly nonlinear way. The following section provides an in-depth background into deep learning methods and the architectures used for classification in Chapter 4.

2.4 Deep Learning for Classification

There's no mistaking that neural networks and deep learning techniques have inundated research efforts in a wide array of fields: computer vision such as automated driving or object recognition in medical imaging, machine cognition and neuroscience in general, signal processing, financial analysis, and many others. These techniques have become ubiquitous

[5]In what follows, recall that "non-parametric" is a slight misnomer in that it means the model is not constrained to have finitely many parameters, not that it has no parameters at all.

for good reason; non-linear combinations of inputs can often provide richer, more general information for regression and classification tasks. This is especially true when signal to noise ratio is poor [38]. Despite their often impressive performance, disentangling their learned parameters in order to gain physical insight into a system or problem is often difficult or impossible. In medicine, for instance, understanding the combination of factors predicting a given disease can be as or more important than determining whether a person has the disease given a set of factors. Genetic sequencing is an important example of this problem, where researchers fundamentally want to understand the interconnectedness of DNA sequences in producing macroscopic consequences. Nevertheless, there are many problems for which a physical understanding is unnecessary. NILM is arguably one of those problems; there are few cases for which the exact combinations of signal features characterizing one type of appliance from all the others is terribly enlightening. In this section, we will examine in reasonable detail the evolution and structure of modern neural networks, and describe the architecture designed to provide generalized appliance classification. The goal is that unsupervised techniques could be used in conjunction with this pre-trained network in order to reliably label disaggregated but unknown appliances.

2.4.1 Perceptrons to Neurons to Neural Networks

In the early days of cognitive science of perception, a model for the neuron was put forth by McCollough and Pitts in 1943 [39]. Their model was a simple structure that took in a vector of binary inputs and produced a binary output. The inputs were weighted according to their relative importance, and compared to a threshold value that the weighted sum of inputs must exceed in order to activate the neuron and produce an output of 1. In other words, the output for a given threshold b and set of weights w is determined by

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j - b \leq 0 \\ 1 & \text{if } \sum_j w_j x_j - b \geq 0, \end{cases} \tag{2.75}$$

Clearly, real neurons are not only activated by multiple input neurons, but their output is also connected to many other neurons. Another aspect we might like to modify in our extension of these simple models to real problems is to be able to both accommodate inputs as well as produce outputs in a range of values between 0 and 1, rather than the binary requirement we set. What we need is an *activation function*, a nonlinear function constrained between 0 and 1, taking real valued input from the weighted sum $w \cdot x - b$. There are a huge number of such functions, but the most popular is known as the *sigmoid neuron*, which is defined as

$$\sigma(z) = 1/(1 + e^{-z}), \tag{2.76}$$

for some input z. This is shown in Figure 2.9.

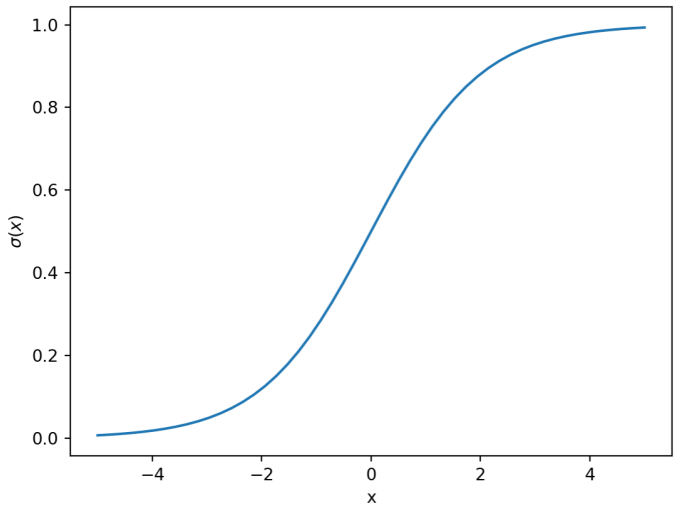

Figure 2.9: Sigmoid activation function as described in text.

A typical 'vanilla' network with a single hidden layer (i.e., a layer not designated as the input or output) is shown in Figure 2.10, which contains weights between each neuron in neighbouring layers, and a bias/threshold for activation of each neuron. These weights and biases can be randomly initialized and the output of the network measured against some objective function known as the *cost* or *loss* function. It is this cost function that determines not only how the network is performing, but—by its gradients with respect to each parameter—how to improve the performance of the network. The process by which the parameters in a network are adjusted to minimize the cost function is called *backpropagation*. The following section provides a more detailed look at the flow of information through the network as well as how backpropagation works to optimize network parameters.

2.4.2 Backpropagation and Cost Minimization

We describe the vector of activations in an arbitrary layer of the network l by the weighted sum of the previous layer activations (a matrix-vector product) modified by a bias vector and passed through the nonlinear activation function:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{2.77}$$

Element-wise, each neuron takes as input the weighted activations of each neuron in the previous layer, $\sum_k w^l_{jk} a^{l-1}_k$. The activation of each l^{th} layer neuron is then measured against its corresponding bias, b^l_k, and is finally compressed by the sigmoid activation function to produce an activation in the range $[0, 1]$. For convenience, we will refer to the uncompressed activation (i.e., the weighted input) as $z^l = w^l a^{l-1} + b^l$. From a randomly initialized set of weights and biases, the output of the network is produced, propagating z^l from

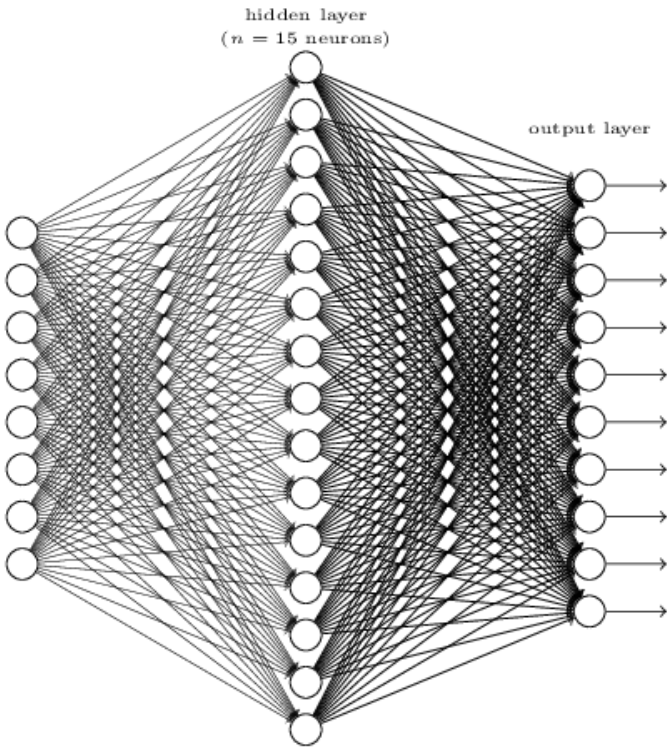

Figure 2.10: Example of simple neural network with single hidden layer. Taken from [40].

$l = 2, 3, \ldots, L$. At the output layer, we know during training what we would like the output to be. We can calculate the cost C associated with each neuron's output in the final layer L relative to its desired output, but we also want to know how to tweak the various weights and biases so as to reduce that cost. To determine how to adjust the network parameters, we might try to compute the partial derivative of the cost with respect to each neuron output in the output layer. It turns out to be algebraically simpler to instead compute the partial of the cost with respect to the weighted input, z^L, since the activation (e.g., the sigmoid function) is a monotonically increasing function, meaning $\partial C/\partial z^L$ and $\partial C/\partial \sigma(z^L)$ will be concurrently minimized. We can therefore define the error in the output layer as

$$\delta_j^L = \frac{\partial C}{\partial z_j^L}. \tag{2.78}$$

By the chain rule, we can rewrite the above in terms of the output activations:

$$\delta_j^L = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}, \tag{2.79}$$

but since $a_k^L = \sigma(z_k^L)$, the sum collapses to a single term where $k = j$, and since we also have that $\partial a_k^L/\partial z_j^L = \sigma'(z_j^L)$, equation 2.79 becomes

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L). \tag{2.80}$$

In vectorized form this can be written as

$$\delta^L = \nabla_a C \odot \sigma'(z^L), \tag{2.81}$$

where $\odot$ indicates the Hadamard/element-wise product. Additionally, we can use the chain rule as we did in equation 2.79 to factor an arbitrary layer's error into the next layer's weighted input:

$$\delta_j^l = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}, \tag{2.82}$$

and since by definition, $z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1}\sigma(z_j^l) + b_k^{l+1}$, we have

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}\sigma'(z_j^l),$$

which gives us a recursion on the errors from one layer to the next:

$$\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l). \tag{2.83}$$

This can again be written in vector form to give us the second backpropagation equation

$$\delta^l = \left((w^{l+1})^T \delta_{l+1}\right) \odot \sigma'(z^l). \tag{2.84}$$

If we instead expand equation 2.78 in terms of the biases b^l for an arbitrary layer l, we have

$$\delta_j^l = \sum_k \frac{\partial C}{\partial b_k^l} \frac{\partial b_k^l}{\partial z_j^l}, \tag{2.85}$$

we quickly see that since $\partial b_j^l / \partial z_j^l = 1$, we are left with

$$\delta_j^l = \frac{\partial C}{\partial b_j^l} \tag{2.86}$$

Finally, we can also expand equation 2.78 in terms of the weights w^l for any layer l:

$$\delta_j^l = \sum_k \frac{\partial C}{\partial w_{jk}^l} \frac{\partial w_{jk}^l}{\partial z_j^l}. \tag{2.87}$$

We can write $w_{jk}^l = (z_j^l - b_j^l)/a_j^{l-1}$, so that $\partial w_{jk}^l / \partial z_j^l = 1/a_j^{l-1}$, giving us the fourth backpropagation equation:

$$\delta_j^l a_k^{l-1} = \frac{\partial C}{\partial w_{jk}^l}. \tag{2.88}$$

A pseudo-algorithm borrowed from [40] for training a network of arbitrary length using backpropagation is shown in Algorithm 4.

Algorithm 4 TRAINING A NN

```
Input a set of training examples
for each training example x do
    for each l = 2, 3, ..., L do
        z_x^l = w^l a_x^{l-1} + b^l
        a_x^l = σ(z_x^l)
    end for
    compute δ_x^L = ∇_a C_x ⊙ σ'(z_x^L)
    for each l = L − 1, L − 2, ..., 2 do
        δ_x^l = ((w^{l+1})^T δ_x^{l+1} ⊙ σ'(z_x^l).
    end for
end for
```

Using the four backpropagation equations 2.81, 2.84, 2.86, and 2.88, we compute the required nudges to each and every weight and bias so as to push the cost function toward its minimum. As presented, however, what is being minimized is the cost associated with a single training example, whereas the true cost is assumed to be an average over all training examples. To find the global minimum of the error surface, we would need to average the partial derivatives over every training example. This is prohibitively expensive for large training sets, as is often the case in neural networks. Instead, what is often done is *stochastic gradient descent*, where the gradients of the cost function are approximated by taking small batches of training examples. This approximation to the gradient is unlikely to point exactly to a minimum, so the updates that we apply to the weights and biases as a result of the batch cost gradient are modified by a hyperparameter known as the *learning rate* and denoted by η. There are much more sophisticated approximations to the cost gradient that are robust to certain weaknesses of stochastic gradient descent, which will be explored shortly. More formally, we update our weights and biases by

$$w^l \leftarrow w^l - \eta \frac{\partial C}{\partial w^l} \tag{2.89}$$

$$b^l \leftarrow b^l - \eta \frac{\partial C}{\partial b^l}, \tag{2.90}$$

or, using equations 2.86 and 2.88 and using stochastic gradient descent, we have

$$w^l \leftarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T \tag{2.91}$$

$$b^l \leftarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}, \tag{2.92}$$

for all training examples $x = 1, 2, \ldots, m$ in the mini-batch. This is completed for all mini-batches in the training data set, and finally for all epochs of training. The result, assuming convergence, is a set of weights and biases minimizing the training error.

2.4.3 Recent Issues and Their Solutions

More recently, significant work has been done on implementation choices such as the cost function, activation function, optimization procedure, etc. to make neural nets better adapted to certain problems and more robust against computational issues. The following section provides a survey of these advancements.

Neuron Saturation

Since $a = \sigma(z) = w \cdot x + b$, then in equations 2.89 and 2.90 it's clear that the updates to the weights and biases—which constitutes the learning of the network—will be proportional to the differential of the activation function, σ'. Because of this dependence, if a neuron is either highly activated or close to zero, this derivative term will be small, at least for the sigmoid activation. Adapting the weights of a saturated neuron (i.e., a neuron whose activation is close to 1 or 0) is therefore difficult, and slows the learning of a network. There are two solutions we might try to solve this issue. The first is to adopt a cost function such that the partial derivatives of the cost no longer depend on the derivative of the activation function.

Recall that the sigmoid activation function is defined as $\sigma(z) = 1/(1 + e^{-z})$, and so $\sigma'(z) = \sigma(z)(1 - \sigma(z))$. Glossing over the details, it turns out that selecting a (single neuron) cost of the form

$$C = -\frac{1}{n} \sum_x \left[y \log(a) + (1 - y) \log(1 - a) \right], \tag{2.93}$$

(for a desired output y) results in the σ' term cancelling out, and the partial derivatives with respect to the weights and biases become

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j (\sigma(z) - y) \tag{2.94}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y). \tag{2.95}$$

This formulation of the cost is called the *cross-entropy* cost, where now the learning rate of a neuron is directly controlled by its output error, $\sigma(z) - y$, and doesn't saturate as with quadratic cost where the derivative term remains.

The other approach we might like to take to address the saturation and learning slowdown problem is to modify the activation function. There is nothing particularly special about the sigmoid activation; many different types exist with different properties. The output values need not even be normalized, which is in general the case for the sigmoid function. If we wanted to perform tasks such as logistic regression, however, a normalized output layer would be useful since we can interpret the output directly as a probability distribution. One way to do this is the *softmax* function, where at the output layer L, we have $a_j^L = s(z_j^L)$ where $s(\cdot)$ is defined as

$$s(z_j^L) = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}, \tag{2.96}$$

which can quickly be shown to sum to 1 over the output layer. With this modified activation function, however, the condition for cancellation is no longer met. It turns out that a *log-likelihood* cost function defined as

$$C = -\log a_y^L \tag{2.97}$$

allows the same cancellation of the softmax function derivative term. In the log-likelihood cost, the activation a_y^L is the estimate of the probability (i.e. the activation) of a training point belonging to class y. As with the cross-entropy cost, the log-likelihood cost behaves as we would expect: when the network is performing well, the activation for the correct class y is large and the cost is small, and vice versa for a poorly performing network.

Although these methods solve the issue of saturated output neurons, the neurons in the hidden layers can remain saturated [40]. Moreover, initialization of the weights and biases randomly according to a standard Gaussian distribution results in a high likelihood of saturated neurons. We can see this by noting that the activation of a neuron is the summed weighted activations of those in the previous layer. If these are all random Gaussians of zero mean and unit variance, then the activations of the next layer neurons are distributed as a Gaussians with a much larger variance, such that they're all very likely to be much larger than 1 or much smaller than -1, resulting in saturation. This issue can be solved in part by restricting the variance of the initialization distributions, say by setting the variance to $1/(n+1)$, where n is the number of inputs. This squishes the resulting distribution on each weight and bias and therefore the resulting activations of the next layer.

Regularization

With all these weight and bias parameters being modified to fit the training data, overfitting to the point of poor generalization is a major problem in neural networks. The number of parameters grows rapidly with the depth (or number of hidden layers) of the network, and so modern complex network structures often have enormous numbers of free parameters. An obvious strategy is to use huge amounts of highly diverse training examples, but this is

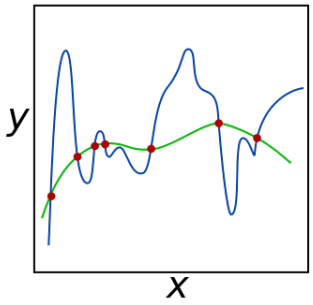

Figure 2.11: Example of regularization for polynomial fit parameters. Work by Nicoguaro, taken from wikipedia.org/wiki/Regularization_(mathematics) under creative commons.

often not feasible due to availability of high quality training data. As a result, *regularization* techniques have been developed to attempt to deal with the issue of overfitting. A simple notion of regularization is sometimes referred to as temporal regularization, in which small validation data sets are used to evaluate the extent of overfitting in the training data. A technique known as early-stoppage is sometimes used to stop training when the accuracy on the validation data set has saturated/plateaued. Another strategy is to modify the cost function to incentivize keeping the parameters as small as possible. If the model fit defined by the parameters only weakly depends on certain weights and biases, the result will be a lower variance estimator and therefore a model more robust to outliers, reducing the likelihood of overfitting the training data. An example of such a strategy for the case of a polynomial fit is shown in Figure 2.11. Two such methods are known as ℓ_1- and ℓ_2-regularization. Regardless of cross-entropy, log-likelihood, or quadratic cost (denoted C_0), ℓ_1- and ℓ_2-regularization take the following respective forms:

$$C = C_0 + \frac{\lambda}{n}\sum_w |w| \tag{2.98}$$

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2, \tag{2.99}$$

which can be interpreted as modifying the update equations to find a balance between reducing the cost and keeping either the ℓ_1 or ℓ_2 norm of the weights as small as possible. The tradeoff between the two is determined by the nonzero hyperparameter λ. The respective updates for the weights in ℓ_1 and ℓ_2 regularization are now

$$w \leftarrow w - \eta\frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n}\text{sgn}(w) \tag{2.100}$$

$$w \leftarrow w - \eta\frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n}w. \tag{2.101}$$

Another common technique is known as *dropout*, which might be thought of as similar to the technique of boosting in random forests, in which a random subset of features is used in determining decision boundaries. In dropout, a random subset of neurons in a layer are deleted, and backpropagation is used to update the remaining weights and biases as usual. After repeating this process, the final output of the network (when adjusted to account for the proportion of dropped neurons) essentially acts as an ensemble of smaller networks. In both dropout and boosting, the intent is to produce an estimator or ensemble of estimators with lower variance and therefore reduce overfitting.

Lastly, expanding the training data set by means of translation, rotation, or other modifications allow a much richer set of data to be easily and cheaply produced. These modified training examples reinforce a level of invariance of the output to these slight changes in input, again helping with overfitting.

Optimization and Behaviour of the Gradient

It turns out that as the number of layers increases in a network, the rate of learning in the earlier layers can be driven to zero or infinity [40]. This is referred to as the vanishing or exploding gradient problem, respectively. Even with more intelligent initialization as mentioned above, larger networks or unlucky initializations in smaller networks can still lead to these problems. The instability originates in the product of terms from all the outer layers involved in backpropagation, which is simply a consequence of the chain rule. This makes deep neural networks more difficult to train, despite their potential to outperform shallow networks. To deal with the exploding and vanishing gradient issues in backpropagation a number of alternatives/extensions of gradient descent have been proposed. The most ubiquitous approach is *Adam* optimization, an acronym for Adaptive Moment Estimation. Adam is an extension of gradient descent involving two key modifications: the concept of momentum and the concept of *Root-Mean-Square* (RMS) propagation. We will briefly discuss these modifications and how they improve the exploding/vanishing gradient problem.

The notion of momentum in gradient descent was introduced in order to improve convergence and smooth out oscillations in the approach to the error surface minima [41]. Traditional gradient descent computed over batches provides an imperfect approximation to the error surface, and as a result convergence is often slow and can be unstable if the step size (or learning rate) is too large. In gradient descent with momentum, we compute as usual the gradients of the cost with respect to the weights and biases for a given batch i of training samples. Then, we compute a velocity/momentum term

$$m_w^{(t+1)} = \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w C_i^{(t)} \tag{2.102}$$

$$m_b^{(t+1)} = \beta_1 m_b^{(t)} + (1-\beta_1)\nabla_b C_i^{(t)}, \tag{2.103}$$

for a non-negative hyperparameter $0 \leq \beta_1 \leq 1$. Instead of updating the weights and biases with the cost gradient weighted by a learning rate as before, we now use these momenta:

$$w^{(t+1)} \leftarrow w^{(t)} - \alpha m_w^{(t+1)} \tag{2.104}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \alpha m_b^{(t+1)}, \tag{2.105}$$

for a stand-in hyperparameter α denoting the learning rate. The intuition here is that the actual gradient of the weights and biases at a given iteration step are contributing less to the weight update by taking the exponentially weighted average from previous iterations. This drives down the oscillations due to imperfect batch estimation of the true gradient, and allows a faster, more direct approach to the error surface minima.

In RMS propagation, often called "RMSprop", we have a very similar formulation, where we define the exponentially weighted average of the second moments of the gradients as

$$\nu_w^{(t+1)} = \beta_2 \nu_w^{(t)} + (1 - \beta_2)\left(\nabla_w C_i^{(t)}\right)^2 \tag{2.106}$$

$$\nu_b^{(t+1)} = \beta_2 \nu_b^{(t)} + (1 - \beta_2)\left(\nabla_b C_i^{(t)}\right)^2, \tag{2.107}$$

where again, $0 \leq \beta_2 \leq 1$. Note that the only difference is that these ν's are quadratic (element-wise) in the gradients, leading to update equations

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\alpha}{\sqrt{\nu_w^{(t+1)}}} \nabla_w C_i^{(t)} \tag{2.108}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \frac{\alpha}{\sqrt{\nu_b^{(t+1)}}} \nabla_b C_i^{(t)} \tag{2.109}$$

RMSprop allows better numerical stability than standard gradient descent, and is robust to larger choices of learning rate, α, leading to faster convergence [41]. The parameters β_1 and β_2 are often called the forgetting factors, since they are chosen to be less than unity and therefore drive to zero the effect of the updates on previous values of the gradients. This gives some intuition as to why these optimization techniques help with the exploding/vanishing gradient problems, since after some number of layers the gradients stop contributing in the backpropagation.

Finally, Adam optimization is just a combination of the above two methods, where for each iteration we compute equations 2.102, 2.103, 2.106, and 2.107 for the momenta and second moments of the gradients. A small modification to those equations is often introduced to deal with *bias correction*, where initialization of the hyperparameters β_1 and β_2 can result in very small initial updates, especially if they are close to unity (as

is recommended for good performance) [41]. In bias correction of exponentially weighted averages, we update the momenta and second moments at each iteration t by, for example

$$\hat{\nu}_w = \frac{\nu_w^{(t+1)}}{(1-\beta_2^t)}, \tag{2.110}$$

and similarly for $\hat{\nu}_b$ as well as $\hat{m}_w$ and $\hat{m}_b$ (with the substitution of β_1 instead of β_2). Then, the updates to the weights and biases in Adam optimization are as follows:

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\alpha \hat{m}_w}{\sqrt{\hat{\nu}_w} + \epsilon} \tag{2.111}$$

$$b^{(t+1)} \leftarrow b^{(t)} - \frac{\alpha \hat{m}_b}{\sqrt{\hat{\nu}_b} + \epsilon}, \tag{2.112}$$

where we add the small constant parameter ϵ to prevent division by zero due to rounding error.

These modifications to gradient descent provide much more rapid convergence and a resilience to the exploding/vanishing gradient problem by virtue of the forgetting factors β_1 and β_2. The overall learning rate, α, is often tuned using validation data sets [40].

2.4.4 More Advanced Neural Net Structures

Aside from the most obvious extension from simple neural networks to 'deep' neural networks involving more hidden layers, there are two advancements in the deep learning space that respectively dominate image recognition tasks and natural language processing or time series tasks. These networks are known as *Convolutional Neural Networks* (CNNs) and *Recurrent Neural Networks* (RNNs). Given that the architecture used in Chapter 4 is based on CNNs and the related notion of residual networks, we will restrict our discussion to those.

