Inference methods for nonintrusive load monitoring applications

André Tiago Ventura Tomás

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. João Manuel de Freitas Xavier

Examination Committee

President: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. João Manuel de Freitas Xavier
Members of the Committee: Prof. José Manuel Bioucas Dias

April 2014
Acknowledgments

A special thank you to Professor João Xavier, for the patience and his willingness to point the way.
And to my parents for their support.
Abstract

Non-intrusive load monitoring methods are becoming increasingly relevant in the context of energy management as they provide a practical way to monitor several equipments with a minimum amount of sensor installment. The objective of this thesis is to develop a flexible suite of inference methods capable of partitioning an aggregate quantity into the sum of its parts. For this purpose we developed two types of methods: non-probabilistic heuristic based methods and probabilistic methods. Furthermore we analyzed the inclusion of side data into some of these methods. The heuristic methods are fully data-driven, fast and easily implemented. The probabilistic methods each assume an underlying probabilist model and are, generally, computationally heavier than the non-probabilistic methods.

For training and evaluation purposes we used a dataset consisting of the direct measurements of the consumptions of ten groups of machines, plus their aggregate consumption, obtained with the rate of 1 sample per 15 minutes. For further evaluation purposes we generated two synthetic datasets from the model parameters from two of the methods. Results obtained from the three datasets used show mixed results in the accuracy of the estimations across all machines.

Keywords

Non-intrusive load monitoring, energy management, inference methods
Resumo

Os métodos de monitorização de cargas não-intrusiva estão se a tornar cada vez mais relevantes no contexto da gestão energética pois permitem a monitorização de vários equipamentos, com uma instalação mínima de sensores, de uma forma prática. Esta tese tem como objectivo desenvolver um conjunto fléxivel de métodos de inferência. Com este objectivo em mente desenvolvemos dois tipos de métodos: métodos não-probabilísticos baseados em heurísticas e métodos probabilísticos. Adicionalmente analizamos a inclusão de informação extra em alguns desses métodos. Os métodos heurísticos não assumem modelos probabilísticos (data-driven), são rápidos e fáceis de implementar. Cada método probabilístico assume um modelo probabilístico para os dados e são, por norma, computacionalmente mais pesados que os métodos não-probabilísticos.

Para efeitos de treino e de validação, usámos um dataset que consiste na medição directa dos consumos de dez grupos de máquinas, mais o seu agregado, obtidos de 15 em 15 minutos. Adicionalmente gerámos dois datasets sintéticos dos parâmetros dos modelos de dois dos métodos, para efeitos de avaliação. Os resultados obtidos dos três datasets mostram resultados mistos na precisão das estimativas de todas as máquinas.

Palavras Chave

Monitorização de cargas não-intrusiva, gestão energética, métodos de inferência
Contents

1 Introduction .................................................. 1
   1.1 Motivation .................................................. 2
   1.2 State of The Art ............................................. 2
   1.3 Contributions ............................................... 3
   1.4 Thesis Outline .............................................. 3

2 Methods ....................................................... 5
   2.1 Introduction ............................................... 6
   2.2 Instantaneous Proportional Fit (IPF) ...................... 6
   2.3 Memory Proportional Fit (MPF) ............................ 7
   2.4 Decoupled Markov Chains ................................ 7
   2.5 Coupled Markov chains .................................... 12
   2.6 Gaussian model ............................................ 17
   2.7 Minimum Mean-Squared Estimation (MMSE) .............. 19
   2.8 Methods with side information ............................ 20

3 Experimental results ........................................ 23
   3.1 Introduction ............................................... 24
   3.2 Synthetic Datasets ......................................... 29
   3.3 Real Data ................................................... 47
      3.3.1 Alternative initializations for the Markov Chains methods ........... 58
   3.4 Side information ........................................... 62

4 Conclusions .................................................. 67

Bibliography ..................................................... 69

Appendix A Derivation of the majorization function .......... A-1
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Probabilistic graphical model for section 2.4: machines are decoupled Markov chains</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Probabilistic graphical model for section 2.6: machines are jointly Gaussian within time blocks of size $\tau$ (here, $\tau = 1$)</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Metric 1: average absolute error</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Histogram of the relative errors</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Confusion matrix</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Behavior of a machine</td>
<td>27</td>
</tr>
<tr>
<td>3.5</td>
<td>Absolute average errors: Heuristic and MMSE methods</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>Absolute average errors: Markov Chains methods</td>
<td>30</td>
</tr>
<tr>
<td>3.7</td>
<td>Absolute average errors: Gaussian model methods</td>
<td>31</td>
</tr>
<tr>
<td>3.8</td>
<td>Absolute average errors: Markov dataset</td>
<td>32</td>
</tr>
<tr>
<td>3.9</td>
<td>Absolute average errors: Gaussian dataset</td>
<td>33</td>
</tr>
<tr>
<td>3.10</td>
<td>Histogram of the relative errors for Machine 6: Markov dataset</td>
<td>34</td>
</tr>
<tr>
<td>3.11</td>
<td>Confusion matrix of the Decoupled Markov Chains method for machine 6: Markov dataset</td>
<td>35</td>
</tr>
<tr>
<td>3.12</td>
<td>Confusion matrix of the IPF method for machine 6: Markov dataset</td>
<td>36</td>
</tr>
<tr>
<td>3.13</td>
<td>Histogram of the relative errors for machine 8: Markov dataset</td>
<td>37</td>
</tr>
<tr>
<td>3.14</td>
<td>Confusion matrix of the MMSE method for machine 8: Markov dataset</td>
<td>37</td>
</tr>
<tr>
<td>3.15</td>
<td>Histogram of the relative errors for machine 5: Markov dataset</td>
<td>38</td>
</tr>
<tr>
<td>3.16</td>
<td>Confusion matrix of the Coupled Markov Chains method for machine 5: Markov dataset</td>
<td>38</td>
</tr>
<tr>
<td>3.17</td>
<td>Confusion matrix of the Gaussian model(Block size 3) method for machine 5: Markov dataset</td>
<td>39</td>
</tr>
<tr>
<td>3.18</td>
<td>Histogram of the relative errors for machine 6: Gaussian dataset</td>
<td>39</td>
</tr>
<tr>
<td>3.19</td>
<td>Histogram of the relative errors for machine 7: Gaussian dataset</td>
<td>40</td>
</tr>
<tr>
<td>3.20</td>
<td>Confusion matrix of the Gaussian model (block size 1) method for machine 7: Gaussian dataset</td>
<td>40</td>
</tr>
<tr>
<td>3.21</td>
<td>Histogram of the relative errors for machine 5: Gaussian dataset</td>
<td>41</td>
</tr>
<tr>
<td>3.22</td>
<td>Histogram of the relative errors for machine 9: Gaussian dataset</td>
<td>41</td>
</tr>
<tr>
<td>3.23</td>
<td>Confusion matrix of the MPF method for machine 9: Gaussian dataset</td>
<td>42</td>
</tr>
<tr>
<td>3.24</td>
<td>Snapshot of machine 6 in a random day: Markov Chains and Gaussian models - Markov dataset</td>
<td>43</td>
</tr>
<tr>
<td>3.25</td>
<td>Snapshot of machine 6 in a random day: MMSE and Heuristics - Markov dataset</td>
<td>44</td>
</tr>
<tr>
<td>3.26</td>
<td>Snapshot of machine 6 in a random day: Markov Chains and Gaussian models - Gaussian dataset</td>
<td>45</td>
</tr>
<tr>
<td>3.27</td>
<td>Snapshot of machine 6 in a random day: MMSE and Heuristics - Gaussian dataset</td>
<td>45</td>
</tr>
<tr>
<td>3.28</td>
<td>Snapshot of machine 4 in a random day: Markov Chains and Gaussian models - Markov dataset</td>
<td>46</td>
</tr>
<tr>
<td>3.29</td>
<td>Snapshot of machine 4 in a random day: MMSE and Heuristics - Markov dataset</td>
<td>46</td>
</tr>
<tr>
<td>3.30</td>
<td>Real data: Average absolute errors</td>
<td>47</td>
</tr>
<tr>
<td>3.31</td>
<td>Histogram of the Relative errors for machine 6: real data</td>
<td>48</td>
</tr>
<tr>
<td>3.32</td>
<td>Confusion matrix of the Gaussian model (block size 1) method for machine 6: real data</td>
<td>49</td>
</tr>
<tr>
<td>3.33</td>
<td>Histogram of the Relative errors for machine 1: real data</td>
<td>50</td>
</tr>
<tr>
<td>3.34</td>
<td>Confusion matrix of the MMSE method for machine 1: real data</td>
<td>50</td>
</tr>
<tr>
<td>3.35</td>
<td>Histogram of the Relative errors for machine 5: real data</td>
<td>51</td>
</tr>
<tr>
<td>3.36</td>
<td>Confusion matrix of the MPF method for machine 5: real data</td>
<td>51</td>
</tr>
<tr>
<td>3.37</td>
<td>Snapshot of machine 4 in a random day: Markov Chains and Gaussian models - real data</td>
<td>52</td>
</tr>
<tr>
<td>3.38</td>
<td>Snapshot of machine 4 in a random day: MMSE and Heuristics - real data</td>
<td>53</td>
</tr>
<tr>
<td>3.39</td>
<td>Snapshot of machine 8 in a random day: Markov Chains and Gaussian models - real data</td>
<td>54</td>
</tr>
<tr>
<td>3.40</td>
<td>Snapshot of machine 8 in a random day: MMSE and Heuristics - real data</td>
<td>55</td>
</tr>
<tr>
<td>3.41</td>
<td>Snapshot of machine 10 in a random day: Markov Chains and Gaussian models - real data</td>
<td>56</td>
</tr>
<tr>
<td>3.42</td>
<td>Snapshot of machine 10 in a random day: MMSE and Heuristics - real data</td>
<td>56</td>
</tr>
<tr>
<td>3.43</td>
<td>Snapshot of machine 5 in a random day: Markov Chains and Gaussian models - real data</td>
<td>57</td>
</tr>
<tr>
<td>3.44</td>
<td>Different Markov Chains initializations: Average absolute errors - real data</td>
<td>59</td>
</tr>
<tr>
<td>3.45</td>
<td>Different Markov Chains initializations: Histogram of relative errors for machine 6 - real data</td>
<td>60</td>
</tr>
<tr>
<td>3.46</td>
<td>Different Markov Chains initializations: Histogram of relative errors for machine 8 - real data</td>
<td>60</td>
</tr>
<tr>
<td>3.47</td>
<td>Different Markov Chains initializations: Histogram of relative errors for machine 10 - real data</td>
<td>60</td>
</tr>
<tr>
<td>3.48</td>
<td>Side information: Average absolute errors for the Decoupled Markov Chains method when machine 8 is observed</td>
<td>61</td>
</tr>
<tr>
<td>3.49</td>
<td>Side information: Average absolute errors for the Gaussian model (block size 1) method when machine 7 is observed</td>
<td>63</td>
</tr>
<tr>
<td>3.50</td>
<td>Side information: Average absolute errors for the MMSE method when machine 8 is observed</td>
<td>64</td>
</tr>
<tr>
<td>3.51</td>
<td>Side information: Average absolute errors for the Decoupled Markov Chains method</td>
<td>64</td>
</tr>
</tbody>
</table>
3.52 Side information: Average absolute errors for the Gaussian model (block size 1) method 65

3.53 Side information: Average absolute errors for the MMSE method 65
Abbreviations

**NILM** Non-Intrusive Load Monitoring

**P** Real Power

**Q** Reactive Power

**FSM** Finite State Machine

**kWh** Kilowatt hour

**IPF** Instantaneous Proportional Fit

**MPF** Memory Proportional Fit

**MMSE** Minimum Mean-Squared Estimation

**MAP** Maximum a posteriori

**ML** maximum likelihood

**MM** Majorization-minorization

**pmf** probability mass function

**KL** Kullback-Leibler
1 Introduction

## Contents

1.1 Motivation .......................... 2
1.2 State of The Art ...................... 2
1.3 Contributions ....................... 3
1.4 Thesis Outline ....................... 3
1.1 Motivation

With the increasing in both energy demand and energy production costs, energy management is becoming increasingly more relevant. The efficient use of energy has several upsides, that reflect not only on the cost to the consumer, but also on the environmental impact energy production has. Reducing energy costs can be easily achieved by smartly scheduling the use of equipments (for example using a high load machine, such as a dishwasher, on off-peak hours when the energy cost is cheaper). However, a single household contains dozens of electrical equipments and monitoring and managing all these appliances can be a costly and complex task. Non-Intrusive Load Monitoring (NILM) methods allow the monitorization of the consumption of individual appliances, without the need to install individual sensors in each of them, by looking at their aggregate consumption and partitioning it into its constituents. This is very practical since there is no need to install a large number of sensors which reduces costs of installation making NILM methods ideal for large scale applications. The data collected this way can then be used to promote a smarter use of the equipments.

1.2 State of The Art

One of the initial NILM methods, proposed by Hart [1] in 1992, followed a pattern recognition based approach, using Real Power ($P$) and Reactive Power ($Q$) as features to differentiate appliances. The load variation these appliances incur when they are turned on are mapped to the $PQ$ feature space, where they form clusters. Changes in the load signal, commonly referred to as events, are mapped into the $PQ$ plane, where they are compared to the known clusters based on the distance between the existing clusters and the location of the load variation. This simple method is unable, however, to recognize appliances with similar $PQ$ features. The authors of [2] further extend this method by proposing a staged algorithm that generates hypotheses that might explain an event and then evaluates these hypotheses using well defined metrics, such as, but not only, the one suggested by [1]. At the end of each stage if an hypothesis scored poorly it is dismissed, otherwise the algorithm continues its evaluation.

A different approach is used in [3]. The proposed method is unsupervised and optimization based. This method creates hypothetical Finite State Machines (FSMs) by running a genetic algorithm to cluster detected events. Then, for each FSM a shortest path problem is solved in order to estimate the behavior of each FSM throughout the day. Unlike the pattern recognition methods previously mentioned, this method only considers variations in Real Power, disregarding the Reactive Power. The method suggested by [4] is similar, however, instead of assuming each appliance is a FSM they assume all events are generated by appliances with two states (on and off) and that appliances with more than two state produce events that can be explained by two-state appliances. After the clustering step is completed a matching pursuit algorithm is used to match each event to one or more sources.

The authors of [5] suggested a supervised and optimization based approach where the problem of disaggregating the total load is formulated as an integer quadratic programming problem. The
proposed method uses current waveforms of appliances, instead of power, meaning that, unlike the previous methods, only the current needs to be measured, instead of both voltage and current. This also means no sort of preprocessing of the measured data is required, since the method is capable of directly using the measurements. The use of support vector machines and neural-networks is suggested in [6] to classify different loads, by extracting features from the current waveform. These methods also require a period of training in order to successfully identify different appliances from their harmonic “signatures”.

1.3 Contributions

This thesis develops a suite of probabilistic inference methods to partition the aggregated consumption of several equipments. These methods focus on estimating the power consumption of each equipment, without any sort of event detection, based solely on the observed aggregate consumption. The proposed methods allow the exploration of several degrees of model complexity (capturing, or disregarding, time correlation, dependence, or independence, between machines, etc) and online computational feasibility. From a theoretical viewpoint, we show that projecting time series onto the space of uncoupled markov chains reduces to a graph matching problem. Also, we devise a majorization-minorization strategy to infer the hidden states of observation coupled Markov Chains. These theoretical results may be of independent interest for other inference problems. In order to train and evaluate the performance of the proposed methods a dataset comprised of measurements from ten groups of equipments (not individual appliances, however the methods are easily generalized) from a fast-food restaurant was used.

1.4 Thesis Outline

In chapter 2 we present the suite of probabilistic, as well as non-probabilistic, methods, starting with the formulation of the optimization problems and followed by the algorithms used to solve them. We also provide a way to incorporate side information in some of the methods in this chapter. In chapter 3 we analyze the performance of the suite of methods in two parts. For the first part we generated two synthetic datasets, from the probabilistic models, assumed by two of the methods, inferred from the measured consumptions of each machine. In the second part we use part of the dataset acquired from the restaurant to measure the performance of the methods, comparing their estimations with the measured consumptions. Finally, we conclude in chapter 4 and provide possible future work.
2 Methods

Contents

2.1 Introduction ........................................................... 6
2.2 Instantaneous Proportional Fit (IPF) ............................... 6
2.3 Memory Proportional Fit (MPF) .................................. 7
2.4 Decoupled Markov Chains ......................................... 7
2.5 Coupled Markov chains .............................................. 12
2.6 Gaussian model ...................................................... 17
2.7 Minimum Mean-Squared Estimation (MMSE) .................. 19
2.8 Methods with side information .................................... 20
2.1 Introduction

In this chapter, we develop several methods to infer the machines’ consumptions from the observed aggregate consumptions. The methods differ in their modeling assumptions and permit to cover a wide range of possibilities. We start by simple, non-probabilistic based heuristics in sections 2.2 and 2.3. In section 2.4, we model the $M$ machines’ time series as independent Markov chains. This allows to capture time correlation while maintaining the online inference tractable; we follow a majorization-minorization approach combined with a dynamic-programming strategy to infer the states of the Markov chains, from the observed aggregate consumptions. In section 2.5, we pair the $M$ machines into $M/2$ virtual ones which are then modeled as independent Markov chains. This allows to capture some correlation between machines, and still makes the online inference feasible. An interesting problem concerns the pairing of machines; we show that the problem reduces to a classical matching problem on a certain graph (a result of independent interest for time series analysis). In section 2.6, we model the machines’ consumptions’ as Gaussian distributions in a flexible framework that permits to explore several types of correlation. In section 2.7, we propose a minimum mean-square error approach. Finally, in section 2.8 we incorporate side information into some of the methods previously developed.

2.2 Instantaneous Proportional Fit (IPF)

This heuristic is fully data-driven (no statistical model assumed) and fast to implement. For a given slot $t$, we first look into the training set to obtain the relative contribution of each machine to the aggregate consumption. That is, we compute

$$\eta_m(t) = \frac{1}{D} \sum_{d=1}^{D} \frac{x_m^{(d)}(t)}{y^{(d)}(t)}$$

(2.1)

where $x_m^{(d)}(t)$ is the observed consumption of the $m$th machine in slot $t$ and training day $d$, and

$$y^{(d)}(t) = \sum_{m=1}^{M} x_m^{(d)}(t)$$

is the associated observed aggregate consumption. In words, $\eta_m(t) \in [0,1]$ represents the percentage of the aggregate consumption in slot $t$ attributable to the $m$th machine (on average).

Given an observed aggregate consumption $y(t)$, we estimate $x_m(t)$ by scaling it to the observation via $\eta_m(t)$:

$$\hat{x}_m(t) = \eta_m(t) y(t).$$

(2.2)

These estimates are consistent in the sense that the observation is fully “explained”:

$$y(t) = \sum_{m=1}^{M} \hat{x}_m(t).$$

Note that the major computation (2.1) can be done offline; also, the full training set can be discarded afterwards (we only need to keep the $\eta_m(t)$’s). The online computation is the simple multiplication (2.2).
We call this approach IPF because machines are proportionally fitted to the observation in slot $t$, based only on the aggregate consumption $y(t)$ — the information $y(1), y(2), \ldots, y(t-1)$ is ignored (i.e., the “system” $y(\cdot) \mapsto \hat{x}_m(\cdot)$ is instantaneous).

### 2.3 Memory Proportional Fit (MPF)

This heuristic is fully data-driven and uses the same strategy of IPF section 2.2: machines are scaled to the observation, i.e., the $m$th machine in slot $t$ is estimated as

$$\hat{x}_m(t) = \eta_m(t)y(t), \quad (2.3)$$

(see (2.2)). The difference lies in how the scaling factor $\eta_m(t)$ is computed.

In MPF, we use the current day’s history

$$\mathcal{Y}(t) := (y(1), y(2), \ldots, y(t-1), y(t))$$

to determine the scaling factor. We start by locating, in the training set, the most similar day on the basis of this history:

$$d^* = \arg\min_{d=1, \ldots, D} \|\mathcal{Y}(d)(t) - \mathcal{Y}(t)\|, \quad (2.4)$$

where

$$\mathcal{Y}(d)(t) := (y^{(d)}(1), y^{(d)}(2), \ldots, y^{(d)}(t-1), y^{(d)}(t))$$

is the history of aggregate consumptions up to time $t$ in training day $d$.