Convolutional Neural Networks

As mentioned, one well-established application of CNNs is in image classification, where a set of characterising features are learned from many training examples with appropriate labels. In the types of networks so far considered, passing a single megapixel image into a network of say 1000 neurons in the first hidden layer will already have a billion weights and one thousand biases. The amount of computational resources and time spent optimizing that many free parameters is immense, and exponentially more so for any modern depth network. Furthermore, using such a fully-connected structure completely throws away important spatial information that could make classification far easier, since every input neuron passes information to each hidden neuron. Recognizing spatial relationships is a key way that the human visual system learns, and CNNs are arguably an attempt at emulating this.

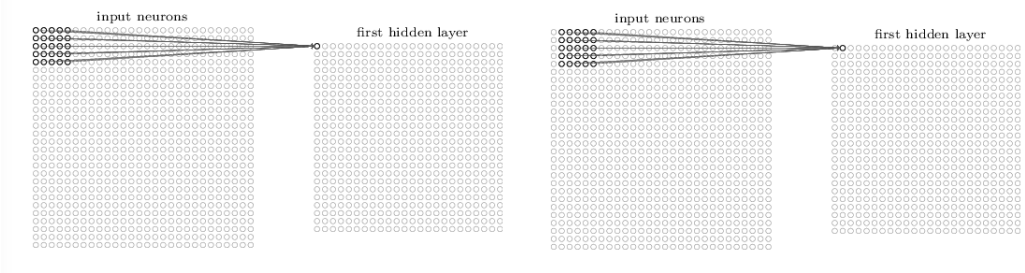

Figure 2.12: Process of convolution of a LRF for a stride of 1. Taken from [40].

The key to CNNs is in the name. They are fundamentally convolution filters, but filters for which the features are learned rather than pre-designed. The *Local Receptive Field* (LRF) is a parameter of CNNs defining the size of the window to be convolved along the array of input neurons as in Figure 2.12. In this figure, a *stride length* of 1 is shown, indicating the number of neurons shifted in producing each neuron of the hidden layer. In this way, the densely packed parameter space of the fully connected structure explored previously is replaced with a set of neurons in the hidden layer in which each is only connected to those input neurons in its LRF. Furthermore, each weight and bias connecting the hidden neurons to their respective neurons in their LRF are shared and set to be identical, forcing a particular hidden layer—called a feature map—to learn the same local feature. It's these shared weights and biases that are referred to as a filter. A stack of such feature maps defines the hidden layer for a CNN, which are differentiated from one another based on the random initialization of the weights and biases. To make explicit the relation to convolution, the output activations for a given feature map are defined by

$$a^{(\text{out})} = \sigma(b + w * a^{(\text{in})}), \tag{2.113}$$

where $*$ indicates the convolution operation.

Another important concept in CNNs are *pooling layers*. Pooling layers are a way to compress the information in the output activations contained in the feature maps. By using max-pooling, for example, we extract the maximum over regions in the feature map of a specified size, say 2x2 regions. In this way, more exact spatial information learned by the feature maps is supplemented by coarser spatial relationships, allowing a reduction in dimensionality and faster training. Similar to regularization, one can also compute ℓ_2-pooling, or mean-pooling, etc.

Residual Networks

Despite all the efforts to reduce the issue of instability in the gradients, modern networks are increasingly deep, and this instability again manifests itself in a reduction of accuracy/performance on the training set for a deeper-than-optimal network [40]. Intuitively, we

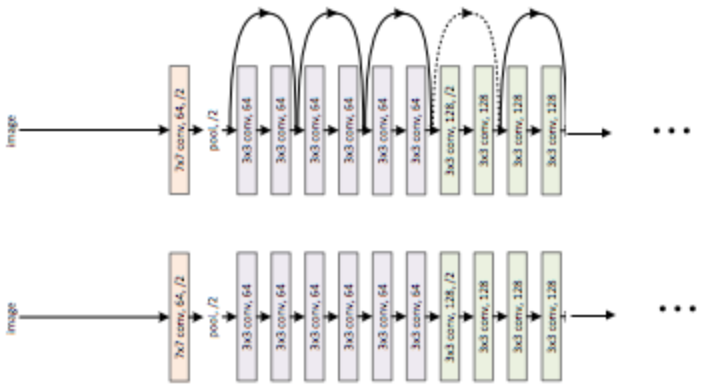

Figure 2.13: Example of a Res-Net structure (top) compared to a "plain" network (bottom). Adapted from [42].

should expect that aside from additional computation time, at the very least adding new layers should maintain the existing performance on the training set. In other words, at the depth after which performance begins to decline, we should expect that at the very least, the deeper layers learn to produce a so-called *identity block*, where the input to these layers passes through unchanged. This would ideally maintain the performance of the network as it becomes increasingly deep. To accomplish this, we might imagine adding the input of a layer to the weighted input two layers down, after the original layers output has been passed to the next layer. This is known as a residual network or *Res-Net*, and is shown in Figure 2.13. In this way, we can nudge the network to be able to straightforwardly bring the residual to zero, rather than having to learn a whole set of weights and biases to produce an identity block. This can be seen from finding the activation $a^{(l+2)}$. If we use *Rectified Linear Unit* (ReLU) neurons in the network, for which $g(x) = \max(0, x)$, then we can write that

$$a^{(l+2)} = g(w^{(l+2)}a^{(l+1)} + b^{(l+2)}).$$

In order for the activations $a^{(l+2)}$ to be the same as $a^{(l)}$ (i.e., for the network to learn an identity block and maintain performance for increased depth), a quite complicated adjustment of parameters across layers in backpropagation would be required. If instead we provide a *skip connection* from $a^{(l)}$ to $a^{(l+2)}$ before the nonlinear operation, we now have

$$a^{(l+2)} = g(w^{(l+2)}a^{(l+1)} + b^{(l+2)} + a^{(l)}). \tag{2.114}$$

With this skip connection, the network can very quickly learn that setting $w^{(l+2)}$ and $b^{(l+2)}$ to zero results in $a^{(l+2)} = g(a^{(l)}) = a^{(l)}$ (for ReLU activation). In other words, in the limit that the identity block is optimal, zeroing the weights and biases for the $(l+2)$ layer is all that's required for a Res-Net.

These Res-Net blocks are often cascaded together, allowing a sort of sequential protection against the degradation of training accuracy due to gradient instability [41]. There is a restriction due to the fact that the dimensions of the input and output of a Res-Net block must be identical, at least in the formulation described here. This allows for typical dense connected layers or something like a single stride convolution, also called a 'same' convolution. A slight modification can be used in case of dimensionality mismatch by substituting $a^{(l)}$ in equation 2.114 with $W_s a^{(l)}$, where W_s is a matrix either meant to pad $a^{(l)}$ with zeros, or it could be a matrix of weights to be learned (sometimes called a 'highway net'). Either way, the result is to match the dimensionality of $a^{(l+2)}$.

The proposed appliance trace classification network makes use of the components mentioned in this background section. The selected architecture will be discussed in Chapter 4.

2.5 Metrics for NILM evaluation

Since the initial efforts to disaggregate home power demand, the relevant performance metrics for NILM have been debated. A brief summary of the most prominent existing methods for evaluation are provided, borrowing from the previous work compiled in [43]. Some are metrics concerning classification accuracy, while others are metrics concerning estimation accuracy. In the notation that follows, $\hat{y}_i(t)$ refers to the inferred power draw for appliance i at time t, while $y_i(t)$ refers to the ground truth for appliance i at time t. If sub-metered ground truth y_i is not available, the total aggregate can be compared with the sum over inferred appliance traces in the estimation accuracy metrics.

Normalized Disaggregation Error

$$\text{NDE} = \frac{\sum_{i,t} |\hat{y}_i(t) - y_i(t)|}{\sum_{i,t} y_i^2(t)} \tag{2.115}$$

Estimation Accuracy

$$\text{Est. Acc.} = 1 - \frac{\sum_{i,t} |\hat{y}_i(t) - y_i(t)|}{2 \cdot \sum_{i,t} y_i(t)} \tag{2.116}$$

Root Mean-Squared Error

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1,t=1}^{m,T} (\hat{y}_i(t) - y_i(t))^2}{T}}, \tag{2.117}$$

for m individual appliances.

Mean Absolute Error

$$\text{MAE} = \frac{\sum_{i=1,t=1}^{m,T} |\hat{y}_i(t) - y_i(t)|}{T} \tag{2.118}$$

Basic Accuracy

$$\text{Acc.} = \frac{\text{correct}}{\text{correct} + \text{incorrect}} \tag{2.119}$$

F1-score

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{2.120}$$

where, for binary classification, $precision$ = true positives/(true positives + false positives), and $recall$ = true positives/(true positives + false negatives). This metric can be extended to multistate appliances by averaging over true positives for each relevant state.

Chapter 3

Related Work

Since its inception, inaugurated by the seminal work in [44], non-intrusive appliance/load monitoring has seen a burst of activity, both in methods for solving the problem and in the amount of public data available. The following chapter describes a small subset of current research, focusing on more classical approaches such as HMMs due to the scope of this thesis. Finally, a brief summary of existing datasets is provided.

3.1 Classical Approaches

As might be evident given the extensive background provided in Chapter 2, HMMs have proven effective in NILM, and several variants have been investigated in depth to date. In [45], the authors present an *additive-* and *difference*-factorial HMM structure, where emissions of the individual HMM chains correspond to contributions to the aggregate signal and sharp changes in the signal, respectively. This allows a complementary inferential framework wherein the aggregate output is followed by the additive model, while the difference model captures the changes in the system directly. To compensate for events or appliance states for which no factorial chains exist, the authors introduce a flexible "generic-component", along with *Total Variation (TV) Regularization*. TV regularization places an ℓ_1-norm constraint on the additive and difference models, thereby encouraging sparsity in the inferred contributions of the chains to an observed aggregate value, or to a change in the aggregate. Inference in this model involves relaxing the constraints on an exact MAP inference scheme, where convexity in the log-likelihood of the model is recovered by two steps. First, an intuitive constraint is what the authors refer to as the *one-at-a-time* condition; there is at most one HMM state change at a given instant. Depending on the sampling rate of the data in consideration, this is usually valid with high probability. Second, they relax the condition that indicator variables for hidden states and transitions between states be in $\{0, 1\}$, allowing the problem to be recast into a mixed-integer linear optimization problem, which can be handled by existing solvers. Jointly optimizing the posterior over the hidden variables for both the additive and difference models is similar in spirit to imposing con-

straints over the possible duration space in the factorial HDP-HSMM ($\mathcal{D}$, see Section 2.3.2). As will be discussed in Chapter 4, the difference signal can be used directly to constrain the conditional posteriors of the state sequence and associated durations.

A further constraint for this model, termed *Signal Aggregate Constraints*, are introduced in [46]. This constraint can be cast in a similar posterior regularizing framework as was done in the *one-at-a-time* condition in [45]. The idea is to modify the MAP inference optimization problem to constrain the total energy of each factorial component to an approximate average value over some span of time. This value can be extracted from data or from appliance usage statistics over average national consumption, for example. Over a set of 100 homes, normalized disaggregation error (see Section 2.5) for a single test-day given 15-26 days of training data, the authors showed that signal aggregate constraints improved disaggregation performance significantly, even with appliance energy consumption constraints averaged across all 100 homes.

In [47], the authors construct factorial chains for individual appliance operating states, but then combine these chains into a hierarchical Markov chain governing the dynamics between the factorial chains. This sidesteps the issue of the loss of state dependencies in factorial models, and is a relatively simple but interesting extension that could be made to the present work.

An alternative method to capture load dependencies is of course to avoid factorial modelling entirely. However, modelling all state combinations in a house is exponential in the cardinality of the system, i.e., the total number of appliance states. This leads to large storage requirements for transition and emission matrices, as well as time complexity for navigating paths through the state space during inference via the Viterbi algorithm or similar methods. In [13], however, the authors introduce the *Super-State Hidden Markov Model* (SSHMM). As mentioned, single chain HMMs involve enumerating every possible combination of load states in the home, which has benefits of preserving load dependency (all loads are represented in a single π matrix), but at the (previously assumed) cost of complexity in run-time and storage. The novelty of this method is in its appreciation of the increased sparsity of the π and ϑ matrices as the state space expands, since many of the possible super-states of the home, along with transitions between them, effectively never occur and can be ignored. This raises questions of ergodicity in the state chain, and whether stochastic approximate inference in nonparametric versions of these models would reliably recover the true posterior for unbounded iterations. Nevertheless, exact inference in parametric models allows this SSHMM method to perform very well and relatively quickly in a supervised setting, and has renewed interest in these models for a non-trivial number of loads in a home.

Finally, the clear motivation for the work contained in this thesis can be found in [9], the background for which has been discussed at length in Chapter 2. Although NILM was treated as an example case for the usefulness of HDP-HSMMs, the authors showed marked

disaggregation improvement (at least in terms of estimation accuracy) over simple factorial HMMs trained using EM methods. This work precipitated significant research in these and related methods such as linear dynamical systems [48].

Other approaches to NILM tend towards more classical signal processing methods. In signal processing such as audio or other communications, a common problem is the unknown mixing of component signals, known as blind source separation. This class of problems, in their most basic form, are characterized by a mixing matrix A that is applied to all individual source signals $s(t) = (s_1(t), ..., s_n(t))^T$, resulting in a combined signal $y(t) = (y_1(t), ..., y_m(t))^T$ such that, for some intrinsic noise source $e(t)$,

$$y(t) = A \cdot s(t) + e(t). \tag{3.1}$$

The NILM task is to "un-mix" the aggregate signal essentially by inverting the above equation to recover the inferred sources $\hat{y}(t) = (\hat{y}_1(t), ..., \hat{y}_{n'}(t))$ via an un-mixing matrix B:

$$\hat{y}(t) = B \cdot y(t) + e'(t), \tag{3.2}$$

for some modelling error, $e'(t)$. In NILM, this system of equations is nearly always underdetermined in realistic use-cases; the number of modelled appliances n' is a small subset of the total number of signals n contributing to the mixed signal. This often permits many equally valid solutions to the system of equations, and so techniques leveraging source independence properties such as principal and independent component analysis are used to narrow the space of potential solutions. As in the discussion of HMMs, constraints on sparsity of representation in the signal are also often used, which penalize solutions including more signal components than necessary. These considerations are often used in *Non-negative Matrix Factorization*, a fruitful area of NILM research [49],[50],[51]. Another constraint used in these and other methods is that appliance contributions to the aggregate signal have binary weight with respect to a particular appliance mode. In other words, an appliance power draw at a particular time is the result of only one internal operating mode. This is one example of integer programming, a class of problems in which some or all of the variables in question are integers; binary, in this case. Integer programming has a rich history in NILM, as we already saw in the conversion of MAP inference in [45] to a mixed-integer linear optimization problem. In [52], the authors impose further constraints and relaxations on the linear integer optimization problem of combining current signatures to match the total current. They constrain the solution ambiguity mentioned above by allowing standby modes for appliances, reinforcing sparsity in the number of proposed appliances that are concurrently activated, and finally by penalizing solutions that violate certain known internal appliance state transitions, such as a clothes washer starting its high spin state before any water pump activation or other wash states. The authors show significant robustness of solutions relative to standard integer programming.

The ill-conditioned nature of the NILM problem is also exposed in this source separation framework. For large condition numbers (defined as $||B||||B^{+}||$, where B^{+} is the pseudo-inverse of B and $||\cdot||$ is a matrix norm, often the Euclidean norm), small changes in the input can result in large changes in the output. In other words, solutions are unstable. Such problems are often difficult to optimize and are prone to computational errors [53]. For this reason, any and all prior knowledge about the component signals should be leveraged in order to constrain the space of potential solutions. This prior knowledge includes signal features such as transients, temporal correlations, human behavioural patterns, etc. Using signal features learned from existing appliance traces for disaggregation is a common method for supervised NILM. For example, Liao et al. built decision trees using the two-dimensional description of rising edge and falling edge for a given event [54]. Over three houses in the REDD data set, they showed a high F-score ($> 85\%$) for decision-tree-based event classification. There are many classification approaches which may include features extracted in time-domain such as derivative information [55], the time-frequency domain via wavelet decomposition [56], harmonic content in the frequency domain via *Fast-Fourier Transform* (FFT) [57][58][59], and many more. Incorporating a wider feature space also allows the use of random forests as in [60].

3.2 Deep Learning Approaches

Deep learning techniques very naturally extend themselves to NILM, although it was not until recently that the first efforts surfaced. Since the initial publication by Kelly et al. [61] approaching disaggregation via a *Long Short-Term Memory* (LSTM) RNN and a denoising auto-encoder method, many other attempts were made. To name a few, many researchers have focused on using RNNs and their modifications [62][63][64], while others have attempted CNN-based methods [65][66][67] or deep dictionary learning [68][69], a technique with similarities to non-negative matrix factorization.

Although these methods are supervised in their training, robust regularization methods are often employed with the goal of preventing overfitting and improving generalization to unseen data. For example, in [70], the authors restricted the number of tunable parameters relative to the existing literature. They also made use of early stoppage with an aggressive patience parameter to terminate training. With these complexity and temporal regularization methods, the authors examined disaggregation performance on unseen homes in the same dataset as well as different datasets. They showed intra- and inter-dataset transferability with minimal performance losses relative to their chosen baseline.

Another realm of deep learning NILM research are generative models, such as the *Generative Adversarial Network* (GAN) and the *Variational Auto-Encoder* (VAE). Both of these methods can operate in an unsupervised fashion, where in lieu of labelled training data, feedback quantifying performance is provided to the network relative to the realism

of network output (e.g., [71],[72]) and the likelihood of the data under the proposed model (e.g., [73]), respectively. The GAN framework has also recently been used for generation of artificial appliance signatures to address the poor representation of the distributions over possible appliance models and aggregate combinations in existing datasets [2].

3.3 Unsupervised/Semi-supervised NILM

As mentioned, GANs and VAEs can operate in the unsupervised and semi-supervised learning setting. Other unsupervised methods often leverage intrinsic differences in the appliances present to cluster extracted features. For example, in [74], steady state reactive and real power values are clustered via a Genetic K-means algorithm, an iterative method to cluster data points often based on minimizing intra-cluster variance. The cluster centroids were assumed to be linear combinations of individual appliances which are further deconstructed. Multi-state appliances obfuscate this process, along with poor discrimination between appliance modes with similar consumption levels. In [75], Motif Mining was used to discover recurrent patterns in the difference signal rather than the aggregate power signal. This method of recurrent sub-graph, or *episode*, detection unsurprisingly proved useful for repetitive appliances with well-defined usage patterns, but performance was unclear for appliances which vary in this episodic description.

Another issue in unsupervised NILM methods alluded to in Chapter 3 is that of appliance labelling in unsupervised methods. In methods such as the HDP-HSMM in [9], labels are provided by probabilistic ranges within the state-space defined by component emissions and duration modes. However, these modes are flexible during inference via Gibbs resampling, leading to potential shuffling in the proposed labels that would not be tracked during inference. Furthermore, the priors provided to the model are highly sensitive to bias; appliance models can be significantly different from one another even in an austere power-duration representation passed to the HMM. In other words, there is a high risk that either the priors will be so diffuse that the proposed labelling is all but useless when the posteriors converge, or alternatively will be too specific to previously observed or assumed appliance modes that they will fail to assist the model in convergence or in labelling after disaggregation. Often, authors of proposed unsupervised algorithms provide no method of labelling the disaggregated traces returned by their models, and instead evaluate their accuracy by manually matching with the correct appliances. Clearly, it is a difficult and open question to determine the optimally general subset of appliance features for labelling in unsupervised learning, especially given the relatively data-poor environment of NILM research. That said, significant effort to produce new real and artificial datasets has been made, of which the following section gives a general outline.

3.4 Datasets

In recent years, more and more energy datasets have emerged, which can vary considerably in terms of complexity, methodology, appliance characteristics and usage patterns, setting and grid characteristics, as well as types of measurements such as real/reactive power, current, voltage, etc. (e.g., see [76, 77]). As mentioned, artificial datasets have also been developed, most with a focus on realistic aggregates through usage patterns. The following summary of existing datasets was compiled by [78], and adapted here with permission. These datasets elucidate a small snapshot of the true distributions over appliance demand properties, which inherently limits the generality of supervised methods and exacerbates the difficulty of the labelling problem in unsupervised methods. Nevertheless, significant progress is being made in the field of NILM despite these limitations.

Table 3.1: Popular NILM Datasets

Dataset	Dur	Sampling Frequency	Houses	Attributes	Loc	M/S
REDD [79]	<6 Mo.	16.5 KHz-1 Hz	6	P,V,I	U.S.	M
BLUED [80]	1 Wk.	12 KHz-1 Hz	1	P,Q,V,I	U.S.	M
HES [81]	1 Yr.-1 Mo.	2 Min.	251	E	U.K.	M
Smart* [82]	3 Mo.	1 Hz	3	P,S,V,I, Amb., Occ.	U.S.	M
iAWE [83]	73 D.	1 Hz	1	P,V,I,ϕ,ω Amb., Utility	India	M
ECO [84]	8 Mo.	1 Hz	6	P,V,I,ϕ, Occ.	U.K.	M
DRED [85]	6 Mo.	1 Hz-1 Min	1	P, Occ., Amb.	NL	M
Dataport [86]	>4 Yr.	1 Hz-1 Min	>1200	P	U.S.	M
UK-DALE [87]	$\leq$ 4 Yr.	16 KHz, 1 Hz	3 + 3	P,S,Q,Utility	U.K.	M
AMPds2 [88]	2 Yr.	1 Min.	1	P,S,Q,V,I, Utility, Weather	CAN	M
SmartSim [89]	1 Wk.	1 Hz	N/A	P	N/A	S
REFIT [11]	2 Yr.	1/8 Hz	20	P	U.K.	M
RAE [10]	1 Yr.	1 Hz	2	P,E,Amb.	CAN	M
SHED [90]	N/A	1/30 Hz-6 Hz	N/A	P,I	N/A	S
ANTgen [91]	N/A	1 Hz	N/A	P	N/A	S
SynD [92]	180 D.	5 Hz	N/A	P	N/A	S

Dur: Duration of measurements (D.: day, Mo.: month, Yr.: year), **Loc**: Geographic location, **M/S**: Measured/Synthetic, I: current, V: voltage, S: apparent power ($I \cdot V$), P: active power ($S\cos\phi$), Q: reactive power ($S\sin\phi$), Amb.: Ambient features (e.g. indoor/outdoor temperature, humidity, etc.), Utilities: meters exist for at least one of (water, gas), Occ.: occupancy information exists.

Chapter 4

Methods

This chapter describes the methods and implementation of the two non-parametric unsupervised NILM solutions alluded to in Chapter 1. The GMM-based method involves modelling the demand of all appliance states with an unbounded number of Gaussian components. Each component then has its duration statistics roughly modelled in a non-parametric GMM using the first 3 deactivation edges whose absolute value belong to the same demand component. Pairing activation edges with their corresponding deactivation edges under energy constraints allows an unlabeled disaggregation of the raw aggregate. These uni-modal appliance states can then be labeled and combined into multi-state appliances by a Res-Net deep learning architecture, which will be described in Section 4.3.

Reliable pairing of activation and deactivation edges is difficult due mostly to transient behaviour: the short-lived demand properties as appliances reach their steady-state operating point. For this reason, a signal denoising filter that extracts steady-state demand is first developed in the following section. This allows better clustering performance of activation edges in the GMM and better pairing between the activation and deactivation edges of a given mixture component. Another motivation for this signal filter is to extract *change-points*: discrete changes in the signal not due to noise. As mentioned in Chapter 2, this provides a significant improvement in inference for factorial HDP-HSMMs by reducing $|\mathcal{D}|$. This filter may prove useful elsewhere, as change-point detection is an important tool in many time-series applications [93].

The accepted pairings for each demand component in the mixture model define a more accurate distribution over state durations. These can then be fed—along with the state demand properties—as priors to a factorial HDP-HSMM, discussed in Section 4.4. Various constraints during inference in this flexible model are discussed, which will be shown in Chapter 5 to greatly improve disaggregation accuracy relative to an unconstrained model.