Once $d^*$ is found, we copy the scaling factors in slot $t$ for that day, i.e., we plug

$$\eta_m(t) = \frac{x^{(d^*)}_m(t)}{y^{(d^*)}(t)}$$

into (2.3).

The major step of this method is the implementation of (2.4), the search for the closest resembling training day based on the track record $\mathcal{Y}(t)$. In particular, this forces the full training set to be available at online computation (a major difference with respect to IPF).

The method MPF as presented here estimates $\hat{x}_m(t)$ based on $y(1), y(2), \ldots, y(t-1), y(t)$ (the “system” $y(\cdot) \mapsto \hat{x}_m(\cdot)$ is causal); it is straightforward to generalize the principle to incorporate “future” data $y(t+1), y(t+2), \ldots, y(T)$ — for example, to re-estimate $x_m(t)$ at the end of the day.

### 2.4 Decoupled Markov Chains

**Probabilistic model.** We model each machine’s time series $\{x_m(1), x_m(2), \ldots, x_m(T)\}$ as a (non homogenous) Markov chain, and we assume the chains independent of each other. This allows to capture correlation over time while maintaining the model tractable. Furthermore, we assume that the aggregate consumption is corrupted by temporally independent additive noise

$$y(t) = \sum_{m=1}^{M} x_m(t) + w(t), \quad (2.5)$$

where $w(t) \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with variance $\sigma^2$ (independent from the machines’ states). Figure 2.1 depicts the underlying probabilistic graphical model.
**Maximum a posteriori (MAP) inference.** Given a segment of observed noisy aggregate consumptions
\[ y := (y(1), y(2), \ldots, y(T)), \]
our goal is to estimate all the machines’ consumptions in that segment,
\[ x_m := (x_m(1), x_m(2), \ldots, x_m(T)), \]
for \( m = 1, \ldots, M \). We follow a MAP approach:
\[ (\hat{e}_1, \ldots, \hat{e}_M) = \arg\max_{e_1, \ldots, e_M} P(x_1, \ldots, x_M | y). \tag{2.6} \]

Before proceeding we need some notation. Let \( n_m \) be the number of states of the Markov chain corresponding to machine \( m \), and let \( c_m \in \mathbb{R}^{n_m} \) be its alphabet, i.e., the vector \( c_m \) lists all possible realizations of \( x_m(t) \). As such,
\[ x_m(t) = c_m^T e_m(t) \tag{2.7} \]
for an appropriate vector \( e_m(t) \) of the form \((0, \ldots, 0, 1, 0, \ldots, 0)\). The vector \( e_m(t) \) indicates the state of machine \( m \) in slot \( t \) and is in an one-to-one relationship with \( x_m(t) \) through (2.7). It is more convenient to work with these indicator vectors.

Equation (2.6) translates to
\[ (\hat{e}_1, \ldots, \hat{e}_M) = \arg\max_{e_1, \ldots, e_M} P(e_1, \ldots, e_M | y), \tag{2.8} \]
where \( e_m := (e_m(1), e_m(2), \ldots, e_m(T)) \) indicates the sequence of states visited by machine \( m \).

Using Bayes’ theorem and the modeling assumptions, we can rewrite (2.8) as
\[ (\hat{e}_1, \ldots, \hat{e}_M) = \arg\max_{e_1, \ldots, e_M} \sum_{t=1}^{T} \log P(y(t) | e_1(t), \ldots, e_M(t)) + \sum_{m=1}^{M} \log P(e_m(1)) + \sum_{m=1}^{M} \sum_{t=2}^{T} \log P(e_m(t) | e_m(t-1)). \tag{2.9} \]
Plugging (2.7) into (2.5) shows that
\[ y(t) \sim \mathcal{N}(c^T e(t), \sigma^2) \] (2.10)
where
\[ c := (c_1^T e_1 \cdots c_M^T e_M)^T \quad \text{and} \quad e(t) := (e(1)^T e_2(t) \cdots e_M(t)^T)^T. \] (2.11)

Thus, (2.9) corresponds to the optimization problem
\[
\minimize_{e_1, \ldots, e_M} \sum_{t=1}^{T} \frac{1}{2\sigma^2} (y(t) - e^T e(t))^2 - \sum_{m=1}^{M} \sum_{t=1}^{T} \log P(e_m(t) | e_m(t-1)).
\] (2.12)

Majorization-minimization (MM). A nice feature of the objective function in (2.12) is that the machines appear decoupled in the last term (\(e_m(t)\) does not interact with \(e_n(t)\) for \(m \neq n\)). However, the machines are coupled in the first term by the quadratic
\[ (y(t) - e^T e(t))^2 = (y(t) - (c_1^T e_1(t) + c_2^T e_2(t) + \cdots + c_M^T e_M(t)))^2. \]

This suggests the usage of a MM approach to solve (2.12). Generically, MM replaces a tricky objective function by a simpler, surrogate, upper-bound — known as the majorization function — which is tight at a given iterate; it then minimizes the upper-bound to find the next iterate, and the whole procedure is repeated till convergence (to a local minimizer). The MM approach is attractive if the majorization functions are simple to optimize. In our context, this will be achieved by majorizing the quadratic with another one that separates the machines.

To shorten notation, let \( e := (e_1, \ldots, e_M) \) and let \( f(e) \) denote the (whole) objective function in (2.12). Also, let \( F_\tau(e) \) denote the majorizing function corresponding to the iterate \( \tau \); that is, \( f(\tau) = F_\tau(\tau) \) and \( f(e) \leq F_\tau(e) \) for all \( e \). Let \( e^{(k)} \), \( k \geq 1 \), denote the \( k \)th iterate generated by the MM method; \( e^{(0)} \) represents the initialization. The pseudo-code is:

**Algorithm 1 MM method**

1: procedure MM\((e^{(0)}, \varepsilon)\)
2: \( e^{(1)} \leftarrow \arg\min_e F_{\varepsilon^{(0)}}(e) \)
3: \( k \leftarrow 1 \)
4: while \( \|e^{(k)} - e^{(k-1)}\| > \varepsilon \) do
5: \( e^{(k+1)} \leftarrow \arg\min_e F_{\varepsilon^{(k)}}(e) \)
6: \( k \leftarrow k + 1 \)
7: end while
8: return \( e^{(k)} \)
9: end procedure

The stopping threshold, \( \varepsilon \), was determined to be 5%, in other words, when the relative cost decrease of the surrogate function is below 5%, the algorithm stops.

**The 1st M: Majorizing function.** The majorizing function \( F_\tau(e) \) is derived in appendix A. It has the structure
\[
F_\tau(e) = \alpha + \beta^T e + \|e\|^2 \|e\|^2 - \sum_{m=1}^{M} \sum_{t=1}^{T} \log P(e_m(t) | e_m(t-1)).
\] (2.13)
where the scalar $\pi$ and the vector $\bar{b} = (\bar{b}_1^T \bar{b}_2^T \cdots \bar{b}_M^T)^T$ depend on $\pi$ but not on the variable $e$. The vector $c$ was introduced in (2.11).

Note that the machines are indeed decoupled in the majorizing function because the first term is

$$\pi + \bar{b}^T e + \|c\|^2 \|e\|^2 = \pi + \sum_{m=1}^{M} \left( \bar{b}_m^T e_m + \|c\|^2 \|e_m\|^2 \right).$$

The 2nd M: Minimization. The MM method does not provide an algorithm to minimize the surrogate function in (2.13). Note that the optimization variable $e = (e_1, \ldots, e_M)$ is discrete-valued, since $e_m = (e_m(1), \ldots, e_m(T))$ and each $e_m(t) = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{R}^{n_m}$ is an indicator vector. Thus, we face a combinatorial problem. However, the structure of the cost function allows for a dynamic programming-based approach known as the Viterbi algorithm [8].

The Viterbi algorithm works backwards in time, and processes separately each machine (because machines are decoupled in the surrogate function). Note that, in principle, the Viterbi algorithm could minimize directly the cost function (2.12) by interpreting the set of machines as just one (virtual) machine with an extended state space; however, this approach is computationally unfeasible due to the huge cardinality of the virtual state space $\prod_{m=1}^{M} n_m$.

Applied to a machine in (2.13), the Viterbi algorithm starts by computing the cost of activating a given state at $t = T$. Then, for $t = T - 1$, the algorithm computes the cost of activating a given state and finds the best transition to the next slot. This information is then stored, and the process is repeated until it reaches the start of the day $t = 1$. At this point, it knows the best starting state, and retrieves the best path forward, by using the information previously computed. The pseudo-code follows

**Algorithm 2 Viterbi algorithm**

```python
1: procedure VITERBI(y)
2:     for all Machines do
3:         Compute cost of $e$-th state on slot $T$
4:     for $t \leftarrow (T - 1), 1$ do
5:         for all States do
6:             Compute cost of $e$-th state on slot $t$
7:         for all states do
8:             Compute cost of transitions from $t$ to $t + 1$
9:         end for
10:     end for
11: end for
12: for all States do
13:     Compute cost of the starting state
14: end for
15: Save best path
16: end procedure
```

In conclusion, we minimize (2.12) by incorporating the Viterbi algorithm within the MM method. First, a starting point $e^{(0)}$ is chosen and the associated surrogate function $F_{e^{(0)}}$ is computed. Then, the Viterbi algorithm runs to find a minimizer — denoted $e^{(1)}$ — of the surrogate function $F_{e^{(0)}}$. The minimizer $e^{(1)}$ also drives down the true cost in (2.12). This procedure is repeated until...
convergence is detected. It is important to note that the MM algorithm is not guaranteed to converge to a global minimum. While there are several ways to choose the starting point, our first choice was the most probable path, in other words we choose the most probable state for \( t = 1 \) and then we choose the most probable transition to the next state, until we reach the final slot. Alternatively, we used the estimations of other methods (presented in sections 2.6 and 2.7), as initializations by choosing the closest state for each estimation. However, when unspecified, the most probable path is the chosen initialization.

**Markov chain construction: state space.** So far, the Markov chains have been assumed known. That is, for the \( m \)th machine, we assume that the number of states is known and also their realizations: recall the vector \( c_m \) in (2.7); furthermore, we have assumed that both the probability mass function \( \text{pmf} \) of \( e_m(1) \) and the transition matrices \( P(e_m(t) \mid e_m(t - 1)) \), \( t = 2, \ldots, T \), are known. Note that we are assuming non-homogeneous Markov chains; thus, the transition matrix \( P(e_m(t) \mid e_m(t - 1)) \) is (potentially) distinct from \( P(e_m(s) \mid e_m(s - 1)) \) for \( s \neq t \).