In order for the mixture model to provide the factorial model with accurate priors, as well as to monitor a given home for new appliances in deployment, it is important to have some indication of novel data. Two measures of a concept known as Bayesian surprise are given in section 4.5 to address this problem.

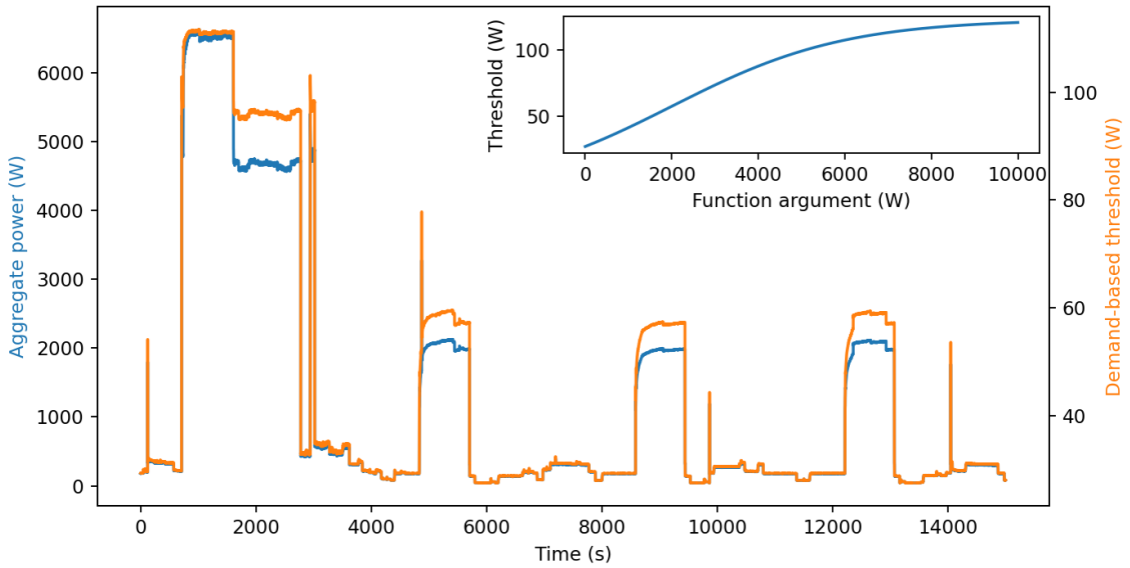

Figure 4.1: Example of demand-based threshold on raw aggregate data. Inset: Sigmoid-based threshold function as a function of demand value.

4.1 Steady-State Block Filter

There were two main considerations in developing the steady-state filter for accurate change-point detection. First, sensor and appliance noise in the system will be larger for high power demand. Second, larger internal transitions should be expected following large initial activation edges of a given appliance state.

To compensate for increased levels of sensor and appliance noise at higher power demand, a hyperbolic tangent-based threshold is first placed on the raw input, which we hereafter call the *demand-based threshold*, τ_d. An example of the demand-based threshold as a function of power demand as well as an example of its value relative to a sample input is shown in Figure 4.1.

The demand-threshold is calculated as

$$\tau_d = m_d \tanh\left(\mathbf{y}/s_d\right) + \tau_{base} \tag{4.1}$$

where τ_{base} is the minimum threshold under which events are ignored, m_d is a multiplier on the tanh function dictating its maximum value ($m_d + \tau_{base}$), $\mathbf{y}$ is the data input to the filter, and s_d is the inflection point of the function, roughly indicating the power scale over which the function reaches its maximum.

An additional threshold, hereafter called the *edge-based threshold*, τ_e, is defined in a similar way, but using its own scale factor and multiplier. This threshold not applied directly to the raw data. Rather, it takes as input the absolute value of the first differences of the

signal. For reasons that will become clear, it only allows those differences more negative than the negative demand-based threshold to contribute, i.e., $\Delta y(t) < -\tau_d$. Differences in the signal less negative than this value are filled backward, preventing these sub-threshold differences from affecting the value of the edge-based threshold. This modification is shown in Figure 4.2. τ_e allows the appliance states themselves to determine the level of thresholding, since larger appliance modes generally contain internal transitions that are much larger in magnitude than smaller appliances. An alternative approach is to fill-forward the first differences larger than the threshold, but Figures 4.3 and 4.4 show the motivations for allowing large *deactivations* to instead determine the final threshold value. In Figure 4.3, we see that most appliance modes have their initial activations masked by transient behaviour. On the other hand, appliance deactivations are transient-free and largely representative of the steady state value. This is evidenced by the relative number of these deactivation edges. Consequently, if the assumption is that large activations are accompanied by larger internal transitions and should therefore be thresholded more aggressively, appliance deactivations are more appropriate to retroactively determine the threshold value. These considerations preclude strict requirements for an *online* filter, i.e., one which returns the filtered signal sample-by-sample as fast or faster than the sampling rate of the data. However, a slight modification could allow the fill-forward approach as discussed above to give an initial filter estimate, which would then be updated following a sufficiently large deactivation. Figure 4.4 shows an activation of the heat pump in the RAE dataset [10], which involves a fan that gradually approaches an operating speed before shutting off. The filling-backward in this case allows the internal transition following main activation to be captured as a single event.

Although not without limitations, this approach is highly flexible and shows good performance in practice. This edge-based threshold determines which events are accepted as valid, given by

$$|\Delta y(t)| > \tau_e$$

These events are referred to as *change-points*, with the samples between these events referred to hereafter as *regions*. Figure 4.5 shows the first differences of the raw data relative to the final edge-based threshold τ_e, noting the inferred regions between change-points marked by red vertical lines.

To achieve some level of robustness of filter behaviour against poor parameter initialization, sub-threshold events are included under strict constraints. Intuitively, if the intermediate values between two change-points (i.e., a region) contain well-defined power levels, it seems reasonable to assume those to be events. To quantify the notion of "well-defined" power levels, a histogram is constructed on each region. The scipy `find_peaks` function is adapted to determine peaks in the histogram, defined by height and separation requirements. Peak height requirements are set relative to the bin with the highest bin count. In

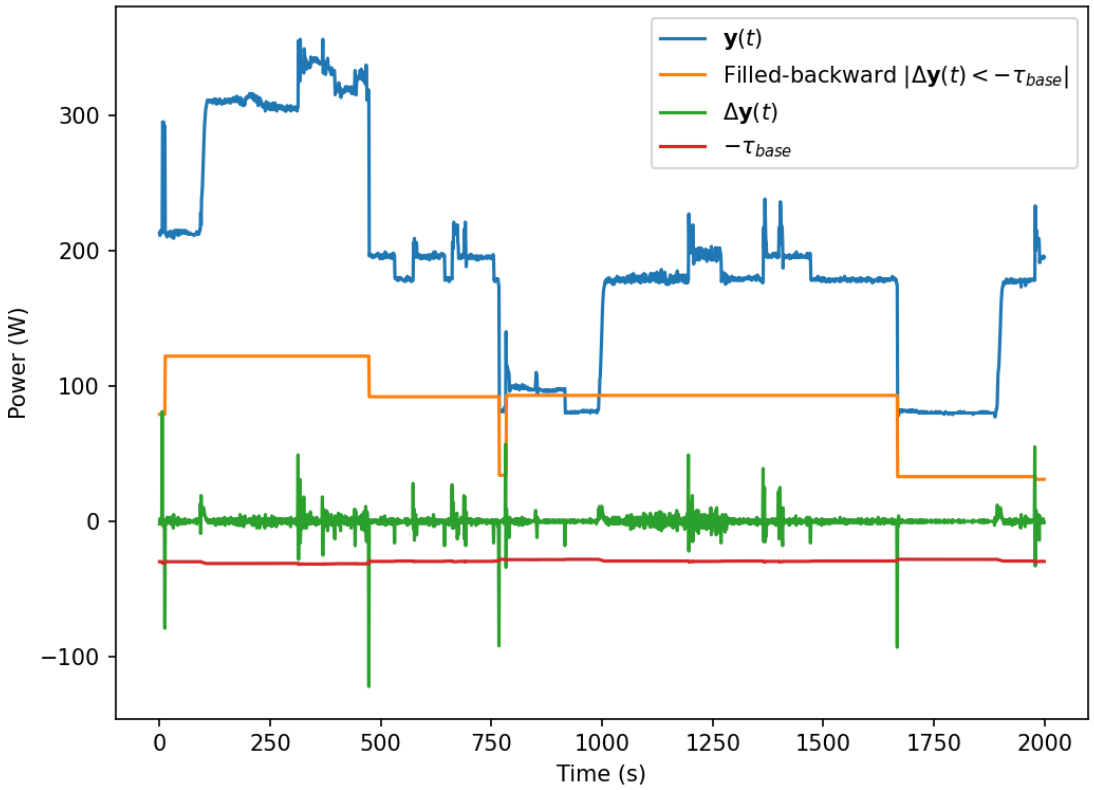

Figure 4.2: Example of filled-backward first differences (orange) for a selected sample of aggregate data (blue) and the corresponding first differences, $\Delta\mathbf{y}(t)$ (green). Differences more negative than $-\tau_{base}$ (red) are filled-backward.

other words, for a secondary peak within a region to be detected, it must have a number of counts above this fraction of the primary peak bin counts. For a primary peak in bin n containing c_n counts, then a secondary peak in bin n' containing $c_{n'}$ counts is accepted if

$$c_{n'} > f_h \cdot c_n, \tag{4.2}$$

where $0 \leq f_h \leq 1$ is the prespecified fractional peak height requirement.

Minimum separation between peaks is set to a specified fraction of the average threshold value over the region. In other words, the peaks can be closer in power value than the actual threshold (which is what helps the filter to be robust in the case of poor parameterization), but not any closer than this minimum separation. This reduces the likelihood of over-enumeration of peaks by the scipy function, while still allowing sub-threshold events provided the other constraints are met. Again, for a primary peak in bin n corresponding to power value P_n, then a secondary peak in bin n' corresponding to power value $P_{n'}$ can be considered only if

$$|P_{n'} - P_n| < f_p \cdot \bar{\tau}_e, \tag{4.3}$$

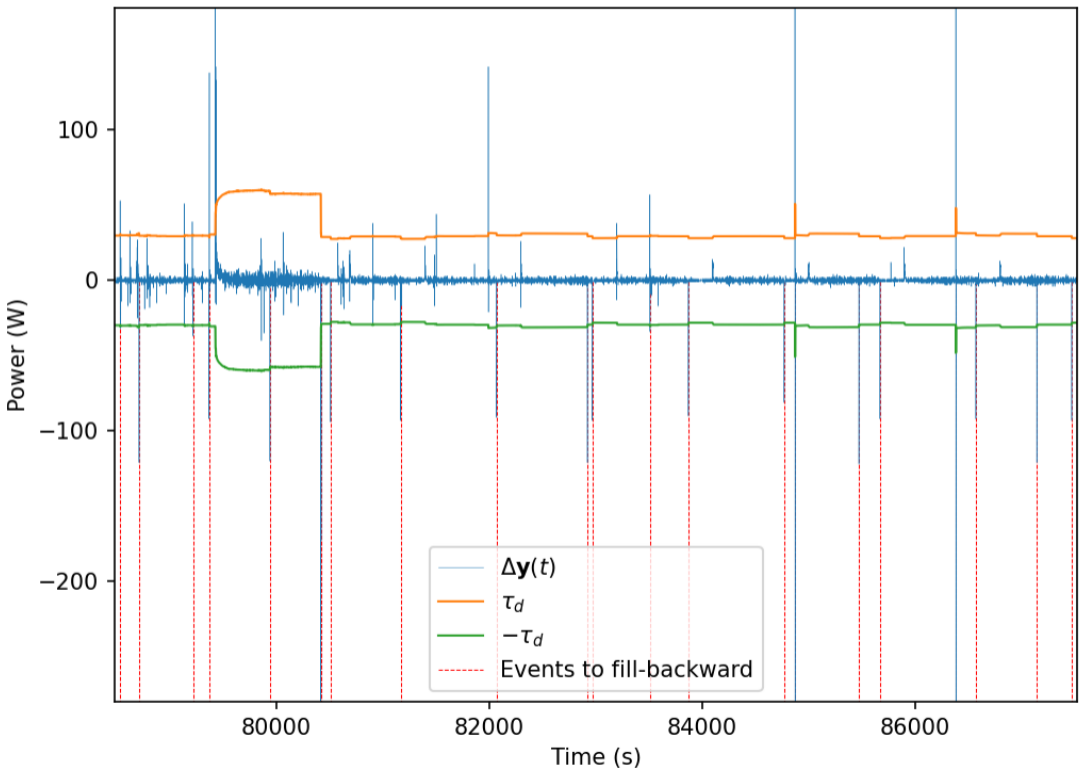

Figure 4.3: Motivation for filling-backward: deactivation edges are transient-free on average.

where f_p is the prespecified fractional threshold power separation, and $\bar{\tau}_e$ is the average edge-based threshold over the region.

If these sub-threshold events are not detected, the mean within the region is imputed and the result can lead to poor edge detection as shown in Figure 4.6.

If multiple peaks are detected in a region, cutoffs are imposed halfway between the power values corresponding to each peak. Samples in the region falling between two cutoff values are replaced by the mean value over the continuous sub-region. Especially for poor parameterizations, multiple discrete power values can occur in different continuous sub-regions. Imputing the mean value over all samples within the cutoff range can lead to significant over or under-estimates of subsequent events, leading to inaccurate state-assignments. This is shown in Figure 4.9. When imputing the mean over a sub-region, we instead compute the mean over continuous chunks of data individually, leading to the more desirable outcome in Figure 4.10.

With the scaling properties and saturating values of the demand-based and edge-based thresholds, along with the additional sub-threshold peak separation and height parameters, finding the optimal parameterization is difficult. In summary, the filter is defined by the following set of parameters: $\{\tau_{base}, m_d, s_d, m_e, s_e, f_p, f_h\}$, where $\tau_d = \tau_d(m_d, s_d)$, and $\tau_e = \tau_e(m_e, s_e)$.

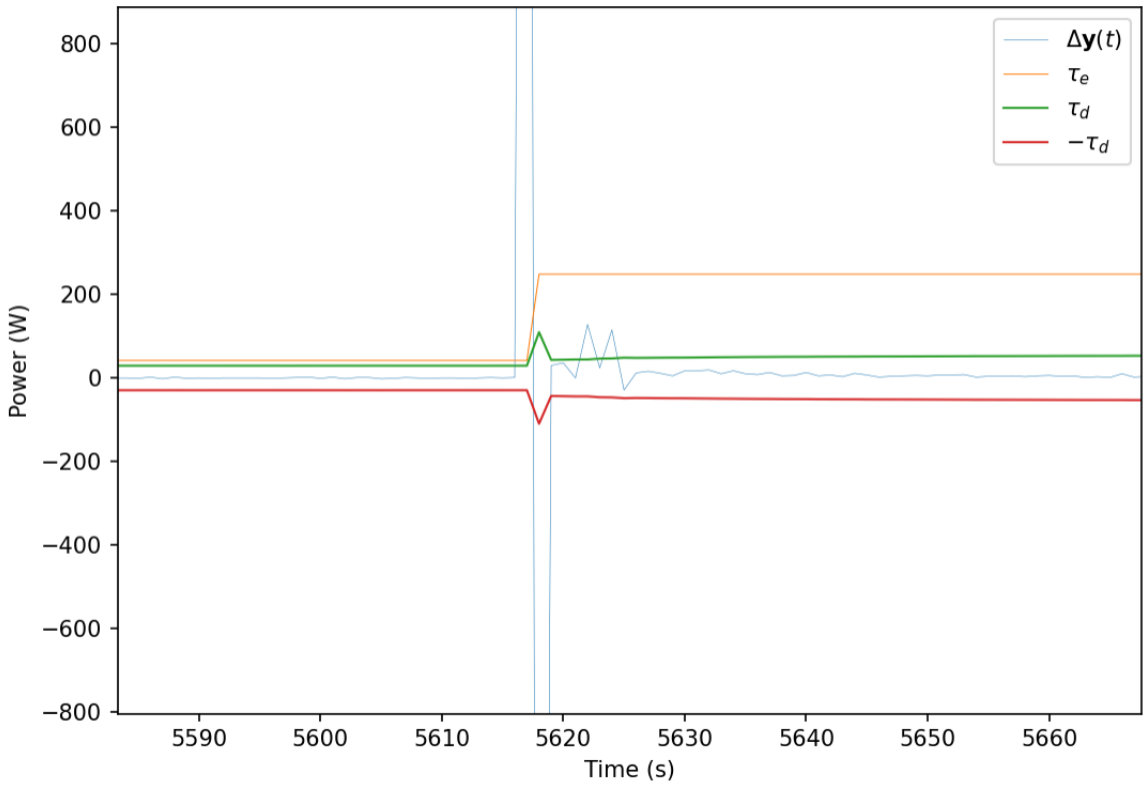

Figure 4.4: Motivation for filling-backward: Internal state transitions for high-demand appliance activations (Heat pump, RAE dataset)

In practice, local "optima" are relatively easy to find using heuristic arguments for thresholding, but these are currently visually inspected and are therefore unlikely to generalize well. Significant effort was invested into explicitly optimizing the filter parameters, but defining an objective function in this case is difficult. The most obvious indicator of successful filtering is disaggregation performance using something dependent on edge clustering, such as the non-parametric GMM method to be described in Section 4.2. However, such metrics are cumbersome when attempting to explore the large state-space associated with the filter parameters. And so a more lightweight objective function is likely required. Intuitively, the filtered signal should trace the aggregate well, but using as few activations as possible. In other words, something to try would be to optimize the RMSE of the filtered signal relative to the raw signal (which penalizes large deviations), but regularize it using edge-sparsity considerations (which maintains as few activations as possible). This objective function would look something like the following:

$$\mathcal{F}(\hat{\mathbf{y}}, \mathbf{y}) = \text{RMSE}(\hat{\mathbf{y}}, \mathbf{y}) - \lambda \text{S}(\hat{\mathbf{y}}), \tag{4.4}$$

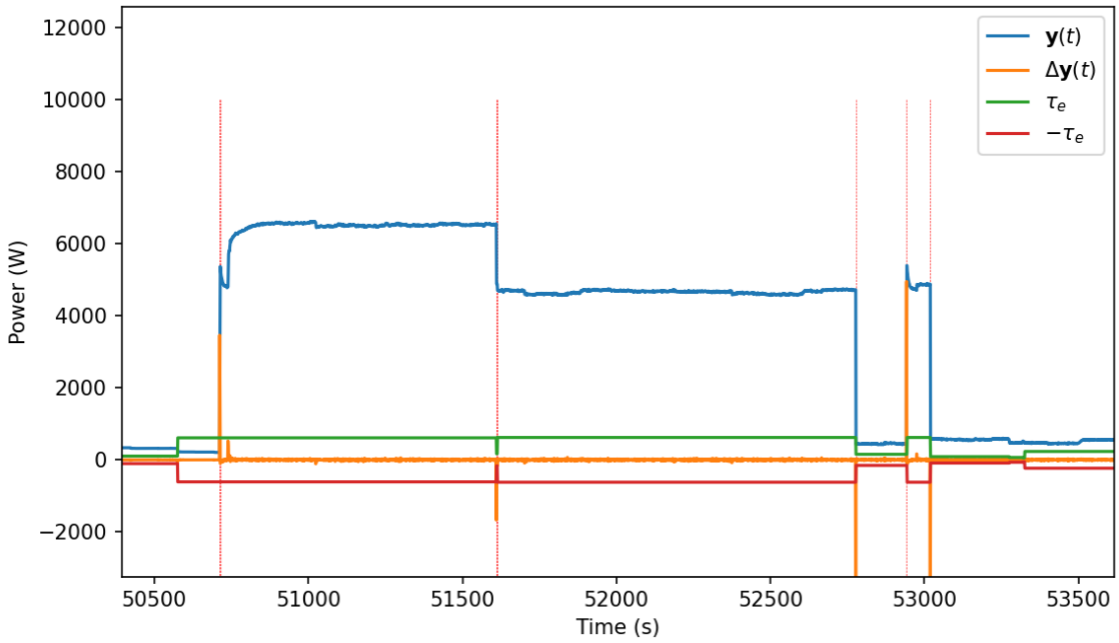

Figure 4.5: Differences in the aggregate (green) exceeding the edge-based threshold in either direction (green/red) are enumerated as change-points, shown by red vertical lines.

where RMSE$(\cdot)$ is the root mean squared error (equation 2.117), λ is the weight term governing the importance of the sparsity loss, S, called the Hoyer sparsity metric [94], given by:

$$S = \frac{\sqrt{n} - \frac{||\delta||_1}{||\delta||_2}}{\sqrt{n} - 1}. \tag{4.5}$$

Here, δ is the discrete first-differences of the filtered signal, while $||\delta||_1$ and $||\delta||_2$ are its ℓ_1- and ℓ_2-norm, respectively.

The optimization method selected was the *Dual Annealing* algorithm, based on [95] and contained in the `scipy.optimize` library. Dual annealing defines a visiting distribution in the parameter space, with the proposed jumps limited by an artificial temperature that is annealed (i.e., exponentially decays) with time. Similar to MH (Algorithm 2), an acceptance probability is calculated for each proposed jump, which depends on the change in the objective function at the proposed landing point. A brief overview of the results of these considerations will be given in Chapter 5.

Clearly, although this filter is agnostic to continuously changing appliances in the raw signal, the resulting filtered signal will likely provide poor results in these cases. In the increasingly prevalent case of residential solar panels, these negative contributions to the aggregate will hide activations in the raw signal. This is clearly a challenge for any NILM solution, although edge-dependent algorithms such as those explored in this thesis will be especially affected. Designing a *Deep Neural Network* (DNN) to detect and extract

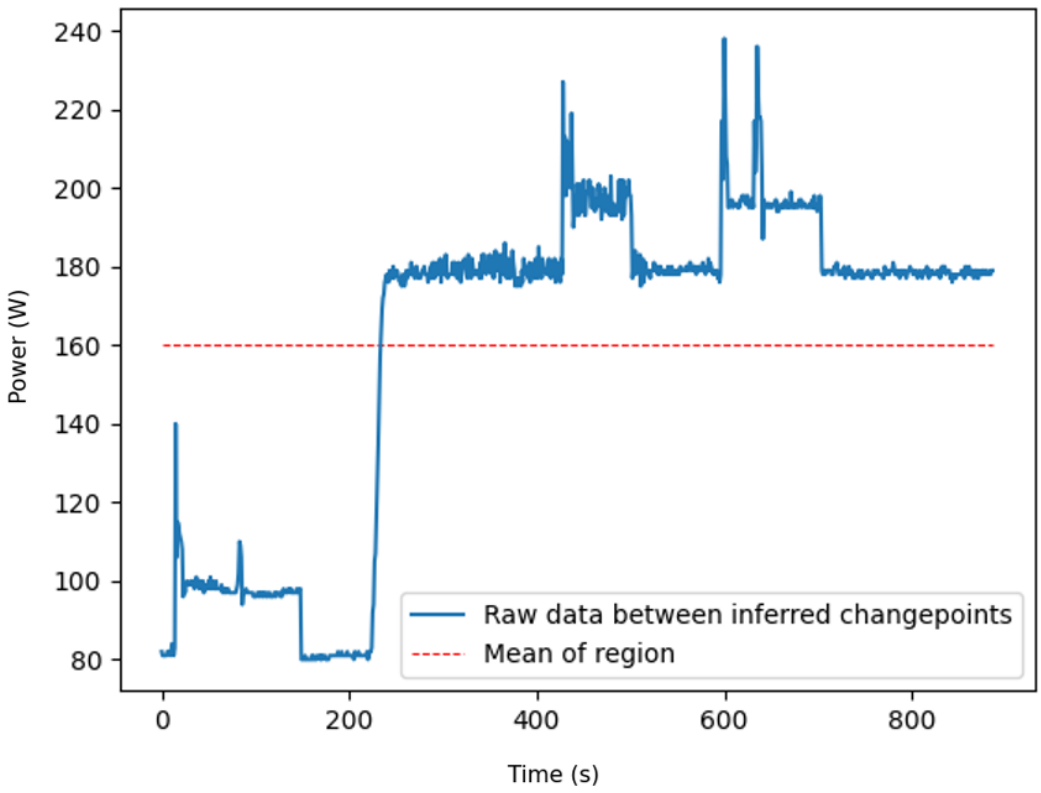

Figure 4.6: Example of poorly parameterized filter behaviour; a single region is clearly comprised of discrete appliance activations.

solar signals from smart meter data would be one way to recover some semblance of filter performance (and NILM algorithm performance) in these cases.

4.2 Non-parametric GMM

With the aggregate signal filtered, appliance states are now far more amenable to Gaussian modelling. Not only are their activations more consistent, the activation edges are much more easily paired with corresponding deactivation edges. Furthermore, the difference in filtered and raw signals provides a straightforward representation of the transient behaviour of appliance states. This section describes the demand modelling of these states using the non-parametric GMM framework, followed by the pairing of activation and deactivation edges to establish state duration distributions. These are used both for inference of edge pairings as well as well as for priors passed to the factorial HDP-HSMM.

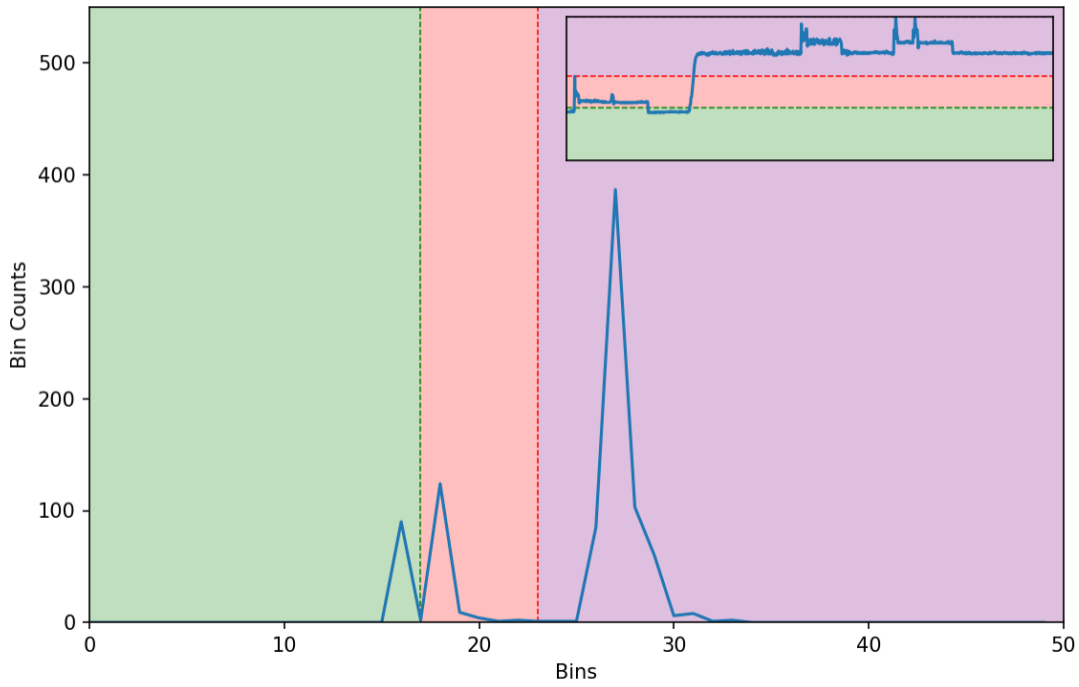

Figure 4.7: Histogram of the same region as Figure 4.6. Peaks are enumerated and their midpoints correspond to power cutoffs for sub-regions shown in the inset.

4.2.1 Demand Modelling

As described in Chapter 2, the non-parametric framework allows a potentially infinite number of unique components. However, the preferential attachment of the DP prior, as well as weak-limit considerations, naturally constrains the number of instantiated components for finitely-sized datasets. Attempting to cluster over the first-differences in the raw signal would lead to highly overlapping components and an unacceptable risk of misassignment. After passing through the steady-state block filter, however, edge values are significantly more consistent across appliance activations.

The filtered signal is fit according to a non-parametric GMM following the variational inferential framework discussed in Chapter 2, specifically equation 2.73. This variational approximation is truncated to a pre-specified maximum of 30 appliance states in this work. The number of states instantiated by the model is dependent on several parameters. The weight concentration parameter, denoted earlier as α, determines the extent to which the atoms drawn from the DP cluster around its associated base distribution ($\mathcal{NIW}(\cdot)$). The form of this joint base distribution is determined by its hyperpriors: the covariance (or scale matrix, Δ) for the inverse-Wishart distribution, and the mean and precision (inverse covariance) for the normal distribution (see equation 2.10). Poor mixture model parameterizations can lead to poor disaggregation performance in the subsequent duration modelling

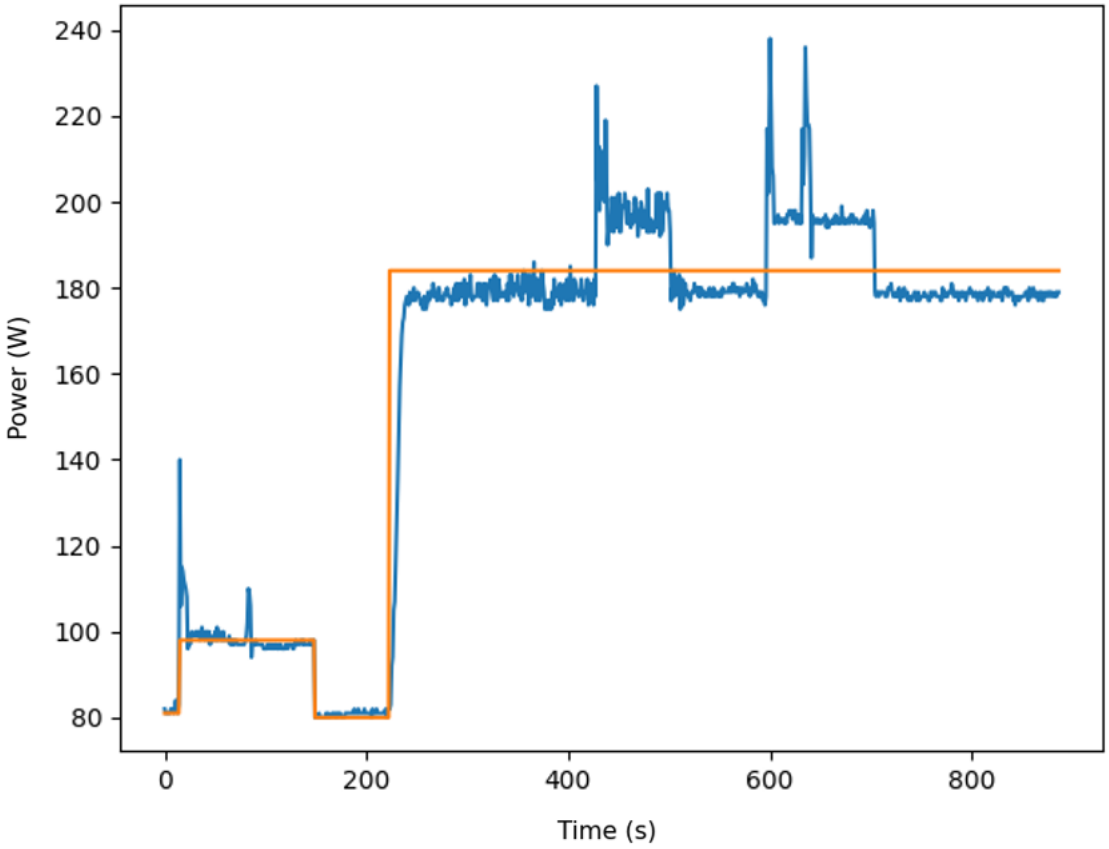

Figure 4.8: Example of same region as Figure 4.6 using histogram peak-finding.

and edge pairing steps. Because of this, an additional round of clustering is performed on clusters for which the associated variance is above some percentage of the cluster mean, set in this work to be 150%. This percentage, although arbitrary, reflects the intuitive fact that larger power appliance modes have larger variances in their block-filtered edges.

For each Gaussian component instantiated by the model (after this sub-clustering), an appliance state class instance is created, specified by the mean, variance, the assigned edges, and an average transient. The transient is simply calculated as the difference between the filtered signal and the raw signal up to the next ON event, and is stored as a running average over component activations. Transient information allows a segregation of appliance modes of similar mean, helping to deal with the state congestion issue inherent in methods such as the factorial HDP-HSMM, mentioned in Chapters 2 and 3. This is done by comparing the transient associated with each activation to its component average in terms of some similarity metric. The selected metric for this purpose was based on the Bray-Curtis distance, defined for the proposed transient τ and running average transient $\bar{\tau}$ by

$$1 - \frac{\sum_i |\tau_i - \bar{\tau}_i|}{\sum_i |\tau_i + \bar{\tau}_i|}. \tag{4.6}$$

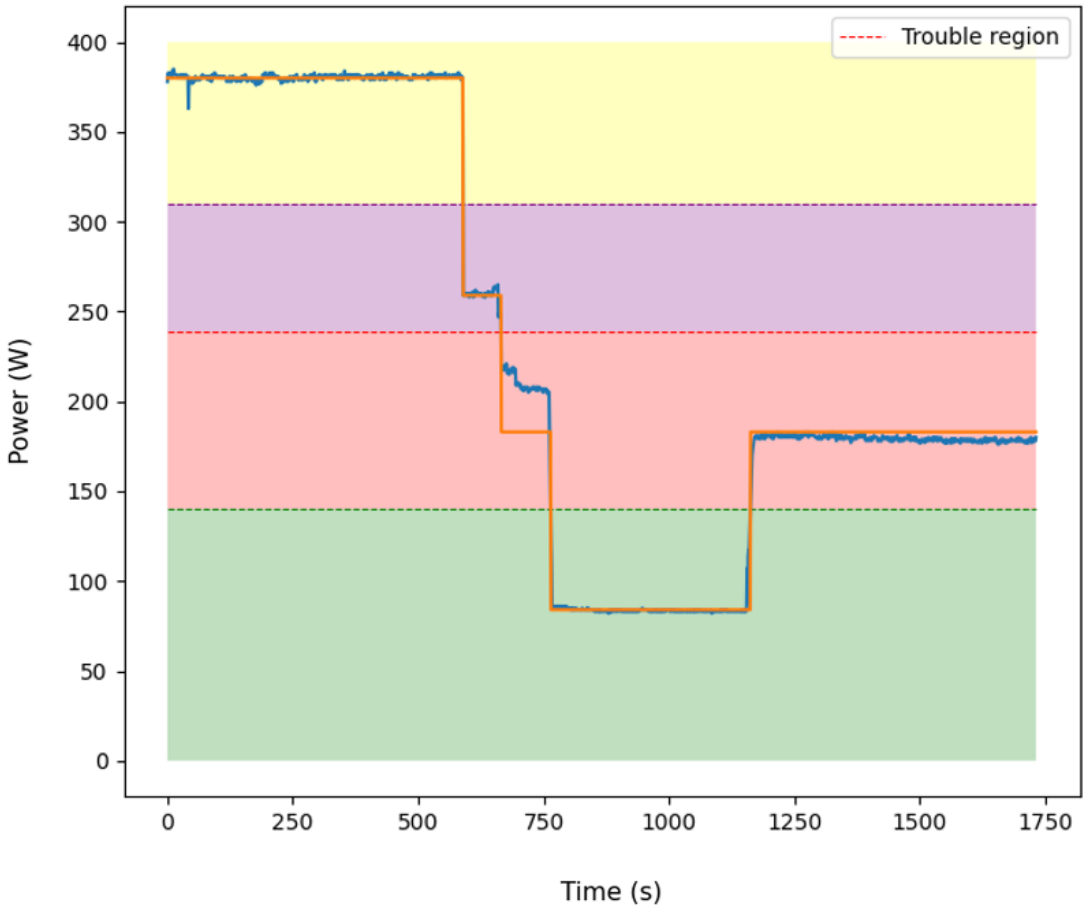

Figure 4.9: Example of poor filter behaviour based on imputing the mean over all samples in a sub-region.

In the limit that both transients are identical, the similarity between them is unity. In the limit that the difference between corresponding points in each transient approaches infinity, the similarity approaches zero. The similarity threshold can be adjusted, allowing for more or less restrictive state assignments. Of those activations rejected from their assigned component, they are compared with other existing state transients, provided their filtered activation value falls within one standard deviation of the proposal state's mean. If none of the existing states' transients provide an acceptable similarity, a new state is instantiated. If the activation in question is removed from its original assigned state, the parameters and transients of the original and accepted state are updated accordingly.

4.2.2 State Duration Modelling

To establish reliable priors for a factorial semi-Markov model, each appliance state must also have its duration statistics modelled. In supervised learning, each submeter can have its operational modes isolated and have the activations and corresponding deactivations

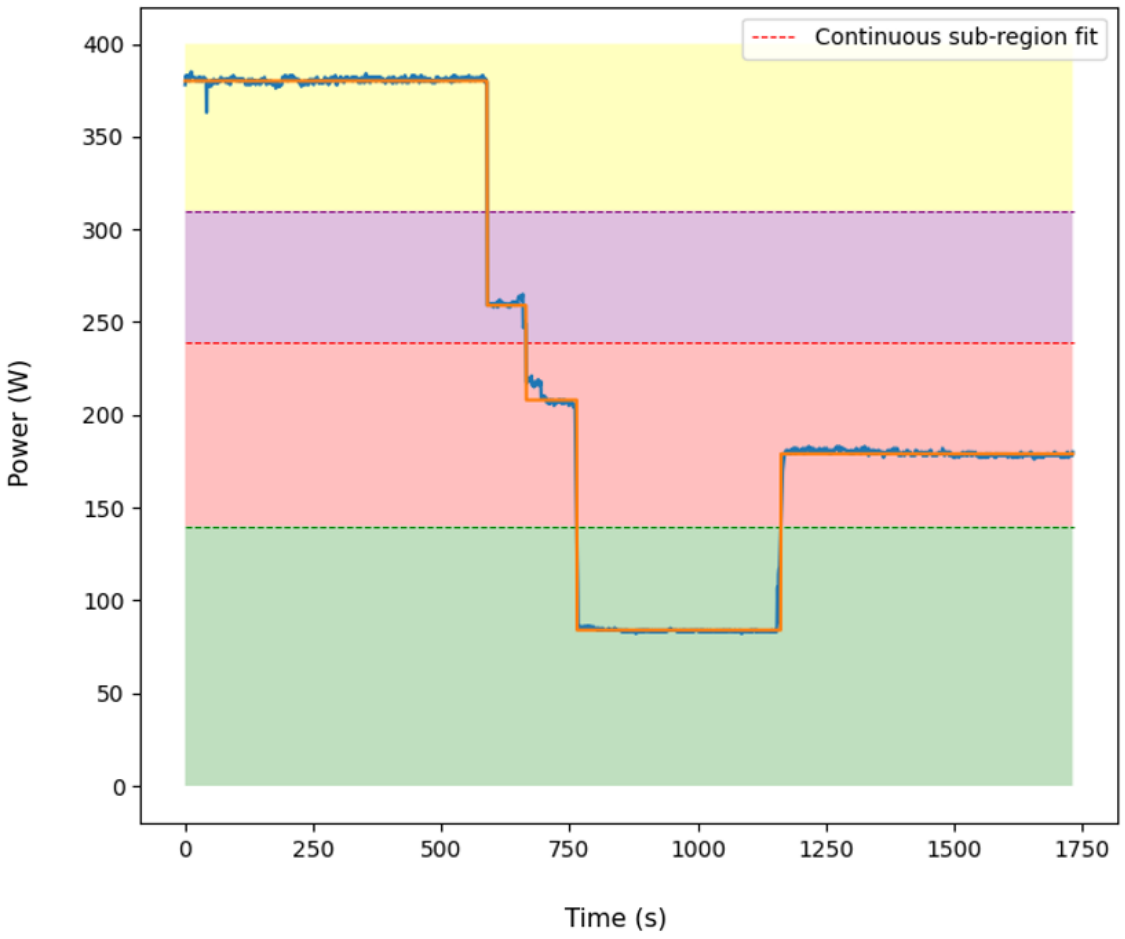

Figure 4.10: Example of continuous sub-region fitting.

of each mode paired with relative ease. In an unsupervised setting, reliably matching activation edges with deactivation edges is equivalent to the task of disaggregation, and is of course more challenging. Based on the power distributions learned in the previous steps for each state, candidate deactivation edges can be selected for each activation edge based on their distance in standard deviations from the state mean. In this work, the first three deactivations following an activation of a given appliance state, and which fall within one standard deviation of the state mean, are stored as duration candidates. The choice of three activations is arbitrary, but it seems reasonable to expect the true deactivation edge to be within that range. This process can be termed *first-n pairing*, where $n = 3$ here. These duration candidates are used to establish an additional non-parametric mixture model, where, given enough samples, the mixture component(s) corresponding to the true state duration(s) are weighed more heavily. An example of this process is shown in Figure 4.11.

Of course, the less well-defined the true state durations are (e.g., user-operated appliances such as lighting or oven/stove-top), the more activations that are required to outweigh

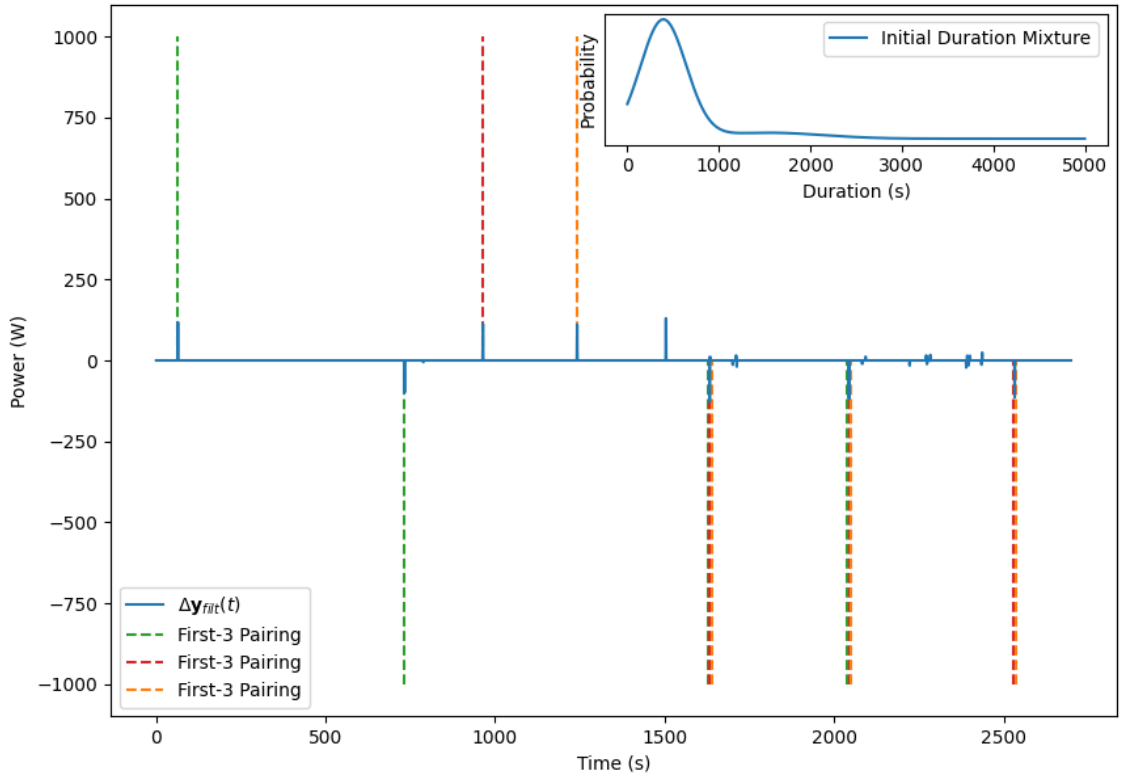

Figure 4.11: Example of first-3 pairing. Inset: corresponding duration mixture distribution

the random components associated with first-n pairing. Nevertheless, this first pass at determining the state duration statistics provides useful additional information for selecting a single proposed edge-pairing.

4.2.3 Edge-pairing

To perform the final edge-pairing, previously un-paired potential activation edges corresponding to each deactivation edge in the filtered signal are ordered in terms of

$$Pr(\delta_s) = \int_{|\Delta y(t)|-\epsilon_P}^{|\Delta y(t)|+\epsilon_P} f^{(s)}(x)dx \cdot \int_{\Delta t-\epsilon_t}^{\Delta t+\epsilon_t} \sum_{k=1}^{K} \pi_k^{(s)} f_k^{(s)}(t)dt. \tag{4.7}$$

Here, δ_s is an activation edge δ belonging to state s, $|\Delta y(t)|$ is the absolute value of a given deactivation edge at time t, ϵ_P is the uncertainty in the observed power value, and $f^{(s)}(x)$ is the demand density of state s, which is Gaussian in this case. In the second term, Δt is the proposed duration, ϵ_t reflects the uncertainty in activation time due to filtering, $f_k^{(s)}(t)$ is the density of the k^{th} component of the duration mixture model associated with state s, and $\pi_k^{(s)}$ is the corresponding weight of that component. In other words, the appliance state's

distribution over demand is integrated over a margin of error surrounding the observed deactivation edge value. Similarly, all mixture components of the state's distribution over duration are integrated over a margin of error surrounding the proposed duration value. The product of these two values provides an estimate for the probability of the combined demand-duration observation. Potential activation edges are considered up to a maximum duration, set in this work to 10,000 1 Hz samples (or around 2.75 hours). This truncation is identical to that mentioned for semi-Markov models in Chapter 2. Truncation helps reduce the number of potential matches, improving computational cost.