Now, we address how this data is inferred from the training dataset. We begin with the vector \( c_m \in \mathbb{R}^{n_m} \) in (2.7) which lists the state space of the \( m \)th machine. First, we need to decide the cardinality \( n_m \) of the state space (\( n_m \) is the number of states of the \( m \)th Markov chain). The training dataset only gives the consumption of the \( m \)th machine during the day. If each observed consumption level was converted to a state, \( n_m \) would be very large, and the Viterbi algorithm would not be computationally feasible. It is necessary to determine an adequate number of states and their respective energy consumptions, in order to compute a solution in reasonable time while still accurately modeling the machine. The number of states for each machine was set manually; for each \( n_m \), the \( K \)-means clustering method [9] was used to determine the \( n_m \) consumption levels. The number of clusters was chosen by finding an acceptable average squared error of the codebook delivered by the \( K \)-means algorithm. We ended up with a total of 70 states, distributed, for each machine as follows:

<table>
<thead>
<tr>
<th>Machine</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

**Markov chain construction: transition matrices.** For each machine \( m \), we estimate the initial state distribution and the transition matrices from the training data set via a maximum likelihood (ML) approach. In words, the ML estimate of the initial state is simply the relative observed frequency of the machines’ states: we count how many times a machine starts the day on each state, and the counting vector is normalized by the size of the training dataset. The ML estimation of the transition matrices is similar. We count the number of transitions across consecutive time slots and normalize:
if the machine jumps from state $i$ to state $j$ when passing from time slot $t$ to $t + 1$, then the $(i,j)$th entry of the transition matrix corresponding to slot $t$ is incremented. After counting every transition for every machine, the row of each of the transition matrices is normalized to have unit sum.

**Markov chain construction: variance of additive noise.** The only parameter left to be estimated is $\sigma^2$, the power of the additive Gaussian noise, recall (2.5). Since $y(t) \sim \mathcal{N}(e^T e(t), \sigma^2)$ (see (2.10)), the ML estimate of $\sigma^2$ is

$$\hat{\sigma}^2 = \frac{1}{TD} \sum_{d=1}^{D} \sum_{t=1}^{T} (y^{(d)}(t) - e^T e^{(d)}(t))^2,$$

where $y^{(d)}(t)$ and $e^{(d)}$, are the observed aggregate consumption and the machines’ states, respectively, in time slot $t$ of the $d$th day of the training dataset ($D$ is the total number of training days).

### 2.5 Coupled Markov chains

**Motivation.** In the previous section 2.4, we modeled each machine as an independent Markov chain. The resulting model is tractable, but it discards potential correlation between the machines. The correlation information may be useful to improve the estimates of the machines’ states, given the aggregate consumption. We now provide a toy scenario (somewhat exaggerated) to illustrate the point. Suppose $M = 2$ machines, denoted $X_1$ and $X_2$, and each one with state space $\{0, 1, 2\}$. Fix $\epsilon > 0$ very small, say $\epsilon = 0.001$. Assume the joint probability distribution $P$ is given by

<table>
<thead>
<tr>
<th>$X_1 \backslash X_2$</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 - $\epsilon/2$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\epsilon$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1 - $\epsilon/2$</td>
</tr>
</tbody>
</table>

In this idealized model, the machines are a copy of each other. If the machines are assumed independent (as in section 2.4), then each one is modeled by the (marginal) pmf $P(X_m = 0) = 1 - \epsilon/2$, $P(X_m = 1) = \epsilon$ and $P(X_m = 2) = 1 - \epsilon/2$.

Now, suppose we observe the aggregate consumption $Y = X_1 + X_2 = 2$. In the decoupled model of section 2.4 the ML estimate of the states is $(\hat{X}_1, \hat{X}_2) = (2, 0)$ or $(\hat{X}_1, \hat{X}_2) = (0, 2)$. However, neither of these ML estimates is possible given the joint pmf $P$ (it has zero probability of occurrence). Instead, if the correct model is adopted, the ML estimate is $(\hat{X}_1, \hat{X}_2) = (1, 1)$.

**Pairing machines.** As mentioned before, we could see the $M$ machines as a single virtual one with an extended state space (the Cartesian product of all the real machines); however, this state space would be too large to be computationally feasible. In this section, we introduce a model that sits between these two extremes: we form $M/2$ virtual machines by pairing the $M$ real machines. This permits to capture correlation between machines, and controls the complexity of the novel state spaces — the state space of each of the $M/2$ virtual machines is at most $n^2$, where $n = \max\{n_1, \ldots, n_M\}$.

---

1In this section, for simplicity, we assume that $M$ is even.
Optimal pairing. Before proceeding, we need to determine how the \( M/2 \) pairs are formed, i.e., which machine goes with which machine to form a single (virtual) machine. From a high level viewpoint, we face the following problem. We are given \( M \) time series (\( M \) is an even number) with finite length \( T \). The \( m \)th time series consists of \( T \) random variables

\[
X_m := (X_m(1), X_m(2), \ldots, X_m(T)),
\]

a realization of which is denoted

\[
x_m := (x_m(1), x_m(2), \ldots, x_m(T)).
\]

We assume that each random variable \( X_m(t) \) is discrete (with finite alphabet), for \( 1 \leq m \leq M \) and \( 1 \leq t \leq T \). Let \( P \) denote the joint pmf, that is, \( P(x_1, x_2, \ldots, x_M) \) stands for

\[
\text{Prob}(X_1(1) = x_1(1), X_1(2) = x_1(2), \ldots, X_M(T - 1) = x_M(T - 1), X_M(T) = x_M(t)).
\]

We are not assuming any independence property in the joint pmf \( P \). We want to approximate \( P \) by \( M/2 \) independent Markov chains obtained by pairing the \( M \) time series (each Markov chain approximates two \( M \) time series). We represent the pairing process by a permutation \( \tau : \{1, 2, \ldots, M\} \rightarrow \{1, 2, \ldots, M\} \); a permutation \( \tau \) represents the \( M/2 \) pairs \((\tau(i), \tau(i + 1))\) (we pair machine \( \tau(i) \) with machine \( \tau(i + 1) \)) for \( i = 1, 3, 5, \ldots, M - 1 \).

It is important to note that we want to find simultaneously the optimal pairing \( \tau \) and the parameters of the optimal Markov chains corresponding to the \( M/2 \) pairs created by \( \tau \). For example, if \( \tau(1) = 3 \) and \( \tau(2) = 7 \), then machine 3 is combined with machine 7, and we want to find the parameters of the best Markov chain that approximates the time series \( (X_3, X_7) \), i.e.,

\[
(X_3(1), X_7(1)) , (X_3(2), X_7(2)) , \ldots , (X_3(T - 1), X_7(T - 1)) , (X_3(T), X_7(T)).
\]

This means that we want to find the pmf of the initial state \((X_3(1), X_7(1))\) and each of the transition matrices corresponding to \((X_3(t), X_7(t)) \rightarrow (X_3(t + 1), X_7(t + 1))\) for \( t = 2, \ldots, T - 1 \); we denote the former by \( Q^{(3, 7)}(1) \) and the latter by \( Q^{(3, 7)}(t|t - 1) \).

Problem statement. We use Kullback-Leibler divergence \([10]\) to measure the dissimilarity between pmfs. In conclusion, we want to solve the optimization problem

\[
\min_{\tau, Q^\tau} D(P \mid\mid Q^\tau), \quad (2.14)
\]

where \( D(\cdot \mid\mid \cdot) \) denotes Kullback-Leibler (KL) divergence, \( P \) is the given joint pmf and \( Q^\tau \) is the pmf of the \( M/2 \) independent Markov chains created by the pairing \( \tau \). As \( Q^\tau \) factors into \( M/2 \) independent Markov chains, we adopt the notation

\[
Q^\tau = Q^{(\tau(1), \tau(2))} Q^{(\tau(3), \tau(4))} \ldots Q^{(\tau(M - 1), \tau(M))}, \quad (2.15)
\]

where \( Q^{(k, l)} \) is the pmf of the Markov chain that approximates the time series \((X_k, X_l)\).
For example, if $M = 4$ and $\tau(1) = 2$, $\tau(2) = 4$, $\tau(3) = 1$ and $\tau(4) = 3$ (we combine machine 2 with 4, and machine 1 with 3), then $Q^T(x_1, x_2, x_3, x_4) = Q^{(2,4)}(x_2, x_4)Q^{(1,3)}(x_1, x_3)$ where

$$Q^{(2,4)}(x_2, x_4) = q^{(2,4)}((x_2(1), x_4(1))) \prod_{t=2}^{T} Q_{t+1}^{(2,4)}((x_2(t), x_4(t)) | (x_2(t-1), x_4(t-1))). \quad (2.16)$$

A similar expression holds for $Q^{(1,3)}(x_1, x_3)$. For this fixed pairing $\tau$, problem (2.14) requires finding the optimal parameters $q^{(2,4)}$, $Q^{(2,4)}_{t+1}$, $q^{(1,3)}$ and $Q^{(1,3)}_{t+1}$ ($t = 2, \ldots, T$) — with the implicit constraint that $q^{(2,4)}$ and $q^{(1,3)}$ should be pmt's (nonnegative vectors summing to one), and $Q^{(2,4)}_{t+1}$ and $Q^{(1,3)}_{t+1}$ should be transition matrices (each row is a pmf). However, as stated above, problem (2.14) entails solving simultaneously for the pairing $\tau$ and the associated Markov chain parameters.

The goal of this section is to prove that (2.14) reduces to a classical matching problem on a certain graph. This reduction allows for efficient solution methods.

**Solving for $Q^T$: fixed pairing $\tau$.** Problem (2.14) has both discrete and continuous variables. The discrete variable is the permutation $\tau$. The continuous variables are the parameters of the Markov chains. We address problem (2.14) by optimizing out the continuous variables, i.e., by rewriting (2.14) in the nested form

$$\text{minimize } \left( \text{minimize } D(P \| Q^T) \right)$$

and solving the inner problem. In other words, we view $\tau$ as given, find the optimal parameters $Q^T$ by solving

$$\text{minimize } D(P \| Q^T), \quad (2.17)$$

and plug them back in the objective function in (2.14) to create a function depending only on $\tau$.