Naively, the paired edge for a given deactivation edge might be taken to be

$$\delta_{pair} = \underset{s}{\operatorname{argmax}} \left[Pr(\delta_s) \right]. \tag{4.8}$$

However, the first-n pairing for the initial duration distributions likely leads to long-duration components that may—due by chance to a larger first factor in equation 4.7—dominate the argmax. One way to address this issue is to subject the argmax to a constraint on the total energy in the region defined by the proposed activation-deactivation edge pair. One such constraint that could be imposed on the proposed pairings can be expressed as:

$$\int_{\Delta t} \left(y(t) - \bar{y}(t) \right) \, dt \gtrsim 0, \tag{4.9}$$

where $\bar{y}(t)$ is the average value of the proposed activation edge and the absolute value of the deactivation edge in question, imputed over the proposed duration, Δ_t. In other words, energy must be conserved. The approximate inequality is used since energy in the filtered signal is not strictly conserved; a negative residual is permissible to some degree. Figure 4.12 shows an intuitive example of this case for two potential activation edge pairs, the latter of which might naively be selected by equation 4.8, but would of course be impossible given energy conservation. Although useful in the case of large-demand appliance modes that dominate the aggregate value, this constraint by itself provides little help in situations where the minimum aggregate value over the duration of the proposed pairing is significantly larger than the state's power demand. A potential solution is to sort the activations and pair the largest first, keeping track of the proposed aggregate signal, and instead insisting that

$$C_E \stackrel{\text{def}}{=} \int_{\Delta t} \left(e(t) - \bar{y}(t) \right) \, dt \gtrsim 0, \tag{4.10}$$

where $e(t) = y(t) - \hat{y}(t)$ is now the remainder of the filtered aggregate signal when the proposed aggregate thus far, $\hat{y}(t)$, is removed. The final paired edge is selected by computing

$$\delta_{pair} = \underset{s;\ C_E \gtrsim 0}{\operatorname{argmax}} \left[Pr(\delta_s) \right] \tag{4.11}$$

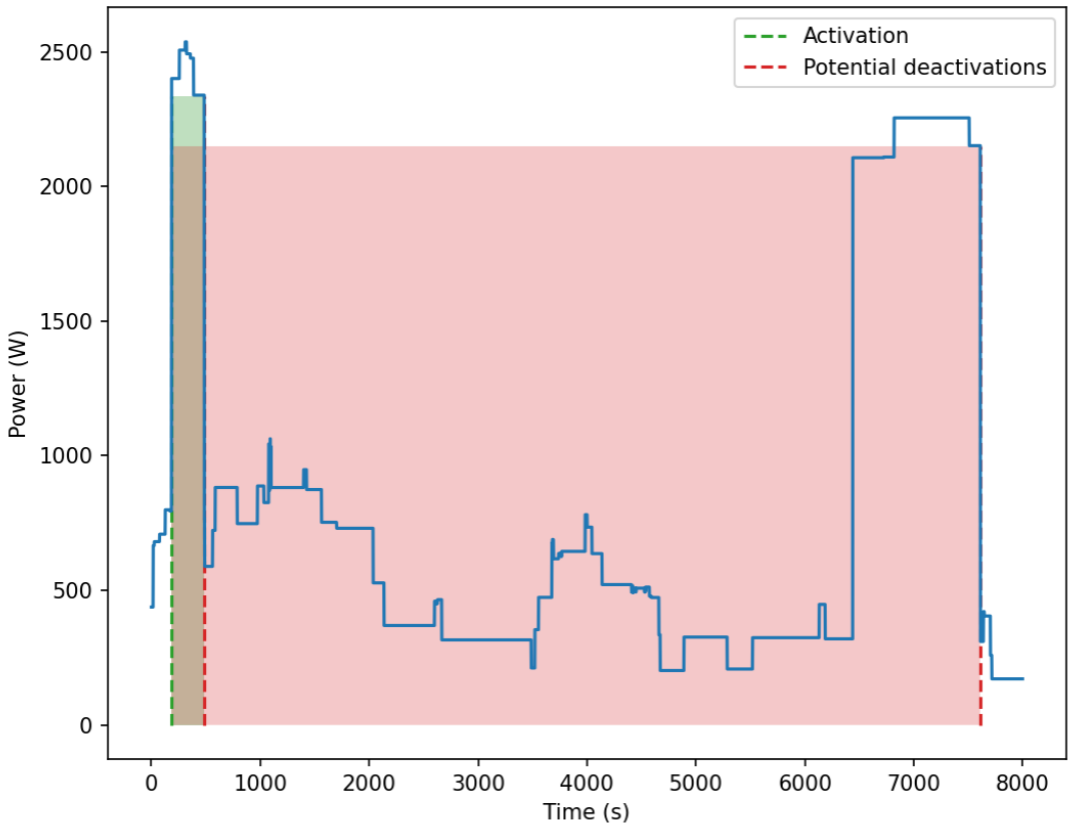

Figure 4.12: Example of obvious energy constraints; the red region contains far more energy than is available in the aggregate.

The accepted pairings for each state under the constraint $C_E \gtrsim 0$ define a much more realistic distribution over possible durations than the first-n pairing initially. Consequently, the non-parametric mixture model corresponding to each state's duration distributions are re-fit according to the accepted pairings. Figure 4.13 shows the significant improvements in both the number of unique components and the variance of the remaining components in a mixture model over constrained duration pairings relative to first-n pairing.

4.3 State Combinations and Labelling

To review the overall workflow up to this point, the raw aggregated signal is passed through a steady-state block filter, and broken down into unique uni-modal power states that contribute to the overall aggregate. In Chapter 2, it was suggested that model-specific differences between appliance types in a home should be leveraged for disaggregation, rather than relying on generalized, appliance-averaged priors. This motivated the use of a post-disaggregation labelling procedure using well-established deep learning classification meth-

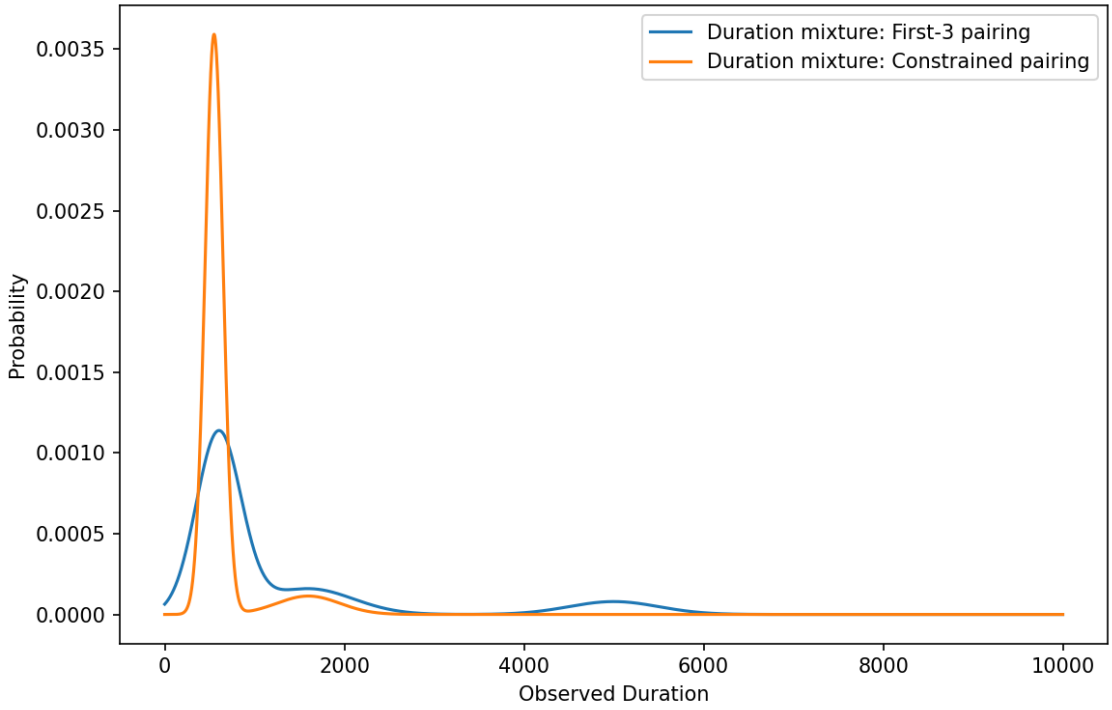

Figure 4.13: Example of re-fit duration mixtures following constrained pairing.

ods. The classification network used here consists of five residual blocks comprised of one-dimensional same convolutions and stride two convolutions, with the latter resulting in downsampling. At the start of each residual block, the inputs were batch normalized according to [96]. The network featured leaky-ReLU activations for all layers except for the softmax activation at the output. Before the final activation, the final downsampled output is passed simultaneously through a max pooling and average pooling operation, followed by a dense layer activation of the concatenated pooling output, dropping out some fraction of these neurons (50% in our case). A summary of the network is shown in Table 4.1. We used Adam optimization with $(lr, \beta_1, \beta_2) = (0.001, 0, 0.99)$, modified to include a decoupled weight decay regularization term as introduced in [97].

For training and validation, we selected several appliance types/groups from the 50 homes (sampled at 1 Hz) in the academic-use subset of the Pecan Street Dataport dataset:

1. Fridges/Fridge-Freezers
2. Washing Machines
3. Tumble Dryers
4. Dish Washers
5. Microwaves

6. Heating Appliances (includes furnaces, hot water units, baseboard heaters, heat-pumps)

7. "Unknown" Small Appliances (includes ovens, televisions/entertainment centers, computers, small electric heaters)[1]

The training and validation sets were broken into approximately five hour windows, centered on particular activations defined as first differences larger than some noise threshold (30 W in this case). Of these windows, only those satisfying two conditions were allowed for training. First, the same Hoyer sparsity metric mentioned in Section 4.1 was calculated on each window. Second, the energy contained in the windows was required to be above some threshold. Removing low energy or overly noisy windows (i.e., a low sparsity) allows only those windows with appreciable structure to modify the network. The training dataset was balanced by forcing a desired number of windows for each appliance from each house. Over-representation was accounted for by randomly dropping the required number of windows, while under-representation was handled by creating randomized repetitions of the available windows. Before each epoch, a random starting point for each window was selected up to half of the window length. From these random starting points, the final windows of 2.5 hour duration were selected. This avoided biasing the network toward particular activation locations within each window, which constitutes a kind of data augmentation.

With the Res-Net classification network trained, the uni-modal appliance states produced as output following Section 4.2, are combined into multi-state appliances (if necessary) with their corresponding labels in what one might call a *greedy-merge* method. In greedy-merging, uni-modal appliance states are first sorted according to their proposed energy. The largest contributor to the energy is passed individually through the network, accepting as its label the maximizing neuron in the soft-max output (provided it exceeds some threshold, say 60% confidence). Once a label is assigned, subsequent states are merged with the initial state if they improve the soft-max output associated with the accepted label. This process is continued until there are either no remaining states to assign, or no remaining states exceed some threshold for unique labels. Figure 4.14 shows this process for the example case of a fridge.

The network was individually trained on both block-filtered data as well as raw data. In the block-filtered case, the returned traces from the mixture model disaggregation could be passed directly to the classifier for assignment. For the raw data case, each state's running average transient was added to each activation, along with 1% Gaussian noise for the remainder of the activation duration.

[1]The unknown appliance group was an exploratory category meant to capture relatively uncertain returned states from the mixture model disaggregation.

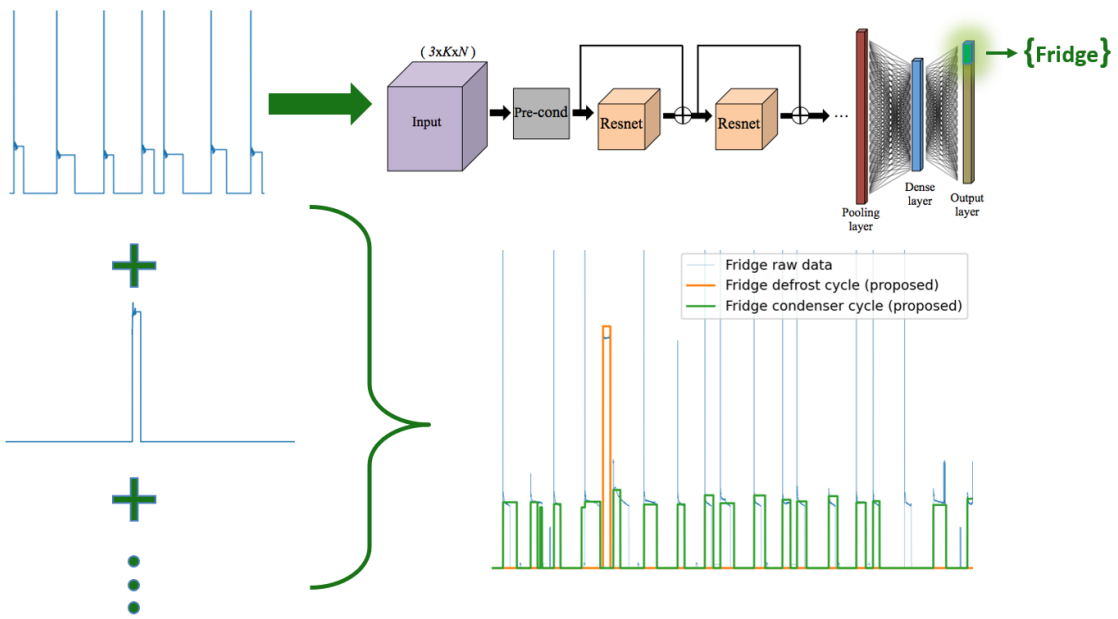

Figure 4.14: Example of greedy-merge method for fridge states.

The proposed method thus far involves disaggregation of raw data and the labelling of disaggregated components. It can therefore be considered a complete NILM solution. However, there are several obvious limitations involved in using this as a self-contained solution. First, appliance energy consumption is estimated based on pairing activations and deactivations in the filtered first-differences. Although segregating the GMM-assigned appliance states based on transient similarity to the state average is useful in many cases, it remains highly reliant on consistent filter behaviour. Poorly initialized filter parameters or aggregate samples outside of the range of stable filter behaviour can result in poor mixture model disaggregation and subsequent labelling. Although the duration modelling and energy considerations provide additional constraints for possible activation-deactivation pairings, state assignment from the initial mixture model and transient behaviour is highly dependent on the accuracy of the filtered signal. There are several examples of poor filter behaviour in Section 4.1. Clearly, filtered edges can be misassigned and the resulting transient (the difference between the filtered and aggregate signals) can appear novel relative to those so far observed, instantiating new states. To address this and similar issues, two potential solutions come to mind:

The first and most naive approach would be to spend significant time perfecting the filter behaviour; checking it against as many different datasets and appliance combinations as possible. In the event that a suitable metric can be developed to evaluate the filter performance, this approach may involve less trial-and-error than it may seem. For instance, the metric discussed in Section 4.1 will be shown in Chapter 5 to show promise for this task. However, there are undoubtedly limitations intrinsic to the filter structure dictating

how well it can perform, even when optimized relative to a perfect metric. An alternative solution is to instead assume that the returned appliance states give a reasonable estimate of the true power and duration distributions associated with the individual appliance modes. Misassignment and poorly filtered segments of data may skew the distributions of existing states, but even these skewed distributions nevertheless serve as highly informative priors.

Since the individual states are already merged and labeled by the Res-Net classifier, a natural step would be to treat each labeled combination of states as factorial Markov chains contributing to the overall aggregate.

4.4 Factorial HDP-HSMM

Using the merged appliance modes, a factorial chain state sequence is initialized based on the proposed appliance's duration and power state assignments in the data seen during the mixture model disaggregation. Priors for the transition and initial state distributions, as well as the component-specific duration truncations, are all learned from these pre-disaggregated traces. To manage model complexity, the off duration statistics for a given factorial component are limited in the experiments that follow to two Poisson mixture components, although this can easily be increased if desired.

As discussed in Chapter 2 and 3, inference in additive factorial models often requires regularization in the form of constraining the conditional posteriors over the state sequence based on the difference signal. In Chapter 3, the concept of constraining the space of possible state durations, $\mathcal{D}$, was introduced based on using change-points in the signal. We can impose further constraints on $\mathcal{D}$ to allow similar one-at-a-time constraints where a difference in the aggregate signal should only be accounted for by a single appliance activation. This constraint is relaxed to allow for multiple coincident activations, but with the probabilities of coincident events scaled down by a factor of 100. This scaling suggests that about 1 in 100 events in the aggregate will involve coincident activations. This is a slight underestimate relative to the investigations in [7] for 1 Hz sampling rate data. However, it stands to reason that the benefits to convergence involved in shrinking the space of possible explanations to the data exceed the risks of ignoring these potential coincident events. Of course, for lower sampling rates, the likelihood of coincident activations increases, and the probabilities of those explanations should be scaled accordingly. In future work, a relatively simple extension of this constraint is to make use of it for a first sampling pass, but relax it entirely for a second pass, allowing multiple activations for differences in the signal not explained by any existing components. Nevertheless, even under these relaxed conditions, there remain certain "guaranteed" constraints (dependent on filter behaviour). For example, a state activation should never be inferred when the difference observed in the aggregate between change-points is less than some multiplier of the state variance smaller than the mean (i.e., $\mu_i - n_{sd}\sigma_i$); likewise for the magnitude of the paired deactivation for that state.

We could allow flexible states in this framework as was done in [45], but with the added advantage of using an unbounded number of them, used as needed in inference. Although this was attempted, the resulting disaggregation was unstable and so it must unfortunately be left to future work. Finally, the same energy constraint described by equation 4.10 can be imposed on the factorial model once the space of potential durations $\mathcal{D}$ is made small enough to be computationally manageable. An additional modification was made in the inference process to remove resampling over state means and their variances. This is done for two reasons. First, significant effort was made to provide accurate power state values and merge them into proposed appliances; resampling over these distributions is re-doing the work done in the mixture model disaggregation. Secondly, it addresses an instability caused when the variance of the states are allowed to grow, and the *variance sequence* (i.e., the sampled values of the emissions around the state mean conditioned on the components' state sequence) can be adjusted with too much flexibility to match the observed aggregate. In practice, this resulted in appliance states being inferred between edges far too small or large (or activation-deactivation edges not nearly matching in magnitude), reducing performance. With these reasonable constraints in place, significant improvements were observed, both in solution stability as well as in the number of iterations required for convergence.

Windows of prespecified length were fed sequentially to the model, chosen for this work to be of 24 hour duration. For each window and a pre-selected number of iterations, the model resamples the state sequence and the associated state durations, the duration distributions conditioned on the state sequence, followed by the auxiliary Gibbs sampling of the transition matrices, and finally the initial state distributions. Given a current window, only the state sequence of previous windows is resampled, taking into account the current windows' updates to the duration, transition, and initial state distributions.

The motivating work for this thesis was that by Matt Johnson and Alan Wilsky, the former of whom also provided an extensive codebase[2] for Bayesian modelling in general, and factorial HDP-HSMMs in particular. Of course, it was adapted to include the above modifications to the initialization and inference procedures, but the general framework is very gratefully borrowed.

4.5 Bayesian Surprise

Once the labelled state chains are passed to the factorial HDP-HSMM, inference of the state sequence of incoming data provides a complete NILM solution. However, appliances in a modern home can change abruptly. The addition, removal, or replacement of appliances in a home can quickly render inflexible models obsolete. To ensure the longevity of NILM solutions in residential homes, some indicator of novel appliance activations is required. A

[2] github.com/mattjj/pyhsmm

natural approach is to leverage the nonparametric mixture model over filtered edges that is already computed in the mixture model disaggregation phase. If this clustering is continued even after the factorial HDP-HSMM has taken over the inference procedure, we can monitor the instantiation of new clusters and adjust the model accordingly. The framework explored for this purpose is that of *Bayesian surprise*, a measure of dissimilarity to assess the effect of data on the belief distributions of an observer [98, 99]. In this case, we take the belief distributions to be the predictive distributions defined by the nonparametric mixture model. The variational approximate predictive distribution was given in equation 2.74. In [100], the surprise computed over these observable quantities is given the term *postdictive surprise*, and we suggest computing this approximate postdictive surprise for a sliding window of w events (preceded by N events) as:

$$S_o = d\big[p(x_{N+1}|x_{1:N}, \alpha, G_0) \;||\; p(x_{N+w+1}|x_{1:(N+w)}, \alpha^*, G_0)\big], \tag{4.12}$$

where d is some divergence metric (usually Kullback-Leibler divergence), and α^* is the posterior update for the concentration parameter if a prior was placed on it (see Section 2.2.2).

Since the basis for the final disaggregation is defined by a (semi-)Markovian generative model, temporal relationships between appliance states are also learned and contribute to inference. We therefore also introduce the notion of *transitional surprise*. In this method, we treat the sequence of events classified by the nonparametric mixture model as a Markov chain, such that the current state of the system is determined only by the state before it. For a system of K appliance states, this transitional surprise constitutes comparing the rows of the $K \times K$ transition matrix. This approximation to the dynamics is clearly crude, but even weak convergence of the transition matrix to some stationary form can prove useful. Over the same window of w events, we compute the transitional surprise over the truncated maximum number of states K as:

$$S_t = \sum_{k=1}^{K} d\big[T_k(z_{1:N})||T_k(z_{1:N+w})\big], \tag{4.13}$$

where at time t, $T_{i,j} = p(z_{t+1} = j|z_t = i)$. The notation $T_k(z_{1:N+w})$ denotes the transition row built using event indicators z for observations $1, 2, ..., N + w$.

The general idea, then, is to define a threshold over postdictive and transitional surprise that is predictive of the stagnation of model performance with increased amounts of (similar) training data. In order to simplify this concept of a surprise threshold under which data is no longer considered surprising, S_o and S_t are normalized according to their maximum values. Since the initial value of the above divergences can certainly be exceeded as observations are made, the maxima were updated and preceding surprise values were renormalized to the revised maxima. Since in an online setting it would be unreasonable to wait indefinitely for

surprising windows, we suggest a patience parameter, ρ. In the experiments that follow, we used $\rho = 100$; that is, 100 windows are observed beyond the most recent window exceeding the surprise threshold. If no other windows exceed the threshold, the previously surprising window is returned as the cutoff point.

In addition to indication of novel events and prediction of how much data is necessary from a given dataset to train a model, Bayesian surprise has other potential benefits. As mentioned in Chapter 3, intra- and inter-dataset transferrability was demonstrated for certain complexity and temporal regularization methods in [70]. Nevertheless, these methods still make use of all available training data. Bayesian surprise metrics provide an attractive alternative/supplement to early stoppage, which by contrast truncate the training set entirely. Finally, further experiments may show that convergence of transitional and postdictive surprise are only weakly indicative of a plateau in model performance. Even so, it may be highly desirable for researchers to merely gauge the effectiveness of new methods or network modifications without spending copious amounts of time retraining using all available data. In these cases, truncating the training set using surprise-based methods allows a significant reduction in research costs, both in terms of computational time spent training and research time spent trying to optimize what may prove to be fruitless methods.

In Chapter 5, we examine useful threshold values and explore cross-house transferability using three houses from the REFIT dataset and five benchmark methods supported by NILMTK [101]: Denoising Autoencoders (DAE) [102], LSTM-based Recurrent Neural Networks (RNN) [103], Windowed Gated Recurrent Unit-based RNNs (WindowGRU) [104], Sequence-to-Sequence autoencoders (Seq2Seq) [103], and Sequence-to-Point convolutional networks (Seq2Point) [105]. The included appliances in these experiments were the dish washer, the washing machine, the refrigerator, the kettle, and the toaster. The Mean-Absolute Error (MAE) was used as a performance metric, defined by equation 2.118. For each house, the available data was split into a training set and test set by a 90%/10% split. 15% of the training set was reserved for validation. The surprise metric was computed on the remaining training data, such that each algorithm was training and validating on the same data. Each algorithm was trained over 15 epochs using Adam optimization with a batch size of 1024 samples. For a given house, each algorithm had its random seed fixed across surprise-based training set reductions, removing initialization variability from their appliance-averaged performance. Preprocessing of the data such as normalization was handled internally by NILMTK.

Finally we illustrate the usefulness of including the concept of transitional surprise using a popular super-state Hidden Markov Model [13]. Clearly, a Markovian model should suffice to show whether our Markovian notion of transitional surprise is useful. We used house 1 from the Rainforest Automation Energy (RAE) dataset [10], which consists of two blocks: a 9 day block beginning on February 7, 2016, and a 63 day block beginning March 6, 2016. Block 1 was used as the test set, and block 2 (and its surprise-based subset) was used for

training the models. The seven appliances used for training were the clothes washer and dryer, refrigerator, dish washer, furnace/hot water unit, and the heat pump.

As an aside, an alternative to this approach is to leverage the unbounded flexible states in the factorial HDP-HSMM, and continuously check these components for consistency in emission/duration properties, as well as the confidence of classification via the labelling network. As mentioned, difficulties in implementation of these flexible states precludes this option for the time being, although future work should explore this option. That said, this option is specific to this particular model, and removes the potential benefits to model development and transferability.

Layer	**Activation**	**Output Shape**
Signal	-	1×2176
CP1D	-	1×2184
DS-Conv-9	LReLU	64×1088
CP1D	-	64×1096
Conv-9	LReLU	64×1088
CP1D	-	64×1096
DS-Conv-9	LReLU	64×544
CP1D	-	64×552
Conv-9	LReLU	64×544
CP1D	-	64×550
DS-Conv-7	LReLU	64×272
CP1D	-	64×278
Conv-7	LReLU	64×272
CP1D	-	64×278
DS-Conv-7	LReLU	64×136
CP1D	-	64×142
Conv-7	LReLU	64×136
CP1D	-	64×142
DS-Conv-7	LReLU	128×68
CP1D	-	128×74
Conv-7	LReLU	128×68
CP1D	-	128×74
DS-Conv-5	LReLU	128×34
Dropout	-	1×256
Linear	-	1×7

CP1D: One-dimensional Constant Padding.
Conv-n: One-dimensional same convolution with kernel size n.
DS-Conv-n: One-dimensional Down-Sampling convolution with kernel size n and stride length 2.
LReLU: Leaky Rectified Linear Unit (leaky negative slope of 0.2).

Table 4.1: Res-Net Classifier Summary

Chapter 5

Results

This chapter provides an overview of each component discussed in Chapter 4: the steady-state block filter, the mixture model disaggregation, the Res-Net-based classification, the factorial HDP-HSMM disaggregation, and finally Bayesian surprise. The steady-state block filter is examined qualitatively relative to some complex aggregate cases, and the optimization method discussed in Section 4.1 is examined in a similar way. Quantitatively, the filter is compared in terms of run-time with an existing method for denoising, as well as in terms of disaggregation performance using a knapsack optimization technique.

The mixture model and factorial HDP-HSMM disaggregation techniques are evaluated based on the manual labelling of disaggregated traces over three sample homes. Several performance metrics are explored.

The Res-Net classification is evaluated based on energy breakdown of an artificial aggregate (meaning comprised only of known sources) relative to ground truth. Furthermore, the networks fraction of correct window assignments and confidence in correct and incorrect decisions are explored for three homes. To uncover some reasons for relatively poor performance, the same evaluation was conducted on ground truth sub-meters in an unseen home from the same dataset as used for training, as well as an unseen, inter-dataset home.

The concept of Bayesian surprise in NILM is evaluated by comparing the decay in performance improvement relative to the decay in postdictive and transitional surprise over three REFIT homes. The suggested surprise threshold is explored as a regularization technique by testing on an unseen REFIT home after training on a training set truncated using this threshold. Finally, transitional surprise is explored further by showing that the decay in performance improvement in a Markovian setting follows closely that of the transitional surprise in a RAE home.

5.1 Steady-state Block Filter

The most straight-forward evaluation of this sort of filter is to simply examine the filtered signal relative to the raw signal. Filter output for aggregate examples of varying complexity

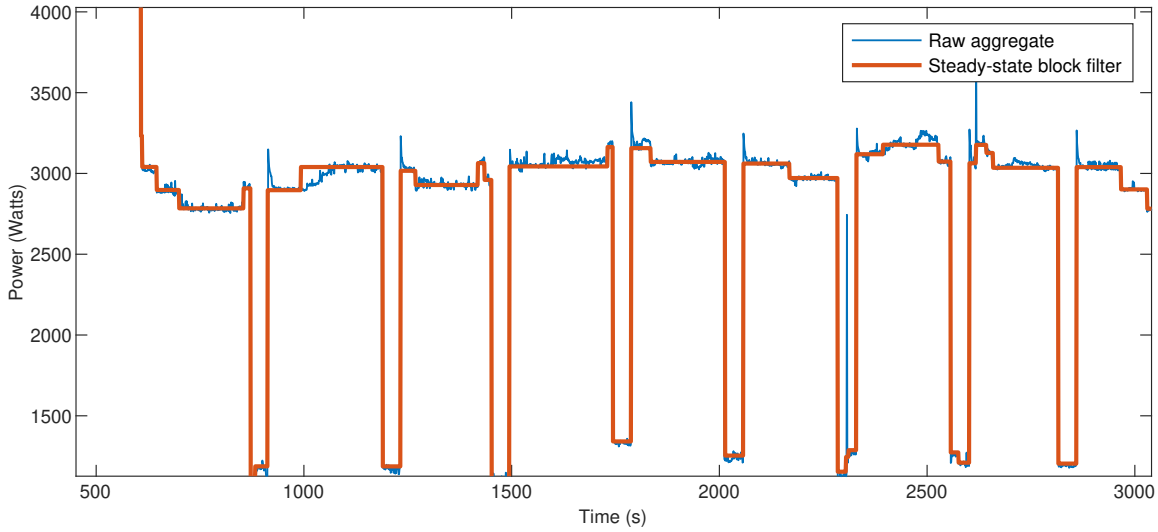

Figure 5.1: Example of filter behaviour.

is provided in Figures 5.1 and 5.2. Figure 5.3 shows examples of edge cases involving staggered and slow-rise activations, as well as the post processing of transients.

Quantitatively, the filter can be compared with an existing method developed in [106], which proposes a pipeline of signal processing filters to achieve the same desired result. Table 5.1 shows the run-time comparison of the current method with that of the filter pipeline. Over 1000 trials, the present method performs at an average of over 6000 times faster (the differences in hardware notwithstanding[1]). In [106], the author makes use of the filtered signal using an unsupervised/semi-supervised knapsack optimization method. The knapsack method was applied to two filtered versions of the same data, with the results compiled in Table 5.2. As shown, the current method more accurately captures the energy of all tracked appliances, and outperforms on disaggregation accuracy scores across all appliances.

As mentioned in Chapter 4, an optimization procedure featuring dual annealing over a cost function defined by the conflicting RMSE loss and Hoyer sparsity loss was implemented. An example of the filter behaviour under various values of the sparsity weight parameter, λ, is shown in Figure 5.4, with a detailed region shown in the inset. As shown, $\lambda = 0.1$ provides promising results, close to what we might expect under the filter parameters chosen heuristically. An annealing schedule on the sparsity loss weight might also be useful. With this, an initially aggressive filtering can be gradually relaxed under minimization of the RMSE. This is left to future work.

[1]Experiments with the steady-state block filter were conducted using a late 2011 Mac Pro with a 2 GHz Intel i7 processor and 4 GB of memory. Experiments with the filter pipeline were conducted using a Mac Pro 2017 with a 2.3 GHz Intel i5 processor and 8 GB memory. Both methods were implemented in Python 3.6.

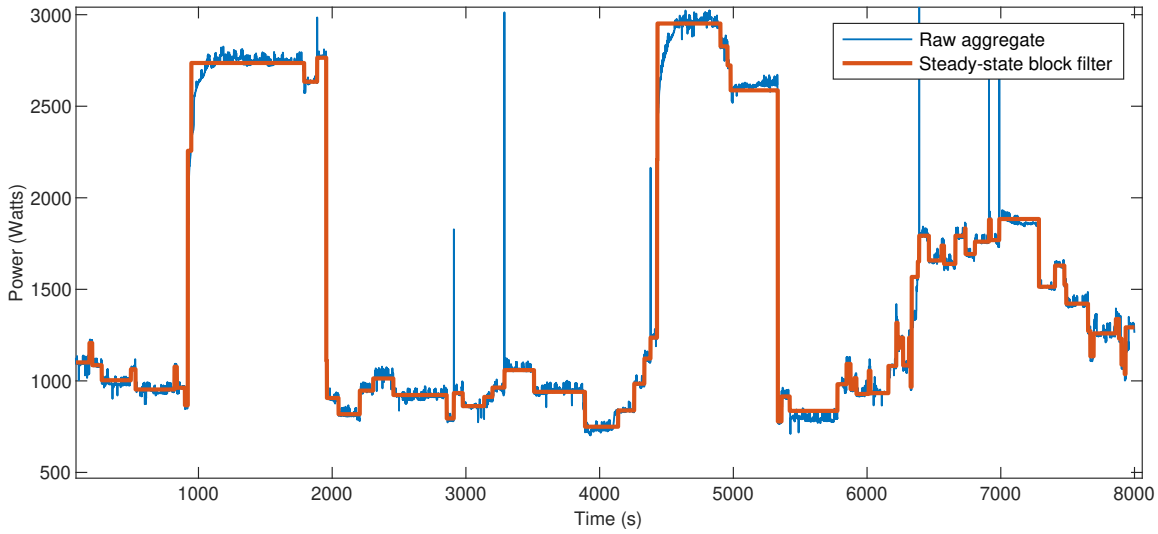

Figure 5.2: Example of filter behaviour.

Table 5.1: Run-time Comparison

Steady-state block filter		**Filter pipeline [106]**	
Process/Step	Time (sec)	Process/Step	Time (sec)
		Median Filter	1.6
		Bilateral Filter	12.7
Block generation	0.1267	Anisotropic Filter	0.1
Post-processing	0.0098	Edge-Preserving Filter	875.4
		Edge Sharpening	0.8
Total	0.1365	Total	890.6

5.2 Mixture Model Disaggregation

Given that the Res-Net labelling tool remains in development and is highly dependent on availability of adequate training data, the results for the disaggregation methods and the labelling network are examined separately. Tables 5.3 to 5.5 show mixture model disaggregation performance for the traces manually matched to the best suited sub-meter. The table columns are defined as follows:

1. Energy Assigned (%): This is the total energy contained in the trace(s) relative to the total energy in the matched appliance
2. F1-score (%) (equation 2.120)
3. Mean Absolute Error (W) (equation 2.118)
4. Root Mean Squared Error (W) (equation 2.117)

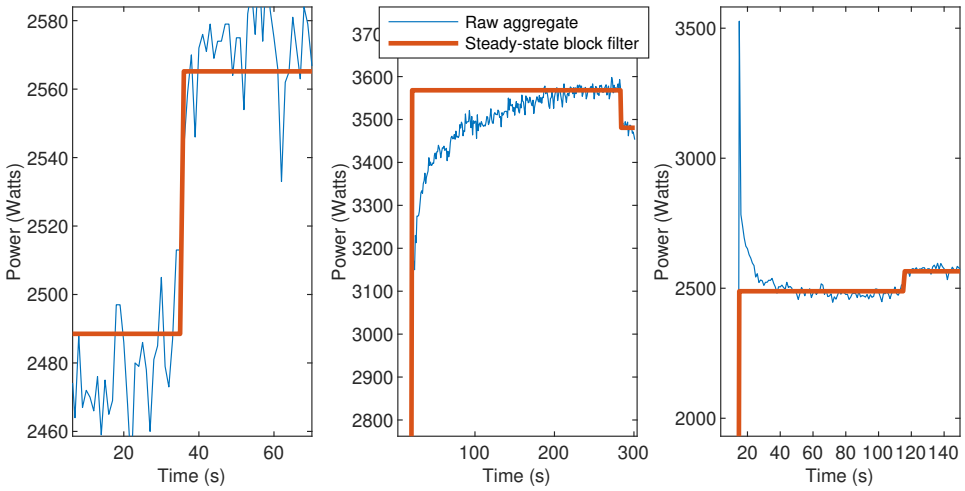

Figure 5.3: Example of edge-cases requiring histogram peak-finding, harsh edge-based thresholding, and post-processing of transients.

5. Ground Truth Fraction of Energy (%): Percentage of the total energy contained in the associated appliance's sub-meter.

As shown, the mixture model is able to capture several important appliance groups with reasonable accuracy. Energy assignment in several cases appears excellent, but for appliances like the fridge in Table 5.3 and boiler in Table 5.4, is paired with a low F1-score. The F1-score is a useful metric for comparison since it ignores true negatives in binary classification. As such, it is likely representative of model performance on unseen activations of the same appliance. By contrast, poor F1-score coupled with perfect energy assignment is more representative of luck than real learning of the correct edge assignments. This can be due to devices which are similar in their demand-duration representation, causing shuffled state assignments and a blurring of the respective transients of each state. This motivates more careful segregation of transient behaviour before state assignment, rather than comparing each activation to the running average of all assigned states. This modification will likely improve these situations.

5.3 Res-Net Labelling

Evaluating the Res-Net architecture for labelling is challenging. Clearly, labelling *all* returned traces by the mixture model or factorial HMM methods would result in a poor evaluation, since the aggregate contains many appliances not explicitly sub-metered. For this purpose, we instead construct artificial aggregates composed only of the sub-metered appliances, and compare the energy breakdown of the disaggregated and labeled traces by both disaggregation methods. Of course, in this scenario disaggregation is a much simpler

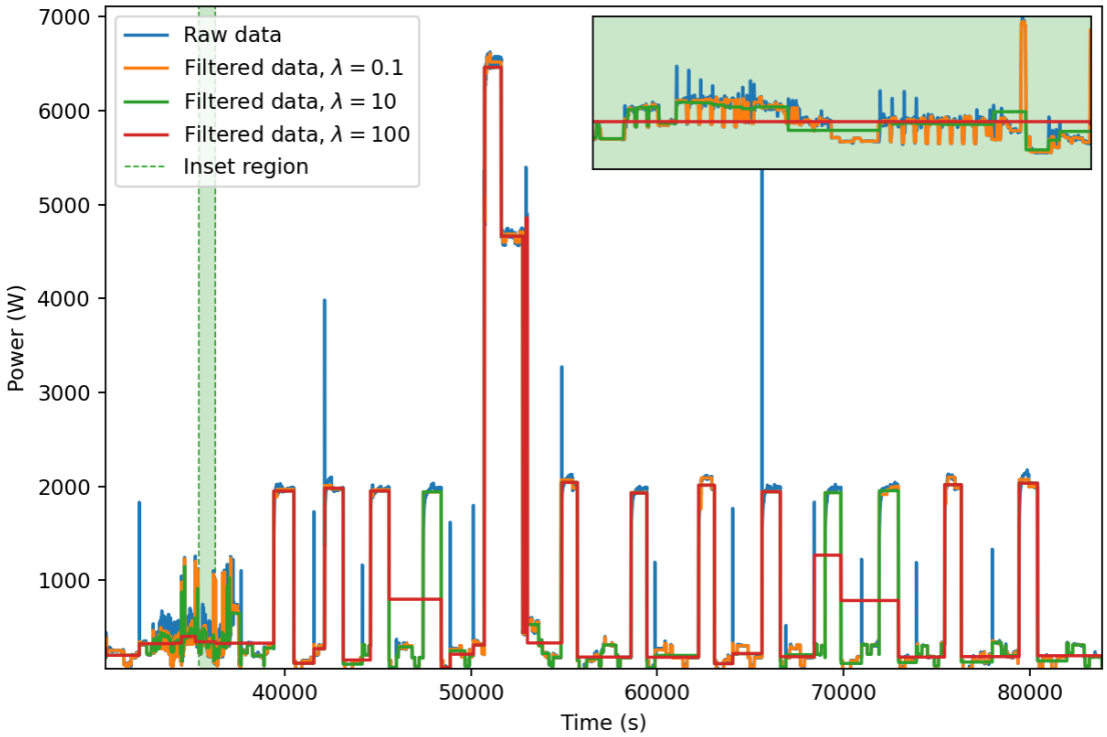

Figure 5.4: Example of various "optimized" filter outputs, for increasing weights on the Hoyer loss. Inset: detailed region for comparison.

task, but this section is meant for the evaluation of the labelling accuracy of disaggregated traces, not disaggregation accuracy itself. Figure 5.5 shows this evaluation for a Dataport home in Austin, TX not included in training or evaluation of the network.

Clearly, both disaggregation methods involve poorly labeled components. The mixture model disaggregation results in one or more heating appliance components labeled as a fridge and significantly overestimates the contribution of the dryer and dishwasher. The factorial model by contrast has nearly all fridge components assigned to heating instead, and likely components associated with heating assigned to the microwave. These sorts of issues are an opportunity to invoke similar signal aggregate constraints as in [46], where information regarding average usage could inform the likelihood of the returned labels.

Another method for evaluating the network is to examine the fraction of windows of the disaggregated components classified correctly and incorrectly, relative to manual labelling. These windows were constructed and filtered according to the discussion in Chapter 4. Of those windows classified correctly, we examine the average confidence (where confidence is defined as the output of the softmax neuron corresponding to the correct class, c: $\sigma(\mathbf{z})_c$). Of those windows classified incorrectly, we examine the average incorrect confidence in a similar way, by the output of the softmax neuron of the chosen (incorrect) class (i.e., $\max_{\mathbf{z}} \sigma(\mathbf{z})_{c'}$). In this way, we get a picture of how certain the network is when it is both correct and

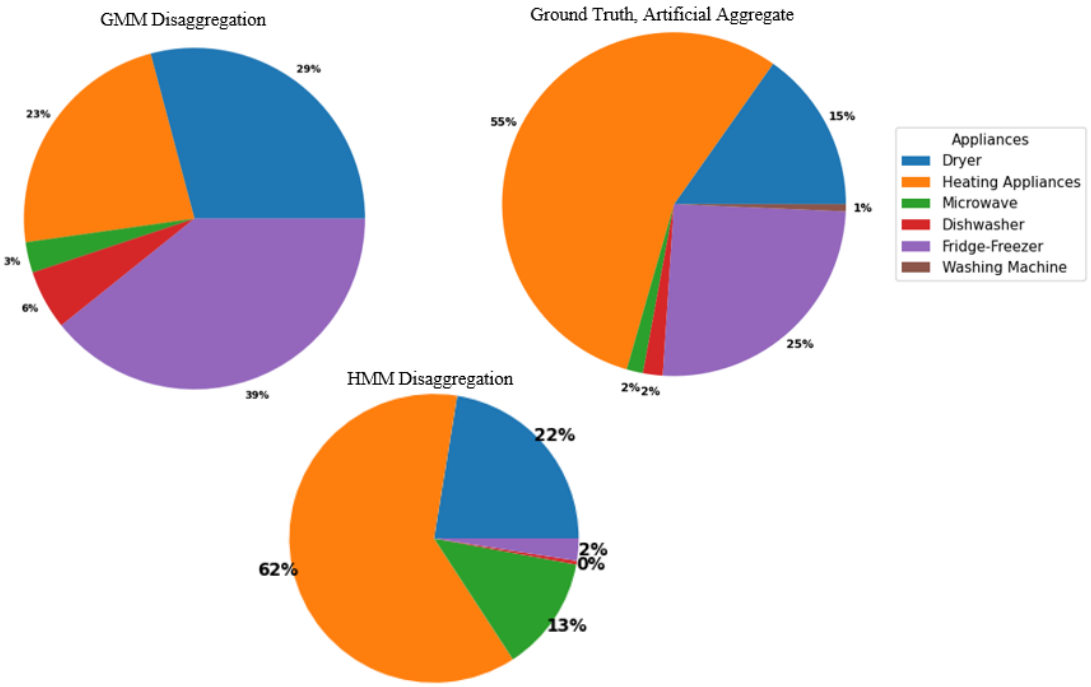

Figure 5.5: Ground Truth (upper right) and auto-labeled GMM disaggregated components (upper left) and Factorial HDP-HSMM disaggregated components (lower) using an artificial aggregate for an Austin, TX home.

incorrect. Ideally, the network should have a low confidence when incorrect, and a high confidence when correct.

In Tables 5.6 to 5.8, these metrics are reported for both the disaggregation output returned by the mixture model method, as well as that returned by the factorial HDP-HSMM method. The factorial method in this case was passed the individual components of the mixture model disaggregator directly, rather than greedy-merging beforehand. This step is to establish whether labelling performs better after the factorial method rather than before.

Clearly, the preliminary results of the labelling method are less than encouraging. Although labelling the Dataport home showed slightly more reasonable performance in terms of correctly assigned windows, even in this situation of same-region testing, the average incorrect confidence exceeds the average correct confidence for two appliances/appliance groups. In general, the correct confidence is far lower on average than I would have hoped, and the incorrect confidence far higher. Despite relatively simple manual labelling to verify the appliances corresponding to the returned traces, the network appears to struggle with this task. This would indicate that the poor breakdown of energies contained in the artificial aggregate in Figure 5.5 is likely not due to chance or due to a particularly difficult signal to disaggregate.

There are several caveats to the relatively disappointing results reported here: First, Tables 5.6 through 5.8 were calculated using the network trained with block-filtered data and the corresponding transient-free disaggregated traces. The results for the network trained using raw data and tested with the disaggregated traces superimposed with the running-average state transients showed similar performance. However, isolating state transients with more restrictive similarity thresholds as mentioned in Section 5.2 would be an important investigation for the performance of the labelling network. Secondly, a batch size of 64 windows was required to be passed to the network for labelling, given the architecture design. Sparsely activated traces returned following disaggregation, along with the subsequent filtering done over window sparsity and contained energy, could result in fewer than 64 windows over the duration of the aggregates used for testing. The batch size was filled by randomly repeating valid windows, which could have inflated both positive and negative results. Finally, the training set was of course biased to two relatively small regions in the United States. Although we might expect North American appliances to have largely shared makes and models, this expectation was not verified. Furthermore, behavioural differences in appliance usage given the relative climates and times of year can bias the network. An important examination along these lines is to check how associated the poor labelling performance is with unrealistic returned appliance traces following disaggregation, rather than with the network biased toward certain appliance models and behavioural characteristics. Table 5.9 shows the same analysis of labelling performance, but instead using the block-filtered ground truth sub-meters from House 1 in the RAE dataset and the unseen Dataport home in Austin, TX. For the RAE home, the network does perform better on average with ground truth traces, although not to the degree that one might expect under the assumption that the disaggregators are solely to blame. For example, with ground truth traces, heating and refrigeration are both poorly labeled, and average incorrect confidence remains excessively high for several appliances. Exploring the same analysis on the unseen Dataport home shows much higher performance for these appliances, suggesting that similarities in appliances associated with the geographical region may be affecting labelling performance significantly. A more representative training set is therefore an important step in improving the usefulness of this labelling method.

5.4 Factorial HDP-HSMM Disaggregator

Given the complex task of inference in these models and the significant effort put into regularizing the conditional duration posteriors, an important outcome to evaluate is the rate of convergence in Gibbs sampling as well as performance in general. Figure 5.6 shows estimation accuracy (equation 2.116) over five appliances[2] in a raw aggregate use-case,

[2]fridge, furnace, dryer, dishwasher, heatpump

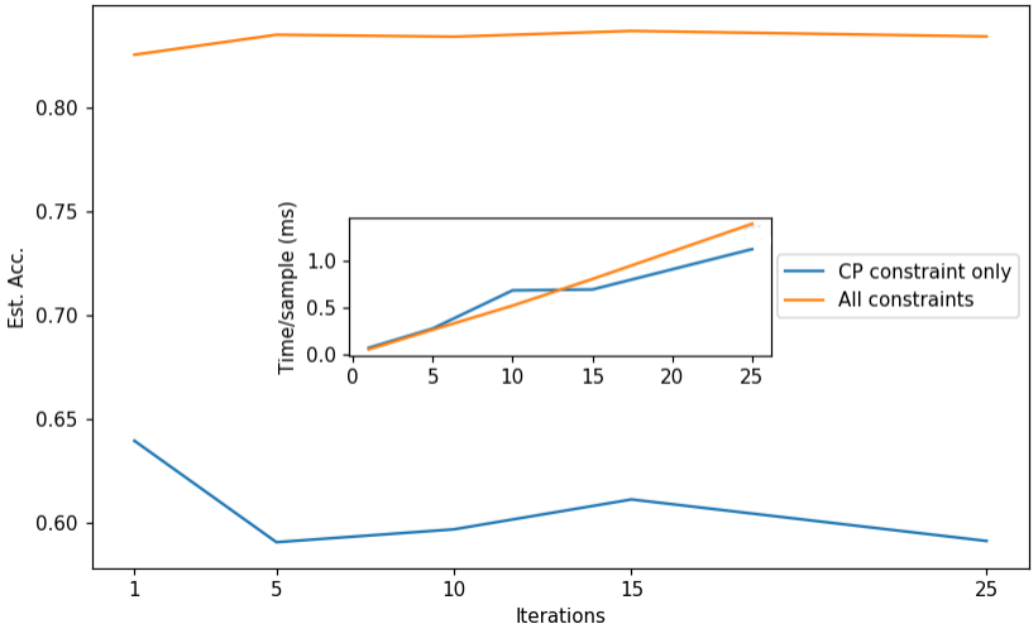

Figure 5.6: Estimation accuracy (equation 2.116) for change-point constraints only, as in [9], and full constraints in inference.

relative to iteration number for two inference schemes. The first is that proposed in the motivating work by [9], which includes only change-point considerations in restricting $|\mathcal{D}|$ (in addition to truncation). The second uses all constraints mentioned in Section 4.4, which include the relaxed one-at-a-time constraints and various energy considerations. As shown, these constraints significantly restrict the space of possible explanations to the observed aggregate, with fully constrained model achieving nearly 20% higher estimation accuracy at worst, and over 25% at best. Furthermore, the additional time complexity involved in computing the constraints is recovered by the reduction in $|\mathcal{D}|$, with the fully constrained model scaling similarly for the number of iterations explored here.