To solve (2.17), we start by writing its objective function. Using the definition of KL-divergence\footnote{For two pmfs $p = (p_i)_{i \in \mathbb{Z}}$ and $q = (q_i)_{i \in \mathbb{Z}}$, $D(p \| q) = \sum_{i \in \mathbb{Z}} p_i \log(p_i/q_i)$ with $0 \log(0/0) := 0.$} we have

$$D(P \| Q^T) = \sum_{x_1, \ldots, x_M} P(x_1, \ldots, x_M) \left( \log P(x_1, \ldots, x_M) - \log Q^T(x_1, \ldots, x_M) \right). \quad (2.18)$$

The first term does not depend on the variable $Q^T$ and can be dropped. Problem (2.17) reduces to

$$\text{maximize } Q^T, \quad \sum_{x_1, \ldots, x_M} P(x_1, \ldots, x_M) \log Q^T(x_1, \ldots, x_M). \quad (2.19)$$

Thanks to the factored form (2.15), there holds

$$\log Q^T(x_1, \ldots, x_M) = \sum_{i=1,3,\ldots,M-1} \log Q^{(\tau(i), \tau(i+1))}(x_{\tau(i)}, x_{\tau(i+1)}). \quad (2.20)$$

That is, $\log Q^T$ separates additively into components of the form $\log Q^{(k,l)}$ for $(k, l) = (\tau(i), \tau(i + 1))$. Furthermore, due to the fact that each $Q^{(k,l)}$ represents a Markov chain, $\log Q^{(k,l)}$ decouples over time (see also example (2.16)):

$$\log Q^{(k,l)}(x_k, x_l) = \log q^{(k,l)}((x_k(1), x_l(1))) + \sum_{t=2}^{T} \log Q_{t+1}^{(k,l)}((x_k(t), x_l(t)) | (x_k(t-1), x_l(t-1))). \quad (2.21)$$


Equations (2.20) and (2.21) show that (2.19) separates into problems of the form
\[
\max_{q^{(k,l)}} \sum_{x_k(1),x_l(1)} P(x_k(1),x_l(1)) \log q^{(k,l)}((x_k(1),x_l(1)))
\]
and
\[
\max_{Q^{(k,l)}_{t-1}} \sum_{x_k(t),x_l(t)} P(x_k(t),x_l(t)|x_k(t-1),x_l(t-1)) \log Q^{(k,l)}_{t-1}((x_k(t),x_l(t))| (x_k(t-1),x_l(t-1)))
\]
Both problems (2.22) and (2.23) allow closed form solutions. Problem (2.22) can be rewritten as
\[
\min_{q^{(k,l)}} \sum_{x_k(1),x_l(1)} P(x_k(1),x_l(1)) \left( \log P(x_k(1),x_l(1)) - \log q^{(k,l)}((x_k(1),x_l(1))) \right)
\]
whose objective is just the KL divergence between the pmf of \((X_k(1),X_l(1))\) — denoted \(p^{(k,l)}\) — and \(q^{(k,l)}\); clearly, the solution consists in taking
\[
q^{(k,l)} = p^{(k,l)}.
\]
A similar reasoning shows that the solution of (2.23) is
\[
Q^{(k,l)}_{t-1}((x_k(t),x_l(t))| (x_k(t-1),x_l(t-1))) = P^{(k,l)}_{t-1}((x_k(t-1),x_l(t-1))
\]
where \(P^{(k,l)}_{t-1}((x_k(t),x_l(t))| (x_k(t-1),x_l(t-1))) = P(X_k(t) = x_k(t), X_l(t) = x_l(t)| X_k(t-1) = x_k(t-1), X_l(t-1) = x_l(t-1)).\)

To conclude, the solution of (2.17) is the factored pmf
\[
Q^* = \prod_{i=1,3,...,M-1} Q^{(\tau(i),\tau(i+1))}
\]
where
\[
Q^{(\tau(i),\tau(i+1))} = p^{(\tau(i),\tau(i+1))} \prod_{t=2}^{T} P_{t-1}^{(\tau(i),\tau(i+1))}, \quad \text{for } i = 1, 3, \ldots, M - 1.
\]

**Solving for \(\tau\).** We now plug the solution (2.24) into (2.14) or, equivalently, (2.18).

We start by noting that for a given pair \((k, l) = (\tau(i), \tau(i + 1))\) there holds
\[
\sum_{x_1, \ldots, x_M} P(x_1, \ldots, x_M) \log Q^{(k,l)}(x_k, x_l) = \sum_{x_k, x_l} P(x_k, x_l) \log Q^{(k,l)}(x_k, x_l)
\]
by marginalizing out \(x_i\) for \(i \neq k, l\). The same reasoning (marginalizing out) and the factored form (2.25) gives
\[
\sum_{x_k, x_l} P(x_k, x_l) \log Q^{(k,l)}(x_k, x_l) = \sum_{x_k(1),x_l(1)} P(x_k(1),x_l(1)) \log p^{(k,l)}(x_k(1), x_l(1)) +
\]
\[
\sum_{t=2}^{T} \sum_{x_k(t-1),x_l(t-1),x_k(t),x_l(t)} P(x_k(t-1), x_k(t), x_l(t-1), x_l(t)) \log p^{(k,l)}(x_k(t), x_l(t)| x_k(t-1), x_l(t-1)).
\]

The first term in (2.26) is
\[
\sum_{x_k(1),x_l(1)} P(x_k(1),x_l(1)) \log p^{(k,l)}(x_k(1), x_l(1)) = \sum_{x_k(1),x_l(1)} P(x_k(1),x_l(1)) \log P(x_k(1), x_l(1))
\]
and is the negative entropy\(^3\) of the pair \((X_k(1), X_l(1))\), i.e., \(-H(X_k(1), X_l(1))\). The second term in (2.26) is
\[
\sum_{x_k(t-1), x_l(t-1)} P(x_k(t-1), x_l(t)) \log P_{t-1}^{(k,l)}(x_k(t-1), x_l(t))
\]
where
\[
\sum_{x_k(t-1)} P(x_k(t-1)) \left[ \sum_{x_l(t)} P(x_l(t) | x_k(t-1), x_l(t)) \log P_{t-1}^{(k,l)}(x_k(t), x_l(t) | x_k(t-1), x_l(t)) \right] 
\]
\[
= \sum_{x_k(t-1), x_l(t-1)} P(x_k(t-1), x_l(t-1)) \left[ \sum_{x_k(t)} P(x_k(t) | x_k(t-1), x_l(t)) \log P_{t-1}^{(k,l)}(x_k(t), x_l(t) | x_k(t-1), x_l(t)) \right] 
\]
and is the negative conditional entropy of \(((X_k(t-1), X_l(t-1)), (X_k(t), X_l(t)))\), i.e., \(-H(X_k(t), X_l(t) | X_k(t-1), X_l(t-1))\).

Collecting our results, we conclude that
\[
\sum_{x_k, x_l} P(x_k, x_l) \log Q^{(k,l)}(x_k, x_l) = -\mathcal{H}(X_k, X_l) \tag{2.28} 
\]
where
\[
\mathcal{H}(X_k, X_l) = H(X_k(1), X_l(1)) + \sum_{t=2}^{T} H(X_k(t), X_l(t) | X_k(t-1), X_l(t-1)) 
\]
could be interpreted as the “Markov entropy” of the time series
\[
(X_k, X_l) = ((X_k(1), X_l(1)), (X_k(2), X_l(2)), \ldots, (X_k(T), X_l(T))). 
\]

Finally, inserting (2.28) into (2.18) shows that, for the optimal \(Q^*\), we have
\[
D(P || Q^*) = \sum_{i=13, \ldots, M-1} \mathcal{H}(X_{T(i)}, X_{T(i+1)}) - H(P). 
\]

This means that (2.14) boils down to solving
\[
\text{minimize } \sum_{i=13, \ldots, M-1} \mathcal{H}(X_{T(i)}, X_{T(i+1)}). \tag{2.29} 
\]

Problem (2.29) can be interpreted as a classical matching problems on graphs. We create a graph with \(M\) nodes (the \(m\)th node represents the time series \(X_m\)), and we place an edge with weight \(\mathcal{H}(X_k, X_l)\) between each pair of nodes \(k\) and \(l\). In this graph, (2.29) is asking for a matching of the nodes (i.e., we must marry each one with another one) with minimal weight. There are efficient algorithms for solving matching problems [11]. Problem (2.29) has an intuitive explanation: in order to minimize the total entropy of the final \(M/2\) machines, we must form pairs with low entropy, i.e., join highly correlated machines.

In our setup, we have a relatively small \(M\): \(M = 10\). We solved problem (2.29) by exhaustive search, i.e., we computed the “Markov” entropy \(\mathcal{H}(X_k, X_l)\) for each pair \((k, l)\) of machines (based on the data on the training set), then we generated all possible combinations of the five pairs, and picked the one with smallest objective value.

\(^3\)The entropy of a pmf \(p = (p_1)_{i \in Z}\) is \(H(p) = -\sum_{i \in Z} p_i \log p_i\) with \(0 \log 0 := 0\).
Online inference. After the pairing is done, we find ourselves back in the framework of section 2.4 — instead of $M$ Markov chains, we have now $M/2$ (where before there were two machines, one with $n_k$ and the other $n_l$ states, now there is a single virtual machine with $n_k n_l$ states). The consumption of each state of this new machine is simply the sum of the consumptions of the machines in the pair. Everything else remains the same.

2.6 Gaussian model

Probabilistic model. In this section, we propose a model that allows dependence between machines within a time block $\{t, t+1, \ldots, t+\tau-1\}$ where $\tau \geq 1$ is the block size; but non-overlapping blocks are assumed independent. More precisely, let

$$x(t) = (x_1(t), x_2(t), \ldots, x_M(t)) \in \mathbb{R}^M,$$

denote the vector of machine consumptions’ in slot $t$, and let

$$x_\tau(t) = (x(t)^T, x(t+1)^T, \ldots, x(t+\tau-1)^T)^T \in \mathbb{R}^{\tau M}$$

contain all consumptions in the interval $\{t, t+1, \ldots, t+\tau-1\}$. We assume that $x_\tau(t)$ is Gaussian distributed with mean $\mu(t)$ and covariance matrix $\Sigma(t)$: $x(t) \sim \mathcal{N}(\mu(t), \Sigma(t))$. But, we assume that $x_\tau(t)$ is independent from $x_\tau(s)$ for $|t-s| \geq \tau$. Figure 2.2 illustrates the corresponding probabilistic graphical model for $\tau = 1$.

![Figure 2.2: Probabilistic graphical model for section 2.6: machines are jointly Gaussian within time blocks of size $\tau$ (here, $\tau = 1$)](image)

Model flexibility. This model permits to explore both types of correlation — time and spatial (i.e., between machines) — by controlling the block size $\tau$ and the structure of the correlation matrix $\Sigma(t)$.
The reason that we have to restrict $\tau$ to small values, and $\Sigma(t)$ to certain patterns, is the size of the training data set: if we take $\tau$ large and $\Sigma(t)$ dense (to capture a lot of time and space correlation), there would be too many parameters to be reliably estimated from the limited training dataset available.

In this thesis, we examined two approaches. In the first approach, we assumed no time correlation between variables while enabling dependence between machines. That is, we took a time block of size $\tau = 1$ and allowed each $\Sigma(t)$ to be fully dense (equivalently, we did not impose any independence between machines in a given time slot). In the second approach, we admitted time correlation while assuming the machines work in groups, similarly to the model in section 2.5, however instead of grouping the machines in pairs, we grouped them in sets of three, leaving one machine uncorrelated. That is, we choose $\tau > 1$ (we considered $\tau = 3$) and let $\Sigma(t)$ be block diagonal (after a suitable permutation) to express independence between groups of machines.

However, other possibilities can be explored. Independently of the structure in $\Sigma(t)$ (and $\mu(t)$) we derive below a unified method for inference of the machine consumptions in a time block from the observed aggregate composition.

**Maximum a posteriori (MAP) inference.** Let

$$y_{\tau}(t) = (y(t), y(t+1), \ldots, y(t+\tau-1)) \in \mathbb{R}^\tau$$

be the observed aggregate consumption in a given time block. Our goal is to infer $x_{\tau}(t)$ in (2.30) from the measurement $y_{\tau}(t)$. Note that

$$y_{\tau}(t) = Ax_{\tau}(t)$$

where $A \in \mathbb{R}^{\tau \times \tau M}$ is a suitable matrix. For example, for $M = 5$ and $\tau = 3$,

$$A = \begin{bmatrix} e^T & 0 & 0 \\ 0 & e^T & 0 \\ 0 & 0 & e^T \end{bmatrix}$$

and $e = (1, 1, 1, 1, 1)$.

We follow a MAP approach:

$$\hat{x}_{\tau}(t) = \arg\max_{x_{\tau}(t)} P(x_{\tau}(t) \mid y_{\tau}(t)). \quad (2.31)$$

Given our statistical assumptions, (2.31) is equivalent to

$$\minimize_{x_{\tau}(t)} \frac{(x_{\tau}(t) - \mu(t))^T \Sigma(t)^{-1} (x_{\tau}(t) - \mu(t))}{Ax_{\tau}(t) = y_{\tau}(t)}. \quad (2.32)$$

Problem (2.32) is a simple quadratic problem with linear constrains whose solution is well-known [12]:

$$x_{\tau}(t) = \mu(t) + \Sigma(t) A^T (A \Sigma(t) A^T)^{-1} (y_{\tau}(t) - A\mu(t)). \quad (2.33)$$
Estimation of the model parameters. The model parameters $\mu(t)$ and $\Sigma(t)$ can be obtained from the training dataset by ML estimation:

$$\hat{\mu}(t) = \frac{1}{D} \sum_{d=1}^{D} x^{(d)}(t), \quad \hat{\Sigma}(t) = \frac{1}{D} \sum_{d=1}^{D} \left( x^{(d)}(t) - \hat{\mu}(t) \right) \left( x^{(d)}(t) - \hat{\mu}(t) \right)^T,$$

where $x^{(d)}(t)$ denotes the observed $x(\tau)$ in the $d$th day of the training dataset.

2.7 Minimum Mean-Squared Estimation (MMSE)

We recall the concept of Minimum Mean-Squared Estimation (MMSE), see [13] for extra details. Let $(X,Y)$ be two random variables with joint pmf $P(X,Y)$. We observe $Y = y$ and want to infer $X$. The MMSE estimate corresponds to

$$\hat{X} = E\{X \mid Y = y\} = \sum_{x} x P(x \mid Y = y) = \sum_{x} x \frac{P(x,y)}{\sum_{x'} P(x',y)} \quad (2.34)$$

and can be shown to be the optimal estimate under the squared error criterion, i.e., $\hat{X}$ achieves the minimum of $E\{(X - f(Y))^2\}$ for any function $f$. In our case, $X$ denotes a given machine consumption $X = x_m(t)$ and $Y$ denotes an aggregate consumption or a set of aggregate consumptions $Y = (y(t-\tau+1), \ldots, y(t-1), y(t))$ for some $\tau \geq 1$.

Implementation. In order to apply the MMSE estimate, we need the joint pmf $P(X,Y)$, which must be obtained from the training dataset. There are several issues regarding the acquisition of $P(X,Y)$. The main one relates to the granularity of the discretization (we must construct a finite alphabet for $X$ and $Y$ from the training dataset). A higher granularity is a more accurate representation of the continuous consumption but raises other problems: the higher the granularity, the larger the amount of space needed to store the joint pmf, the heavier the computation in (2.34) becomes, and the risk of encountering a consumption $Y'$ in real-time whose bin that has not been observed in the training dataset. This granularity was adjusted manually. Initially this granularity was small, however higher granularity meant better results and so it was increased until an increase in the granularity no longer produced better results. In the end, the number of selected bins for the observations, $Y$, was 75, and for the consumptions, $X$, was 1000. It is important to note, however, that this granularity was not used for the case when we are able to observe a machine, since this lead to memory problems. This happens because we introduced an extra dimension to the joint pmf and should the granularity remain the same, we would have a variable with $1000 \times 1000 \times 75$ entries. To solve this problem, a lower granularity was used for the machine’s consumptions, specifically 100, while the granularity for the observations stayed the same. The results obtained from observing one machine, even though we used a lower granularity, will still be compared with the results obtained from observing none of the machines, using a higher granularity, in order to ascertain if the compromise between observing an extra machine and lowering the granularity, due to memory constraints, to do so is worth it. We constructed manually the alphabets of $X$ and $Y$ in order to address these concerns. In particular, whenever we run into an (online) consumption $Y$ which not been seen in the training dataset (making the denominator in (2.34) zero), we replace it by a valid one in the neighborhood of $Y$. 
2.8 Methods with side information

The suite of methods — developed in the previous sections — is a tool that enables addressing several questions of practical interest. For example, if one could measure one or more machines directly (by installing dedicated sensors) which machines should we select? We are asking which are the most informative machines under a certain probabilistic model, i.e., the machines whose observation provide valuable information to guess the other machines’ states. To answer this question, we must incorporate this side information (some machines’ consumptions, in addition to the aggregate consumption) into our previous methods. This is the purpose of this chapter (after the incorporation, the answer is readily available: it suffices to run the novel methods for the several possible combinations of observed machines, and pick the best option).

In the sequel, we present the required changes in some of the previous models to accommodate the side information.

Decoupled Markov Chains. The Decoupled Markov Chains method requires few changes in order to incorporate the observation of a machine. Since it assumes that all machines are independent entities, observing one machine only influences the observation of the total consumption. Indeed, assume we observe the consumption of the \(m\)th machine, \(x_m(t)\). Given that

\[
y(t) = x_1(t) + \cdots + x_m(t) + \cdots + x_M(t)
\]

it follows that

\[
y(t) - x_m(t) = \sum_{k \neq m} x_k(t)
\]

and the left-hand side of the equality above can be interpreted as the available measurement. Thus, observing a machine is equivalent of removing it from the problem: the MM approach described in the previous chapter runs without any modification, instead of \(M\) machines we have \(M - 1\) machines.

This principle generalizes in an obvious manner to the case in which we observe a subset of machines (not a single one).

Gaussian models. Similarly to the previous case, the Gaussian models method requires little change in order to incorporate the observation of a machine. For simplicity, we consider unit block sizes, i.e., \(\tau = 1\). Suppose the \(m\)th machine is observed, \(x_m(t)\). Problem (2.32) now becomes

\[
\begin{align*}
\text{minimize} & \quad (x(t) - \mu(t))^T \Sigma(t)^{-1} (x(t) - \mu(t)) \\
\text{subject to} & \quad e^T x(t) = y(t) \\
& \quad f^T x(t) = x_m(t),
\end{align*}
\]

(2.35)

where \(e = (1, 1, \ldots, 1)\) and \(f = (0, \ldots, 0, 1, 0, \ldots, 0)\); the vector \(f\) is nonzero in the \(m\)th entry.

The first constraint in (2.35) accounts for the aggregate consumption measurement \(y(t)\), while the second constraint incorporates the novel measurement \(x_m(t)\).

Problem (2.35) is still a quadratic problem with linear constraints, thus obeying the generic structure of (2.32), just with a different \(A\). The solution (2.33) remains applicable with the new \(A\) plugged in.
Finally, the observation of a group of machines can be handled in a similar manner (again, only the matrix $A$ needs to be changed to accommodate the additional observations).

**MMSE.** In this method, the information regarding the observation of one machine must be incorporated into the joint PMF. This means, unlike the previous methods, that it is necessary to estimate new parameters. Before we had $(X, Y)$, now we have $(X, Y, Z)$ where $Z$ represents the novel observation (a machine state, $x_m(t)$). As such, the MMSE solution in (2.34) evolves to

$$\hat{X} = E\{X \mid Y = y, Z = z\} = \sum_x x P(x \mid Y = y, Z = z) = \sum_x \sum_{x'} \frac{P(x, y, z)}{P(x', y, z)}$$  \hspace{1cm} (2.36)

This means that we need to revisit the training dataset to estimate the novel (augmented) joint PMF $P(X, Y, Z)$. The expansion of the joint PMF will cause an increase in the time needed to compute the solution, and extra memory space to store it.
3.1 Introduction

In this chapter we present the experimental results obtained from the implementation of the methods described in the previous chapter. We will do this in three parts.