To examine the performance of the factorial HDP-HSMM disaggregation, we use as priors the traces returned by the mixture model disaggregation method and examine the same metrics on the same datasets. In terms of F1-score for RAE House 1, the factorial model outperforms the mixture model on 3 of 5 appliances, and one only marginally worse. The same is true for RAE House 2, where 3 of 5 appliances have higher F1-score. The central vacuum is relatively poorly disaggregated, but given its small ground truth fraction of energy (0.3%), this appliance would likely be ignored. On REFIT House 2, we see comparable performance for the two methods, but with the factorial method showing significant improvement for the kettle. Undoubtedly, appliances poorly disaggregated by the mixture model method will provide inaccurate priors for the factorial model. As argued,

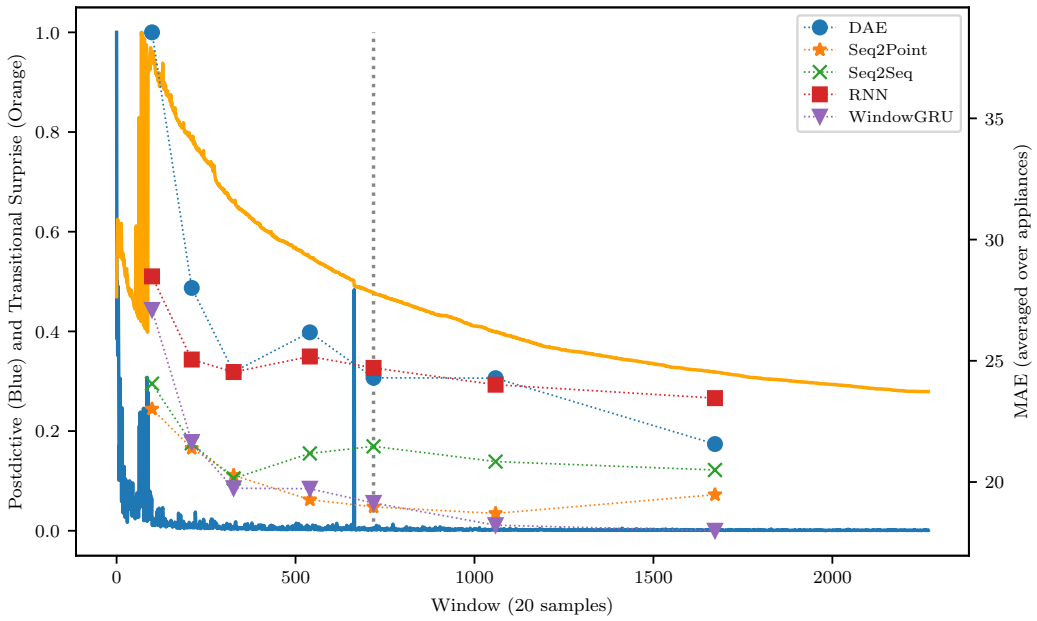

Figure 5.7: Appliance-averaged MAE performance, REFIT House 2

the factorial components are robust to some inaccuracy in these priors given their flexibility during resampling, but the extent to which this is true has not been explored.

5.5 Bayesian Surprise

To establish a relationship between algorithm performance and the proposed surprise metrics, three houses from the REFIT dataset [11] were selected for study using the referenced disaggregation methods in Section 4.5. Figures 5.7, 5.8, and 5.9 show the behaviour of the MAE for the average appliance across the benchmark methods for houses 2, 3, and 5, respectively. The postdictive and transitional surprise was computed using Jensen-Shannon divergence, defined between two distributions p and q by:

$$d_{JS}(p||q) = \frac{d_{KL}(p||m) + d_{KL}(q||m)}{2}, \tag{5.1}$$

where m is the point-wise mean of p and q, and d_{KL} is the Kullback-Leibler divergence, given by

$$d_{KL}(p||q) = \sum_{x \in X} p(x) \cdot \log\left(\frac{p(x)}{q(x)}\right). \tag{5.2}$$

Given the max-value normalization, the postdictive and transitional surprise values can be interpreted as the fraction of the maximum observed surprise, rather than the value of the JS-divergence itself.

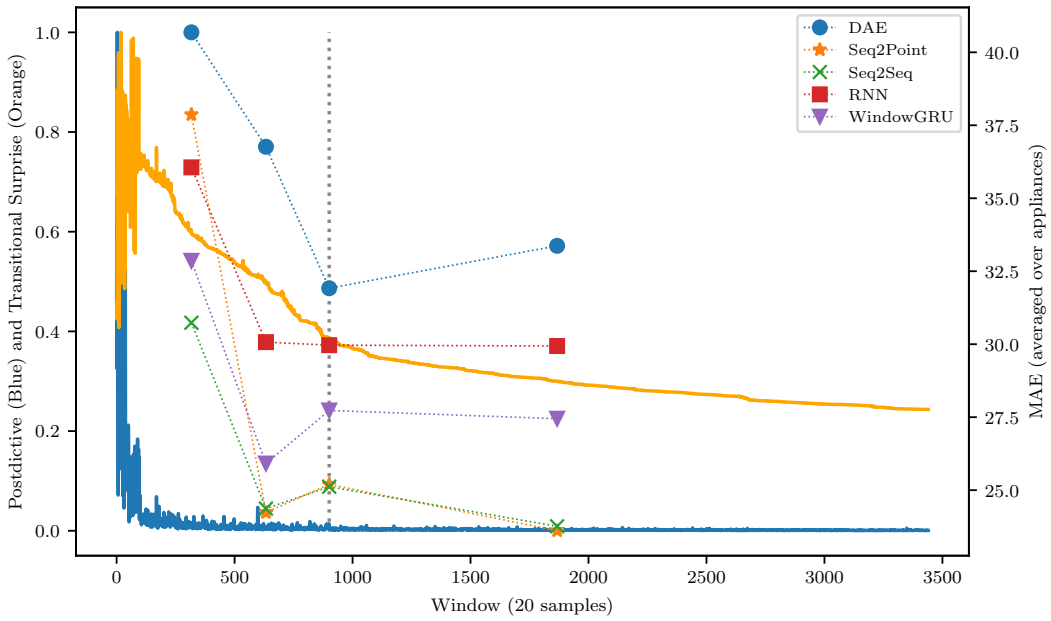

Figure 5.8: Appliance-averaged MAE performance, REFIT House 3

Although of course no sharp transition exists between an optimally and sub-optimally sized training set, the behaviour of these algorithms' MAE in the three REFIT houses suggest that performance can indeed stagnate. Additional similar data, especially in houses 2 and 3, seem unlikely to appreciably improve performance. An example surprise threshold is shown in Figures 5.7, 5.8, and 5.9 as a dotted grey line, indicating an approximate point where performance began to plateau. This cutoff was chosen as a joint threshold over postdictive and transitional surprise, defined by:

$$S_o(w : w + \rho) \leq 0.01 \;\&\; \; S_t(w : w + \rho) \; \leq 0.05, \tag{5.3}$$

where again, w is the window size, ρ is the patience parameter, and S_o and S_t are defined as in equations 4.12 and 4.13, respectively. We used this threshold for further study regarding the potential regularizing effect of surprise-based training cutoff.

To explore the potential regularizing effect mentioned in Chapter 4, we examined the MAE performance of each algorithm when trained on the full REFIT house 3 and the surprise-based subset determined by the joint threshold in equation 5.3. Table 5.13 shows the appliance-averaged MAE performance of each benchmark method when tested on REFIT house 5. All but one method showed improved cross-house transferability with a restricted training set, giving some substance to the claim that truncating the training set may provide regularization against overfitting.

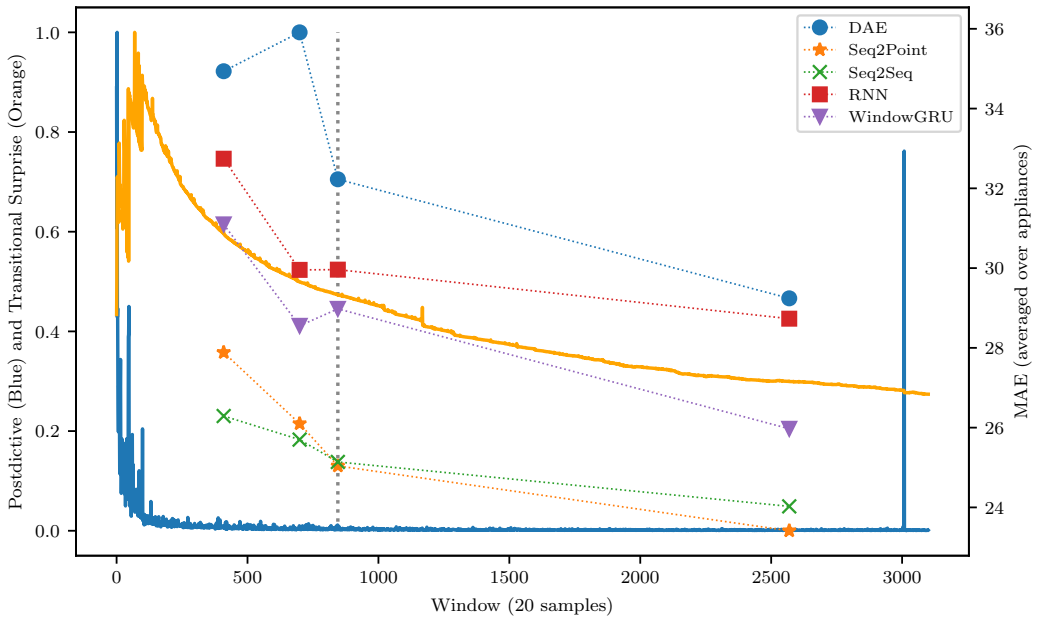

Figure 5.9: Appliance-averaged MAE performance, REFIT House 5

Figure 5.10 shows the Van Rijsbergen's effectiveness measure (defined simply as 1− F1-score) as a function of cutoff point during training. This measure decays slightly faster than that of the transitional surprise, but significantly after the postdictive surprise had converged. This lends credence to the claim that postdictive surprise is an unreliable metric for terminating training in the general case. The difference in decay rate between transitional surprise and the effectiveness measure is understandable given that the SSHMM by definition encodes the Markovian dynamics between super-states of the user's home. The super-state of the home at a given instant in time can be thought of as the complete description of the home, denoting the operational mode of each appliance in the house. Each instant in time increments the underlying transition distributions between super-states of the home, rather than individual appliance states. This will in general encode the state dynamics more efficiently since there is more information used per time-step. Nevertheless, the basic notion of transitional surprise introduced here allows a useful overestimate of the learning rate of the system dynamics. Notably, the behaviour of the effectiveness measure in this case calls into question the specific values given for the joint threshold in equation 5.3. Here, a threshold on the transitional surprise of ≈ 0.4 seems adequate to predict stagnant performance improvements for this dataset. Significant exploration with all available datasets is needed to further narrow down acceptable threshold values. This is left to future work.

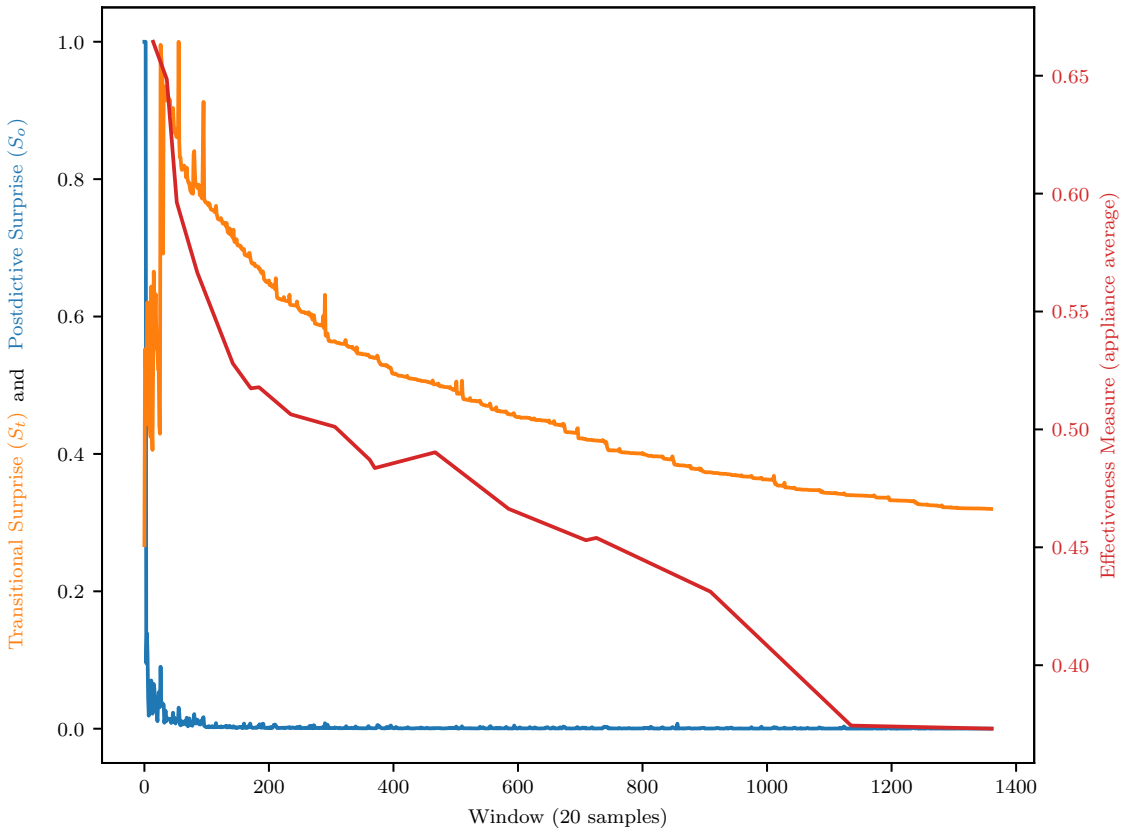

Figure 5.10: Effectiveness measure (1-F1-score) averaged over 7 appliances as a function of surprise-based training cutoff

Table 5.2: Comparison of Energy Truth/Filtered/Tracked (in kWh)

Steady-state block filter				
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%
Filter pipeline [106]				
Appliance	**G.Truth**	**Filtered**	**Est/Tracked**	**Truth *vs* Est**
Clothes Dryer	2.753	2.729	2.604	94.5%
Fridge	0.063	0.065	0.055	87.3%
Furnace	0.174	0.167	0.144	82.8%
Aggregate	2.990	2.961	2.803	93.7%

Table 5.3: Mixture Model Disaggregation: RAE House 1 Block 2 (∼ 63 days, 1 Hz), Manual Labelling

Appliance	Energy Assigned	F1-score	MAE	RMSE	GT FOE
Heatpump	73.0 %	85.3 %	82.8 W	283.9 W	27.0 %
Furnace / Hot Water	53.1 %	59.8 %	66.0 W	85.2 W	19.6 %
Fridge	97.2 %	46.5 %	58.4 W	84.2 W	7.7 %
Dryer	94.1 %	77.7 %	4.2 W	70.9 W	7.5 %
Dishwasher	86.2 %	64.2 %	18.2 W	115.2 W	2.6 %

Table 5.4: Mixture Model Disaggregation: RAE House 2 Block 1 (∼ 63 days, 1 Hz), Manual Labelling

Appliance	Energy Assigned	F1-score	MAE	RMSE	GT FOE
Dryer	71.4 %	52.8 %	15.3 W	219.5 W	20.5 %
Boiler	97.2 %	22.8 %	32.7 W	73.6 W	14.4 %
Fridge	85.7 %	79.0 %	23.0 W	48.6 W	19.5 %
Dishwasher	40.1 %	57.4 %	8.1 W	74.9 W	4.3 %
Central Vac	192.1 %	86.7 %	1.2 W	23.5 W	0.3 %

Table 5.5: Mixture Model Disaggregation: REFIT House 2 (∼ 530 days, 1/8 Hz), Manual Labelling

Appliance	Energy Assigned	F1-score	MAE	RMSE	GT FOE
Dishwasher	126.4 %	67.4 %	49.7 W	320.1 W	13.0 %
Fridge/Freezer	103.7 %	61.7 %	31.7 W	48.8 W	7.5 %
Kettle	108.8 %	40.0 %	30.6 W	292.3 W	5.0 %

Table 5.6: Labelling evaluation: RAE House 1

Mixture Model Disaggregation			
Appliance	**Windows correctly assigned**	**Avg right confidence**	**Avg wrong confidence**
Fridge/Freezer	19.3 %	70.0 %	74.5 %
Dryer	59.4 %	63.4 %	70.6 %
Dishwasher	6.2 %	55.0 %	74.0 %
Heating	65.2 %	61.3 %	67.8 %
Factorial HDP-HSMM Disaggregation			
Fridge/Freezer	30.4 %	83.5 %	65.1 %
Dryer	20.3 %	60.1 %	67.6 %
Dishwasher	15.0 %	58.3 %	71.6 %
Heating	63.0 %	80.4 %	51.8 %

Table 5.7: Labelling evaluation: RAE House 2

Appliance	Windows correctly assigned	Avg right confidence	Avg wrong confidence
Mixture Model Disaggregation			
Fridge/Freezer	46.9 %	77.0 %	50.0 %
Dryer	45.3 %	65.0 %	58.7 %
Dishwasher	21.9 %	55.1 %	82.9 %
Heating	29.7 %	78.5 %	68.0 %
Factorial HDP-HSMM Disaggregation			
Fridge/Freezer	20.4 %	80.7 %	64.4 %
Dryer	26.6 %	66.3 %	70.2 %
Dishwasher	4.7 %	53.6 %	74.2 %
Heating	59.5 %	82.1 %	52.3 %

Table 5.8: Labelling evaluation: Unseen Dataport Home

Appliance	Windows correctly assigned	Avg right confidence	Avg wrong confidence
Mixture Model Disaggregation			
Fridge/Freezer	33.0 %	84.1 %	70.6 %
Dryer	93.8 %	93.0 %	49.7 %
Dishwasher	10.9 %	56.3 %	64.8 %
Microwave	94.5 %	93.7 %	56.3 %
Heating	46.3 %	71.7 %	75.9 %
Factorial HDP-HSMM Disaggregation			
Fridge/Freezer	31.7 %	81.4 %	73.8 %
Dryer	86.6 %	91.7 %	59.2 %
Dishwasher	13.8 %	83.3 %	48.6 %
Microwave	94.1 %	95.4 %	52.9 %
Heating	46.5 %	75.7 %	62.6 %

Table 5.9: Labelling evaluation: Ground Truth Sub-meters

Rae House 1			
Appliance	**Windows correctly assigned**	**Avg right confidence**	**Avg wrong confidence**
Fridge/Freezer	25.2 %	70.5 %	74.8 %
Dryer	73.4 %	80.1 %	58.3 %
Dishwasher	84.4 %	94.2 %	80.5 %
Washing Machine	95.3 %	84.7 %	60.0 %
Heating	45.6 %	73.9 %	73.9 %
Unseen Dataport Home			
Fridge/Freezer	82.5%	90.1 %	69.8 %
Dryer	97.8 %	93.4 %	49.2 %
Dishwasher	91.9 %	91.2 %	63.7 %
Washing Machine	73.8 %	79.7 %	76.9 %
Heating	68.6 %	90.7 %	78.1 %

Table 5.10: Factorial HDP-HSMM Disaggregation: RAE House 1 Block 2 ($\sim$ 63 days, 1 Hz), Manual Labelling

Appliance	**Energy Assigned**	**F1-score**	**MAE**	**RMSE**	**GT FOE**
Heatpump	71.0 %	83.3 %	89.4 W	306.5 W	27.0 %
Furnace / Hot Water	90.0 %	74.2 %	59.6 W	80.5 W	19.6 %
Fridge	126.4 %	56.6 %	53.4 W	85.6 W	7.7 %
Dryer	77.0 %	87.2 %	13.6 W	228.5 W	7.5 %
Dishwasher	75.2 %	42.7 %	18.7 W	115.6.3 W	2.6 %

Table 5.11: Factorial HDP-HSMM Disaggregation: RAE House 2 Block 1 ($\sim$ 63 days, 1 Hz), Manual Labelling

Appliance	**Energy Assigned**	**F1-score**	**MAE**	**RMSE**	**GT FOE**
Dryer	153.4 %	77.0 %	43.0 W	428.1 W	20.5 %
Boiler	103.9 %	33.5 %	29.0 W	67.4 W	14.4 %
Fridge	127.6 %	72.3 %	41.5 W	69.7 W	19.5 %
Dishwasher	66.4 %	72.4 %	6.7 W	67.5 W	4.3 %
Central Vac	259.4 %	66.5 %	2.04 W	39.7 W	0.3 %

Table 5.12: Factorial HDP-HSMM Disaggregation: REFIT House 2 ($\sim$ 530 days, 1/8 Hz), Manual Labelling

Appliance	**Energy Assigned**	**F1-score**	**MAE**	**RMSE**	**GT FOE**
Dishwasher	132.3 %	62.0 %	60.5 W	361.1 W	13.0 %
Fridge/Freezer	110.0 %	62.8 %	34.0 W	55.6 W	7.5 %
Kettle	88.8 %	63.1 %	17.9 W	212.9 W	5.0 %

Table 5.13: REFIT Cross-house ($3 \rightarrow 5$) MAE for full and cutoff training

Benchmark Method	**Full Training**	**Cutoff Training**
WindowGRU	37.83	**33.03** ↓
DAE	34.78	**33.00** ↓
RNN	32.54	**30.62** ↓
Seq2Seq	**27.17**	29.43 ↑
Seq2Point	26.85	**26.74** ↓

Chapter 6

Conclusions

This thesis has explored two self-contained NILM solutions, both of which leverage a fast, steady-state block-filter, which was developed to produce steady-state representations of complex aggregate data. This filter far outperformed an existing method, both in terms of run-time, as well as accuracy of imputed steady-state values (i.e., showed improved performance using knapsack optimization disaggregation). A proof of concept optimization scheme using conflicting RMSE and Hoyer sparsity loss was demonstrated. The locally optimal filter parameters achieved using dual annealing for various sparsity loss weights showed promise for further development. Using the first-differences of block-filtered aggregate data, a non-parametric Gaussian mixture model established appliance mode demand distributions. Duration distributions were established by an additional non-parametric mixture using a first-n pairing scheme. These rough priors were used to select final activation-deactivation pairs, further constraining the solution space using energy conservation considerations. The returned states were then labeled by greedy merging, according to the improvement in the softmax output of a Res-Net-based classifier. Although the performance of the labeling network was disappointing on average, further examination suggested that a more representative training set would likely improve labeling in the future. In addition, examining labeling performance when transient characteristics are isolated more carefully in the mixture model disaggregation would be an important next step.