Part 1: Synthetic data. The first will focus on the results of each method using two synthetic datasets, generated based on the estimated parameters of three data models previously described; in other words, a dataset was generated from the estimated parameters of the Markov chains, from now on referred to as the Markov dataset and the second dataset was generated based on the estimated parameters of the normal distribution referred to as the Gaussian dataset.

Part 2: Real data. The second will focus on analyzing real data. While this dataset identifies the consumptions of ten groups of machines, one of this groups was not measured directly, with it's consumptions being the difference between the measured aggregate consumption and the sum of the measured consumptions of the remaining nine machines. Unlike the other machines this one is not composed of identical equipments. The real dataset was separated in two parts, one used for estimation of the parameters (training dataset) and the other used for evaluation of the methods (validation dataset). We randomly split the data 80% for training purposes and the remainder 20% for evaluation purposes. In the end we used 100 days worth of data to estimate the parameters of each model, and 25 days to evaluate the performance of the methods. The synthetic datasets also share the size of the evaluation set, i.e., 25 days. It is important to note that the days were kept intact, meaning that we split the days at the end of each one, and then either selected the whole day for training or testing. In this part we will also analyze the results of different initializations of the Markov Chains methods.

Part 3: Incorporation of side information. Finally, we will briefly analyze the results of incorporating side information into some of our methods. This will be done by analyzing the absolute average errors of the estimations.

Performance metrics. We will compare each method by utilizing several different evaluation metrics. The first and, arguably, the most simple metric is a comparison of the average absolute error of each method for each machine. If we add the average consumption of each machine to the metric this allows us to ascertain how well each method is, on average, estimating the realization of every machine. The second metric is a comparison, for each machine, of the histograms of the relative error of each method. These histograms will allow us to compare the accuracy of each method, for a given machine. The third metric will be based on the use of confusion matrices. For this purpose the real consumptions of each machine, as well as their estimations, will be divided into bins. This will allow us to visualize the accuracy of each method in a different light. Finally, we will compare the estimated consumptions for all methods throughout random days. This will allow to see how well a method can capture the overall behavior of a machine throughout the day.
**Metrics: Illustrative examples** We can visualize an example of the first metric in Figure 3.1. This bar graph shows us the average absolute errors for each method, across the machines, along with each machines’ average consumption. The vertical axis represents the consumption, in Killowatt hour (kWh), on the horizontal axis we represent each machine, labeled by their numbers. The left-most bar on each group represents the average consumption of the corresponding machine, while the remaining bars represent the average absolute error for the various methods. The average consumption of each machine is added in order to provide some sense of dimension for each machine. As we can see in the example of Figure 3.1 the eighth machine is, on average, the biggest contributor for the aggregate consumption while the ninth machine is on average, the smallest contributor. We can also conclude that, for the three biggest contributors (machines 6, 7 and 8), on average, method 3 presents the smallest average error. The smaller the average error of a method, the better, however this does not tell the whole story.

![Figure 3.1: Metric 1: average absolute error](image)

**Figure 3.1: Metric 1: average absolute error**

Figure 3.2 represents an example of the second metric. Here the vertical axis represents the amount of outputs while the horizontal axis displays the relative error. Both these quantities are in percentage, meaning we are able to visualize the amount of outputs that were estimated with an error below any given mark.

As we can see all three represented methods are competitive, in the sense that the amount of estimations made, with a relative error of ten, or less, percent is high. For both methods 1 and 3, represented in black and blue, respectively, that amount is around 75%; this means that those meth-
ods achieve a relative accuracy less or equal than 10% in 75% of their outputs. However for method 2, represented in red that percentage is about 10% higher, making it, for this particular machine, the best choice.

In Figure 3.3 we show an example of a confusion matrix. On the vertical axis we represent the binned estimates, while on the horizontal axis we represent the binned true consumptions. This classes are represented uniformly across the range of consumptions of each machine, i.e, size (in kWh) of each bin is the same. The size between each bin is

$$\frac{\max(x_m) - \min(x_m)}{n},$$

where $n$ is the number of classes of the $m$th machine, and $\max(x_m)$ and $\min(x_m)$ are the maximum and the minimum consumptions, respectively.

Correct guesses are located on the diagonal, guesses that fall below the diagonal are result of the methods underestimating the consumption, while guesses above the diagonal are overestimations of the consumption. The bar on the right shows a direct translation between the color of a guess and its frequency: the hotter the color, the more frequent the pair of bins. We can see that the represented method's estimation, for this particular machine, is more often than not, the sixth bin, making every bin before it frequently overestimated, and every bin after frequently underestimated. For these last two metrics, given the amount of figures that can be generated, only some cases will be shown and analyzed.
An estimation snapshot. Finally, Figure 3.4 shows, not only the real consumption of a machine but several estimations for it in that day. The vertical axis represents a consumption, again (in kWh) while the horizontal axis represents time. The real consumption of the machine for that day is represented by the black circles, while the colored lines represent each a different method.

As we can see, all three methods seem to capture, somewhat, the general behavior of that partic-
ular machine, in that day. While method 1 (red) estimation seems rather oscillatory both methods 2 and 3 (blue and green) were able to capture the steady behavior. However, between time slot number 20 and time slot number 25, method 1 somehow captured that midlevel of consumption, while the other two simply rose to the peak and remained there.
3.2 Synthetic Datasets

Average Absolute Errors. We begin analyzing the results by comparing the performances of each method for each of the two synthetic datasets (Markov and Gaussian, recall section 3.1). In Figure 3.5 we can observe the average error of three methods, both heuristics and the MMSE estimation. We can see that none of the three methods presented is consistently better than the other two and across every machine the performance of these three methods is very similar for both datasets.

![Figure 3.5: Absolute average errors: Heuristic and MMSE methods](image)

Regarding the Markov Chains estimation methods, Figure 3.6 shows a rather consistent average error across all datasets, for almost every machine, in the sense that both methods have very similar average absolute errors in both datasets when comparing the results of a single machine. While both methods present similar average errors for almost every machine, in some machines one of the methods manages to produce an inferior average error. This is the case of machine 6, where the Decoupled method presents an inferior error for both the Markov dataset and the Gaussian dataset. In machine 8, this method also presents an inferior, although only slightly, average error, for both datasets. In both machine 9 and 10 the Coupled method seems to perform slightly better, presenting an inferior average error for both datasets.

The Gaussian model methods, as can be seen in Figure 3.7, show a rather similar average error across all machines and datasets with no discernible different between the average errors across all
Figure 3.6: Absolute average errors: Markov Chains methods

In Figure 3.8 we can observe the average error for the Markov dataset for all seven methods. None of the methods seems to produce a consistently smaller average error. Similarly, no method's estimations have a consistently larger average error than the others. However, machines with a smaller average consumption seem to be estimated with an average error similar to this consumption, while machines with bigger average consumption are estimated with a considerably smaller average error.

Figure 3.9 displays the average absolute errors for the Gaussian dataset, for all seven methods. Like in the previous Figure, no method seems consistently better or worse than the rest, and machines with a larger average contribution seem to be estimated with a smaller average error, when compared to this average contribution, while machines with smaller average contributions are estimated with average errors.
Figure 3.7: Absolute average errors: Gaussian model methods
Figure 3.8: Absolute average errors: Markov dataset
Figure 3.9: Absolute average errors: Gaussian dataset
Histograms of the relative errors and confusion matrices. We'll now analyze the histograms of the relative errors, beginning with the Markov dataset. In Figure 3.10 we display this histogram for machine 6. We can see that, for the Decoupled Markov Chains method and for the MMSE method, that 90% of the data is estimated with a relative error below 10%, meaning that these methods, for this machine, in this dataset. Both Gaussian model methods and the Coupled Markov chains method estimate around 75% of the data with an error below 10%, while both heuristics seem to have a rather poor performance. Before we proceed let us observe the confusion matrices for some of these methods. The confusion matrix for the Decoupled Markov Chains method, displayed in Figure 3.11, further shows the accuracy of the method for this machine. It also shows that this method often mistakes the two most common bins, and that this machine operates mostly in those two bins. What is not visible in this matrix, given the color coding and the frequency of these events, is that every other class is misclassified, with the exception of the $5^{th}$ class, and even this one is more likely to be misclassified than correctly estimated. Looking at the confusion matrix of the IPF method, displayed in Figure 3.12 we can see that, even though this method is one of the most inaccurate, it still manages to "classify" most of the bins correctly. However, in contrast with the previous method, this method's estimations seem to "bleed" into the adjacent bins.

![Figure 3.10: Histogram of the relative errors for Machine 6: Markov dataset](image)

Figure 3.13 shows the histogram for machine 8. None of the methods' performance stands out, with between 35% and 55% of every method's estimations having a relative error below 10%. Both Gaussian model methods behave similarly, likewise both Markov Chains methods also have a similarly performance. For estimations with an error below 5% the Markov Chain Methods are the most accurate option, however, if we look at the amount of data with errors below the 10%, or higher, mark the Gaussian model methods become the most accurate. The confusion matrices for this particular
Figure 3.11: Confusion matrix of the Decoupled Markov Chains method for machine 6: Markov dataset

machine are all similar. An example is displayed in Figure 3.14, corresponding to the MMSE method, where we can see that the machine operates mostly on the lower classes and that while the bulk of the estimations are centered around the diagonal, misclassifications are common.

We can visualize the histogram of the relative errors for machine 5 in Figure 3.15. None of the methods preforms well in this machine. Both the Markov Chains methods perform better than the other five methods, however, the best method for this machine, the Coupled Markov Chains method, only estimated about 38% of the data with an error below the 10% mark, while the Decoupled Markov Chains comes close at 36% of the data estimated below this mark. In the confusion matrix for the Coupled Markov Chains method, displayed in Figure 3.16, we can clearly observe one of the limitations of the Markov Chains methods. While the machine operates in every bin, these particular methods are only capable of estimating a limited number of bins, defined by the number of states of each machine, in this case the even bins. The rest of the methods estimate only about 10% or less of the data with an error below the 10% mark, the exception being the MPF method. An example of one of the confusion matrices for this methods is displayed in Figure 3.17. We can see that this method's estimations are focused primarily on three of the bins, namely 4, 5 and 6, however these estimations are not accurate since this machine seems to operate in all bins with the same frequency, with the exception to this being the second bin, which is more frequent than the rest.