The second NILM solution leverages these labeled states with their prior distributions over demand and duration to construct a non-parametric factorial Hierarchical Dirichlet Process Hidden Semi-Markov Model. This highly flexible model was modified using constraints previously explored in the literature, as well as with similar energy conservation considerations. These constraints were shown to have a large impact on model performance relative to the motivating work by Matt Johnson and Alan Willsky [9]. Furthermore, these additional constraints appear to have little effect on computational complexity of inference, at least for the number of iterations explored. An important next step in this model was mentioned in Section 4.4. The use of flexible states in inference could be useful in uncovering states missed by the mixture model disaggregation or to capture new states. Lastly,

it was mentioned that the variance of the components fed to this model were fixed, so that the variance sequence could be restricted from trying to exactly match the raw data. In the inferential framework developed by Matt Johnson, a sub-iteration routine is available, in which the amplitude of the sampled variance sequences are gradually annealed. It may be the case that certain annealing schedules can restrict unlikely state assignments, while still encouraging sampled demand to match the observed aggregate. One exciting extension of this process is that these variance sequences are modeled as simple Gaussians. It seems entirely possible to sample from other distributions instead. We might imagine using the rough, running-average transients from the mixture models to inform the distributions from which each component's variance sequence should be sampled. This could be paired with a component-specific annealing schedule to better inform inference in these factorial models.

Finally, a model-agnostic approach to data novelty was proposed under the umbrella of Bayesian Surprise. Ultimately, the concept of surprise involves comparison over distributions as they are updated given new observations. The most useful such distributions are unavoidably model-specific. For example, surprise could be defined relative to the latent space in methods such as auto-encoders, or it could be defined relative to nonlinear auto-regressive dependencies in more complex graphical models. Nevertheless, there are features intrinsic to the data itself that could be used to predict the usefulness of more data in a model-agnostic way. This work explored a postdictive surprise defined over the likelihood of a non-parametric GMM. The mixture model was updated with windows of events defined by first-differences in the block-filtered raw signal exceeding a pre-specified threshold. Furthermore, we explored a transitional surprise defined in a Markovian sense, which was described by the transitional relationships between latent states as determined by the state assignments of the GMM. This crude approximation to the system dynamics was shown to be useful relative to a strictly postdictive notion of surprise, at least in an HMM-based application. An approximate joint threshold was determined by examining the MAE performance of five benchmark methods supported by NILMTK over three REFIT homes. This threshold was used to explore the potential regularizing effect of a surprise-based training cutoff. This is similar to the use of early-stoppage, which is a common method to protect against over-fitting and aid in the transferability of learned parameters. Relative to training over the full REFIT house 3, training on the surprise-based subset showed improved MAE for all but one method when testing over REFIT house 5. This supports the claim that Bayesian surprise can be a useful metric in predicting over-fitting and potentially improve generalization to unseen houses or datasets. Moreover, postdictive surprise using non-parametric mixture models naturally extends to online settings, where deployed NILM algorithms quickly become obsolete without the flexibility to adapt to new appliances or appliance replacements. An important extension of the current work is to explore cross-dataset performance. Similarities between the two REFIT houses in table 5.13 is likely unrepresentative of the general use-case for NILM. Also left to future work is to

explore alternative models for transitional surprise such as constructing super-states from the observed appliance modes. Additionally, future work may include sub-modelling for each component observed in the non-parametric mixture model. This would permit modelling multiple appliance modes in the same range of power values, where Bayesian surprise could further be computed over the sub-model parameters. This extension would be highly valuable to an online setting to track new appliance mode activations. Furthermore, the surprise associated with changes in these sub-model parameterizations could prove to be useful for fault detection in major appliances.

References

[1] 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 *The Eleventh Conference on Innovative Smart Grid Technologies (ISGT 2020)*, 2019 (cit. on p. iv).

[2] A. Harell, R. Jones, S. Makonin and I. V. Bajic, *Powergan: Synthesizing appliance power signatures using generative adversarial networks*, 2020. arXiv: 2007.13645 [eess.SP] (cit. on pp. iv, 52).

[3] R. Jones, C. Klemenjak, S. Makonin and I. Bajíc, 'Stop! exploring bayesian surprise to better train nilm', in *The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation; NILM Workshop*, ser. BuildSys '20, Accepted for publication, ACM, 2020 (cit. on p. iv).

[4] Rainforest Automation, *The effect of real-time alerts on behavioural demand response: BC Hydro case study*, Available online at: rainforestautomation.com/wp-content/uploads/2019/10/Rainforest_PeakSaver_WhitePaper.pdf, 2019 (cit. on p. 1).

[5] Y. Zheng, Z. Dong, F. Luo, K. Meng, J. Qiu and K. Wong, 'Optimal allocation of energy storage system for risk mitigation of discos with high renewable penetrations', *IEEE Transactions on Power Systems*, vol. 29, pp. 212–220, Jan. 2014. DOI: 10.1109/TPWRS.2013.2278850 (cit. on p. 1).

[6] J. I. Considine, *Handbook of Energy Politics*. Cheltenham, UK: Edward Elgar Publishing, 2018, ISBN: 9781784712297 (cit. on p. 1).

[7] 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. 1, 73).

[8] A. Zoha, A. Gluhak, M. Imran and S. Rajasegarar, 'Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey', *Sensors (Basel, Switzerland)*, vol. 12, pp. 16 838–16 866, Dec. 2012. DOI: 10.3390/s121216838 (cit. on p. 2).

[9] M. J. Johnson and A. S. Willsky, 'Bayesian nonparametric hidden semi-markov models', *Journal of Machine Learning Research*, vol. 14, pp. 673–701, 2013 (cit. on pp. 3, 9, 11, 15, 17, 27, 29, 49, 52, 86, 95).

[10] 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. 3, 54, 57, 76).

[11] 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, 2017 (cit. on pp. 3, 54, 87).

[12] P. Baldi, 'A computational theory of surprise', in *Information, Coding and Mathematics*, Springer, 2002, pp. 1–25 (cit. on p. 4).

[13] S. Makonin, F. Popowich, I. V. Bajić, B. Gill and L. Bartram, 'Exploiting HMM sparsity to perform online real-time nonintrusive load monitoring', *IEEE Transactions on Smart Grid*, vol. 7, no. 6, pp. 2575–2585, 2016. DOI: 10.1109/TSG.2015.2494592 (cit. on pp. 4, 10, 49, 76).

[14] V. Petrushin, 'Hidden markov models: Fundamentals and applications', *Online Symposium for Electronics Engineers*, 2000 (cit. on p. 5).

[15] D. Geiger, T. Verma and J. Pearl, 'D-separation: From theorems to algorithms', in *Uncertainty in Artificial Intelligence*, ser. Machine Intelligence and Pattern Recognition, M. HENRION, R. D. SHACHTER, L. N. KANAL and J. F. LEMMER, Eds., vol. 10, North-Holland, 1990, pp. 139–148. DOI: https://doi.org/10.1016/B978-0-444-88738-2.50018-X. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978044488738250018X (cit. on p. 6).

[16] A. Gelman, J. B. Carlin, H. S. Stern and D. B. Rubin, *Bayesian Data Analysis*. Boca Raton, FL, USA: Chapman & Hall/CRC, 1995, ISBN: 39345016723441 (cit. on pp. 8, 25).

[17] L. E. Baum and T. Petrie, 'Statistical inference for probabilistic functions of finite state markov chains', *Ann. Math. Statist.*, vol. 37, no. 6, pp. 1554–1563, Dec. 1966. DOI: 10.1214/aoms/1177699147 (cit. on p. 10).

[18] T. S. Ferguson, 'A bayesian analysis of some nonparametric problems', *Ann. Statist.*, vol. 1, no. 2, pp. 209–230, Mar. 1973. DOI: 10.1214/aos/1176342360 (cit. on p. 12).

[19] E. Fox, 'Bayesian nonparametric learning of complex dynamical phenomena', Ph.D. dissertation, Massachusetts Institute of Technology, 2009 (cit. on pp. 12–15, 22, 24).

[20] J. Sethuraman, 'A constructive definition of the dirichlet prior', *Statistica Sinica*, vol. 4, pp. 639–650, Jan. 1994 (cit. on p. 12).

[21] B. Anderson, 'The realization problem for hidden markov models', *Mathematics of Control, Signals, and Systems*, no. 12, pp. 80–120, 1999 (cit. on p. 13).

[22] D. Görür and C. E. Rasmussen, 'Dirichlet process gaussian mixture models: Choice of the base distribution', *J. Comput. Sci. Technol.*, vol. 25, no. 4, pp. 653–664, Jul. 2010 (cit. on p. 14).

[23] J. V. Gael, Y.Saatci, Y. W. Teh and Z. Ghahramani, 'Beam sampling for the infinite hidden Markov model', in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1088–1095 (cit. on p. 15).

[24] S. Ermon, *Cs-228 notes*, Online: ermongroup.github.io/cs228-notes/. (Accessed March 2, 2020), 2017 (cit. on pp. 18, 30).

[25] L. Baum, 'An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process', in *Inequalities III: Proceedings of the 3rd Symposium on Inequalities*, 1972 (cit. on p. 18).

[26] A. Viterbi, 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967 (cit. on pp. 18, 20).

[27] J. Miller, *Machine learning*, Youtube (*Mathematical Monk*), Harvard T.H. Chan School of Public Health Department of Biostatistics, 2011 (cit. on p. 19).

[28] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Taylor & Francis Group, LLC, 2015 (cit. on p. 20).

[29] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer, 2008 (cit. on pp. 22, 23, 29).

[30] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. NY, USA: Springer-Verlag, 2004, ISBN: 9781441919397 (cit. on p. 22).

[31] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, 'Equation of state calculations by fast computing machines', *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. DOI: 10.1063/1.1699114 (cit. on p. 23).

[32] J. Liu, W. Wong and A. Kong, 'Covariance structure and convergence rate of the gibbs sampler with various scans', *Journal of the Royal Statistical Society*, vol. Series B, pp. 157–169, 1995 (cit. on p. 24).

[33] L. Tierney, 'Markov chains for exploring posterior distributions', *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994 (cit. on p. 24).

[34] Y. Guédon, 'Exploring the state sequence space for hidden markov and semi-markov chains', *Computational Statistics and Data Analysis*, vol. 51, no. 5, pp. 2379–2409, 1995 (cit. on p. 26).

[35] D. M. Blei, A. Kucukelbir and J. D. McAuliffe, 'Variational inference: A review for statisticians', *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017, ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773 (cit. on pp. 30, 31).

[36] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman and D. M. Blei, *Automatic differentiation variational inference*, 2016. arXiv: 1603.00788 [stat.ML] (cit. on p. 31).

[37] D. M. Blei and M. I. Jordan, 'Variational inference for dirichlet process mixtures', *Bayesian Analysis*, vol. 1, pp. 121–144, 2005 (cit. on p. 32).

[38] T. Hastie, J. Friedman and R. Tisbshirani, *The Elements of Statistical Learning*. Springer, 2017 (cit. on p. 33).

[39] W. Mcculloch and W. Pitts, 'A logical calculus of ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943 (cit. on p. 33).

[40] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015 (cit. on pp. 35, 37, 39, 41, 43, 44).

[41] Coursera, *Course 2 & 4*, Youtube (*deeplearning.ai*), 2017 (cit. on pp. 41–43, 46).

[42] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition', in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016 (cit. on p. 45).

[43] S. Makonin and F. Popowich, 'Nonintrusive load monitoring (nilm) performance evaluation', *Energy Efficiency*, no. 8, pp. 809–814, 2015 (cit. on p. 46).

[44] G. W. Hart, 'Prototype nonintrusive appliance load monitor', MIT Energy Laboratory and Electric Power Research Institute, Tech. Rep., 1985 (cit. on p. 48).

[45] J. Z. Kolter and T. Jaakkola, 'Approximate inference in additive factorial hmms with application to energy disaggregation', *Journal of Machine Learning Research Proc. Track*, vol. 22, pp. 1472–1482, 2012 (cit. on pp. 48–50, 74).

[46] M. Zhong, N. Goddard and C. Sutton, 'Signal aggregate constraints in additive factorial hmms, with application to energy disaggregation', in *Advances in Neural Information Processing Systems 27*, 2014, pp. 3590–3598 (cit. on pp. 49, 83).

[47] H. Kim, M. Marwah, M. Arlitt, G. Lyon and J. Han, 'Unsupervised disaggregation of low frequency power measurements', in *Proc. SIAM Conf. Data Mining, Mesa, AZ, USA*, 2011, pp. 747–758 (cit. on p. 49).

[48] R. Dong, L. J. Ratliff, H. Ohlsson and S. S. Sastry, 'Energy disaggregation via adaptive filtering', in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, Oct. 2013. DOI: 10.1109/allerton.2013.6736521 (cit. on p. 50).

[49] D. Piga, A. Cominola, M. Giuliani, A. Castelletti and A. E. Rizzoli, 'Sparse optimization for automated energy end use disaggregation', *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1044–1051, 2016 (cit. on p. 50).

[50] A. Rahimpour, H. Qi, D. Fugate and T. Kuruganti, 'Non-intrusive energy disaggregation using non-negative matrix factorization with sum-to-k constraint', *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4430–4441, 2017 (cit. on p. 50).

[51] ——, 'Non-intrusive load monitoring of hvac components using signal unmixing', in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2015, pp. 1012–1016 (cit. on p. 50).

[52] 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, 2017 (cit. on p. 50).

[53] J. M. Varah, 'On the numerical solution of ill-conditioned linear systems with applications to ill-posed problems', *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 257–267, 1973, ISSN: 00361429 (cit. on p. 51).

[54] J. Liao, L. Stankovic, V. Stankovic and G. Elafoudi, 'Non-intrusive appliance load monitoring using low-resolution smart meter data', *2014 IEEE Int. Conf. on Smart Grid Communications*, 2014 (cit. on p. 51).

[55] 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, 2018. DOI: 10.1109/tsg.2016.2598872 (cit. on p. 51).

[56] H.-H. Chang, 'Non-intrusive demand monitoring and load identification for energy management systems based on transient feature analyses', *Energies*, vol. 5, no. 11, pp. 4569–4589, 2012. DOI: 10.3390/en5114569 (cit. on p. 51).

[57] K. Chahine and K. El Khamlichi Drissi, 'A novel feature extraction method for nonintrusive appliance load monitoring', *Applied Computational Intelligence and Soft Computing*, vol. 2013, pp. 1–7, 2013. DOI: 10.1155/2013/686345 (cit. on p. 51).

[58] H.-H. Chang, L.-S. Lin, N. Chen and W.-J. Lee, 'Particle-swarm-optimization-based nonintrusive demand monitoring and load identification in smart meters', *IEEE Transactions on Industry Applications*, vol. 49, no. 5, pp. 2229–2236, 2013. DOI: 10.1109/tia.2013.2258875 (cit. on p. 51).

[59] L. Du, Y. Yang, D. He, R. G. Harley and T. G. Habetler, 'Feature extraction for load identification using long-term operating waveforms', *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 819–826, 2015. DOI: 10.1109/tsg.2014.2373314 (cit. on p. 51).

[60] X. Wu, Y. Gao and D. Jiao, 'Multi-label classification based on random forest algorithm for non-intrusive load monitoring system', *Processes*, vol. 7, no. 6, p. 337, 2019. DOI: 10.3390/pr7060337 (cit. on p. 51).

[61] J. Kelly and W. Knottenbelt, 'Neural nilm: Deep neural networks applied to energy disaggregation', in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, IEEE, 2015 (cit. on p. 51).

[62] J. Kim, T.-T.-H. Le and H. Kim, 'Nonintrusive load monitoring based on advanced deep learning and novel signature', *Computational Intelligence and Neuroscience*, vol. 2017, 2017, ISSN: 4216281 (cit. on p. 51).

[63] T.-T.-H. Le, J. Kim and H. Kim, 'Classification performance using gated recurrent unit recurrent neural network on energy disaggregation', in *Proceedings of the 2016 International Conference on Machine Learning and Cybernetics*, 2016 (cit. on p. 51).

[64] J.-G. Kim and B. Lee, 'Appliance classification by power signal based on multi-feature combination multi-layer lstm', *Energies*, vol. 12, 2019, ISSN: 2804 (cit. on p. 51).

[65] D. de Paiva Penha and A. R. G. Castro, 'Convolutional neural network applied to the identification of residential equipment in non-intrusive load monitoring systems', *Computer Science & Information Technology*, pp. 11–21, 2017 (cit. on p. 51).

[66] L. D. Baets, J. Ruyssinck, C. Develder, T. Dhaene and D. Deschrijver, 'Appliance classification using vi trajectories and convolutional neural networks', *Energy and Buildings*, vol. 158, pp. 32–36, 2018, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.09.087 (cit. on p. 51).

[67] A. Harell, S. Makonin and I. V. Bajic, 'Wavenilm: A causal neural network for power disaggregation from the complex power signal', in *44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, 2019 (cit. on p. 51).

[68] V. Singhal, J. Maggu and A. Majumdar, 'Simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep dictionary learning and deep transform learning', *IEEE Transactions On Smart Grid*, vol. 12, 2019, ISSN: 3 (cit. on p. 51).

[69] S. Singh and A. Majumdar, 'Deep sparse coding for non–intrusive load monitoring', *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4669–4678, 2018 (cit. on p. 51).

[70] 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)*, 2019, pp. 8330–8334 (cit. on pp. 51, 76).

[71] M. Kaselimi, A. Voulodimos, E. Protopapadakis, N. Doulamis and A. Doulamis, 'Energan: A generative adversarial network for energy disaggregation', in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1578–1582 (cit. on p. 52).

[72] Y. Pan, K. Liu, Z. Shen, X. Cai and Z. Jia, 'Sequence-to-subsequence learning with conditional gan for power disaggregation', in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3202–3206 (cit. on p. 52).

[73] T. Sirojan, B. T. Phung and E. Ambikairajah, 'Deep neural network based energy disaggregation', in *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, 2018, pp. 73–77 (cit. on p. 52).

[74] 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, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2014.07.010 (cit. on p. 52).

[75] H. Shao, M. Marwah and N. Ramakrishnan, 'A temporal motif mining approach to unsupervised energy disaggregation', *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, Jan. 2013 (cit. on p. 52).

[76] C. Klemenjak, S. Makonin and W. Elmenreich, 'Towards comparability in non-intrusive load monitoring: On data and performance evaluation', in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, IEEE, 2020, pp. 1–5 (cit. on p. 53).

[77] 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. 53).

[78] A. Harell, 'Deep learning applications in non-intrusive load monitoring', Available online at https://theses.lib.sfu.ca/file/thesis/6013, M.S. thesis, Simon Fraser University, 2020 (cit. on p. 53).

[79] 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 p. 54).

[80] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe and M. Berges, 'BLUED: a fully labeled public dataset for Event-Based Non-Intrusive load monitoring research', in *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, Beijing, China, 2012 (cit. on p. 54).

[81] Rainforest Automation, 'Household electricity survey: A study of domestic electrical product usage', *Intertek Testing & Certification Ltd*, 2012 (cit. on p. 54).

[82] A. Mishra, E. Cecchet, P. Shenoy and J. Albrecht, 'Smart * : An open data set and tools for enabling research in sustainable homes', in *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, vol. 111, 2012, p. 108 (cit. on p. 54).

[83] N. Batra, M. Gulati, A. Singh and M. Srivastava, 'It's different: Insights into home energy consumption in india', in *Proceedings of the 5th ACM Workshop on Embedded Systems for Energy-Efficient Buildings*, Nov. 2013. DOI: 10.1145/2528282.2528293 (cit. on p. 54).

[84] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake and S. Santini, 'The eco data set and the performance of non-intrusive load monitoring algorithms', in *Proceedings of the 1st ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2014, pp. 80–89 (cit. on p. 54).

[85] S. Nambi, A. R. Lua and R. V. Prasad, 'Loced: Location-aware energy disaggregation framework', in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2015, pp. 45–54 (cit. on p. 54).

[86] O. Parson, G. Fisher, A. Hersey, N. Batra, J. Kelly, A. Singh, W. Knottenbelt and A. Rogers, 'Dataport and nilmtk: A building data set designed for non-intrusive load monitoring', in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2015, pp. 210–214 (cit. on p. 54).

[87] J. Kelly and W. Knottenbelt, 'The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes', *Scientific Data*, vol. 2, no. 150007, DOI: 10.1038/sdata.2015.7 (cit. on p. 54).

[88] 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, no. 1, p. 160 037, 2016. DOI: 10.1038/sdata.2016.37. [Online]. Available: https://doi.org/10.1038/sdata.2016.37 (cit. on p. 54).

[89] D. Chen, D. Irwin and P. Shenoy, 'Smartsim: A device-accurate smart home simulator for energy analytics', in *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, 2016, pp. 686–692 (cit. on p. 54).

[90] S. Henriet, U. Simsekli, G. Richard and B. Fuentes, 'Synthetic dataset generation for non-intrusive load monitoring in commercial buildings', in *Proc. 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, 2017, pp. 1–2 (cit. on p. 54).

[91] A. Reinhardt and C. Klemenjak, 'How does load disaggregation performance depend on data characteristics? insights from a benchmarking study', in *Proc. 11th ACM International Conference on Future Energy Systems (e-Energy)*, 2020 (cit. on p. 54).

[92] C. Klemenjak, C. Kovatsch, M. Herold and W. Elmenreich, 'A synthetic energy dataset for non-intrusive load monitoring in households', *Scientific Data*, vol. 7, no. 1, pp. 1–17, 2020 (cit. on p. 54).

[93] S. Aminikhanghahi and D. J. Cook, 'A survey of methods for time series change point detection', *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, May 2017, ISSN:

0219-1377. DOI: 10.1007/s10115-016-0987-z. [Online]. Available: https://doi.org/10.1007/s10115-016-0987-z (cit. on p. 55).

[94] P. O. Hoyer, 'Non-negative matrix factorization with sparseness constraints', *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004 (cit. on p. 61).

[95] Y. Xiang, D. Sun, W. Fan and X. Gong, 'Generalized Simulated Annealing Algorithm and Its Application to the Thomson Model', *Physics Letters A*, vol. 223, pp. 216–220, 1997 (cit. on p. 61).

[96] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: 1502.03167 [cs.LG] (cit. on p. 70).

[97] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG] (cit. on p. 70).

[98] L. Itti and P. F. Baldi, 'Bayesian surprise attracts human attention', in *Advances in Neural Information Processing Systems*, 2006, pp. 547–554 (cit. on p. 75).

[99] P. Baldi and L. Itti, 'Of bits and wows: A bayesian theory of surprise with applications to attention', *Neural Networks*, vol. 23, no. 5, pp. 649–666, 2010 (cit. on p. 75).

[100] A. Kolossa, B. Kopp and T. Fingscheidt, 'A computational analysis of the neural bases of bayesian inference', *NeuroImage*, vol. 106, pp. 222–337, Feb. 2015. DOI: 10.1016/j.neuroimage.2014.11.007 (cit. on p. 75).

[101] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh and M. Srivastava, 'NILMTK: An Open Source Toolkit for Non-Intrusive Load Monitoring', in *5th ACM International Conference on Future Energy Systems (e-Energy)*, 2014 (cit. on p. 76).

[102] R. Bonfigli, A. Felicetti, E. Principi, M. Fagiani, S. Squartini and F. Piazza, 'Denoising autoencoders for non-intrusive load monitoring: Improvements and comparative evaluation', *Energy and Buildings*, vol. 158, pp. 1461–1474, 2018 (cit. on p. 76).

[103] J. Kelly and W. Knottenbelt, 'Neural NILM: Deep neural networks applied to energy disaggregation', in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2015 (cit. on p. 76).

[104] O. Krystalakos, C. Nalmpantis and D. Vrakas, 'Sliding window approach for online energy disaggregation using artificial neural networks', in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence (SETN)*, 2018 (cit. on p. 76).

[105] C. Zhang, M. Zhong, Z. Wang, N. Goddard and C. Sutton, 'Sequence-to-point learning with neural networks for non-intrusive load monitoring', in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018 (cit. on p. 76).

[106] A. Rodriguez-Silva and S. Makonin, 'Universal Non-Intrusive Load Monitoring (UNILM) Using Filter Pipelines, Probabilistic Knapsack, and Labelled Partition Maps', in *The 11th IEEE PES Asia-Pacific Power and Energy Engineering Conference 2019 (APPEEC)*, 2019 (cit. on pp. 80, 81, 90).