We proceed to the histograms of relative errors of the Gaussian dataset. In Figure 3.18 we observe, again, machine 6. We can see that, with the exception of the heuristic methods, all methods manage to estimate over 60% of the data with an error below the 10% mark, while both Gaussian model methods are able to estimate over 70% of the data below this mark. The confusion matrices for this machine in this dataset are similar to the ones of the previous dataset.

In Figure 3.19 we can see the performance of the methods for machine 7. In this machine the IPF method is the most accurate, with over 45% of the data estimated with an error inferior to 10%. All the
other methods seem to behave rather similarly, estimating about 35% of the data with an error below the 10% mark, with the Gaussian Processes method falling somewhere in the middle between the IPF method, and the remaining methods. If we take a look at the confusion matrix of the Gaussian model block size 1 for this machine, displayed in Figure 3.20, we can see that this machine operates mostly in the first and last bin of consumptions, and that most of this method’s mistakes are for these same bins, although bin 9 is also commonly misclassified.

For this dataset, the machine estimated most inaccurately is, like in the previous dataset, machine 5. The performance of the seven methods can be seen in Figure 3.21. No method manages to estimate more than 20% of the data with an error inferior to 10%. However, the IPF method’s estimation are more accurate than the rest. In this dataset this is common, and, for some machines, this method is capable accurately estimating the consumptions of a machine, as we can see, for example in Figure 3.22, where the IPF is clearly the most accurate method for that machine, in this dataset. Taking a look at one of the confusion matrices for this machine (MPF method), displayed in Figure 3.23, we can see that this machine mostly operates on the first bin, and that the bulk of the method’s estimations are also on the same bin. Despite the histogram of the relative errors suggesting that the methods are inaccurate, if we look at the confusion matrices, in this case for the MPF method, we can see otherwise. This happens because this particular machine often has a very low consumption (which puts most of the machine’s operation in the lowest bin), however the maximum observed consumption is around 25 times higher than the average consumption, which is the limit of the highest bin.
Figure 3.13: Histogram of the relative errors for machine 8: Markov dataset

Figure 3.14: Confusion matrix of the MMSE method for machine 8: Markov dataset
Figure 3.15: Histogram of the relative errors for machine 5: Markov dataset

Figure 3.16: Confusion matrix of the Coupled Markov Chains method for machine 5: Markov dataset
Figure 3.17: Confusion matrix of the Gaussian model (Block size 3) method for machine 5: Markov dataset

Figure 3.18: Histogram of the relative errors for machine 6: Gaussian dataset
**Figure 3.19:** Histogram of the relative errors for machine 7: Gaussian dataset

**Figure 3.20:** Confusion matrix of the Gaussian model (block size 1) method for machine 7: Gaussian dataset
Figure 3.21: Histogram of the relative errors for machine 5: Gaussian dataset

Figure 3.22: Histogram of the relative errors for machine 9: Gaussian dataset
Figure 3.23: Confusion matrix of the MPF method for machine 9: Gaussian dataset
**Machine Snapshots.** We now proceed to illustrate the results, throughout a random day, of each method for some of the machines we have observed so far. We begin with machine 6, the one whose estimations are more accurate (in both synthetic datasets). In Figures 3.24 and 3.25 we can see the consumption of machine 6 generated, according to the previously estimated Markov chain, along each time slot, and the corresponding estimations. We can see that each method manages to capture the behavior of this machine. The heuristic method's estimations (red and black) are not very steady because they depend on the aggregate consumption and the training of the parameters of the methods was done using real data, not data generated according to this particular model. The Coupled Markov Chains method (green) fails to capture the decline in consumption, however it is important to remember that this method's estimations are influence by the behavior of another machine.

![Figure 3.24: Snapshot of machine 6 in a random day: Markov Chains and Gaussian models - Markov dataset](image)

Figures 3.26 and 3.27 illustrate the same machine but in this example the data were generated according to the Gaussian model. One immediate difference is that the consumptions of this machine can be negative. While this is not possible in reality, we decided to leave this consumptions negative intentionally, instead of forcing them to be zero. We can see, like in the previous case of Figure 3.24, that all methods capture the overall behavior of this machine. The Coupled Markov Chains method (green) still fails to capture the decline in consumption at the end of the day.

We now illustrate a snapshot of a different machine (namely 4), in a random day. In Figures 3.28 and 3.29 we can see that the estimations are extremely inaccurate. None of the methods are capable...
of capturing the machine's behavior with the exception of Markov chains methods (blue and green), that somewhat capture the machine's behavior around the middle of the day, partially because the dataset we are observing is the Markov dataset.

We conclude the results for the synthetic datasets. It is important to note that while we have not displayed the results of every machine in both datasets, these results illustrate the overall performance of the methods. For example, in Figures 3.10, 3.13 and 3.15, which show the performance of the methods for three of the ten machines for the Markov dataset, we can see that the Markov Chains methods are among the most accurate. This also is also true for the machines whose results we do not analyze. The same is true for the Gaussian dataset. The Gaussian model methods rank among the most accurate for every machine in the Gaussian dataset, regardless of the overall accuracy of the methods for that machine.
Figure 3.26: Snapshot of machine 6 in a random day: Markov Chains and Gaussian models - Gaussian dataset

Figure 3.27: Snapshot of machine 6 in a random day: MMSE and Heuristics - Gaussian dataset
**Figure 3.28:** Snapshot of machine 4 in a random day: Markov Chains and Gaussian models - Markov dataset

**Figure 3.29:** Snapshot of machine 4 in a random day: MMSE and Heuristics - Markov dataset
3.3 Real Data

**Average absolute errors.** We begin now analyzing the results of each method for the dataset containing real data. Starting with the average absolute errors, shown in Figure 3.30, we can see that, like in the synthetic datasets, that the machines with a higher average contribution to the aggregate consumption are estimated with smaller average absolute errors, when compared to this average contribution. We can also conclude that no method is consistently better. The method with lowest average absolute error varies from machine to machine, however the Gaussian Processes’ estimations frequently have some of the lowest average errors. Likewise, the method with the highest average error also varies from machine to machine, with Decoupled Markov Chains methods often being the worst method, in regards to this method.

![Figure 3.30: Real data: Average absolute errors](image)

**Histograms of the relative errors and confusion matrices.** In Figure 3.31, we can observe the histogram of the relative errors for machine 6, the best case. For this machine every method manages to estimate, at least, 50% of the data with an error below the 10% mark. The most accurate methods are the Markov Chains methods, the **MMSE** method and the Gaussian model (block size 1), with
an accuracy of at least 10% for more than 85% of the data, with both Heuristic methods being the most inaccurate. We illustrate an example of a confusion matrix for this machine in Figure ?? . This matrix corresponds to the Gaussian model (block size 1) method. This represents well the overall accuracy of the methods, and as we can see, this machine primarily operates on the first and last bin of consumptions, which is where most of the methods’ estimations fall. Most of this method’s mistakes consist in overestimating the first bin and underestimating the last bin. This can also be observed for the remaining methods with the exception of the Markov Chains methods, who seem to “confuse” both bins, instead of overestimating (first bin), or underestimating (last bin) them.

![Image of a confusion matrix]

Figure 3.31: Histogram of the Relative errors for machine 6: real data

The accuracy of the methods for machine 1 are shown in Figure 3.33. For this machines none of the methods managed to be more than 10% accurate for more than 60% of the data. However, the most accurate methods, for this mark, are, again, the Markov Chains methods, with the IPF method and the Gaussian model (block size 1) method having a similar accuracy. The worst method for this machine is clearly the Gaussian model (block size 3), with an accuracy of 10% for no more than 40% of the data. Figure ?? illustrates the confusion matrix of the MMSE method for machine 1. We can see this method manages to frequently estimate the first bin of consumptions accurately, however bins of higher consumptions are often underestimated.

The most difficult machine to estimate is machine 5, like in the synthetic datasets. For this machine the most accurate method is only capable of estimating no more than 25% of the data with an error below the 10% mark, while most of the remaining methods are only capable of an accuracy of 10% of no more than 10% of the data. The exception to this being the MPF method, estimating 15% of the data with an accuracy of 10%. Figure 3.36 illustrates one the confusion matrices (MPF method) for
Figure 3.32: Confusion matrix of the Gaussian model (block size 1) method for machine 6: real data

this machine. We can see that while the second bin of consumptions is often accurately estimated, the remaining, for this method, are not, being frequently underestimated. It is also visible that while the machine consumptions primarily fall on the second bin, so do the estimations of the MPF method. Only both the Markov Chains methods are also capable of capturing this behavior, however their estimations fall on a limited number bins, whereas the other methods’ estimations fall on a wider range of bins.
Figure 3.33: Histogram of the Relative errors for machine 1: real data

Figure 3.34: Confusion matrix of the MMSE method for machine 1: real data
Figure 3.35: Histogram of the Relative errors for machine 5: real data

Figure 3.36: Confusion matrix of the MPF method for machine 5: real data
Some snapshots of a few machines. Figures 3.37 and 3.38 display a snapshot of a random day for machine 4. We can see this machine has a very strange behavior along the day which the methods struggle to capture. In fact, the only behavior the methods seem to accurately capture is the lack of activity in the ending and beginning part of the day.

Figure 3.37: Snapshot of machine 4 in a random day: Markov Chains and Gaussian models - real data

In contrast, machine 8, like machine 6 (refer to Figures 3.26 and 3.27), is one whose behavior the methods appear to capture as we can see in Figures 3.39 and 3.40. Even though we can see this machine does not behave neatly like machine 6, the method’s estimations are still able to mimic the machine’s real behavior.

Machine 10 provides some sort of middle ground between the two last examples, illustrated in Figures 3.42 and 3.42. While for this day, the Markov Chains methods (blue and green), are able to reproduce the machines real behavior accurately, the Gaussian model methods (yellow and cyan), the Heuristic methods(red and black) and the MMSE method struggle to do so accurately.

The results so far are mostly representative of the performance of all the methods across all of the machines. While the Markov Chains methods are displayed (refer to Figures 3.31, 3.33 and 3.35) as being the most accurate, this does not happen for every machine. In fact, no method is more accurate for every machine. Similarly, no method is consistently less accurate than the rest. The performance of the methods depends largely on the behavior the machine displayed in the training dataset. The more consistent the behavior the more accurate the methods seem to be. In terms of overall behavior the machines could be separated into three groups. The first group of machines has a predictable
behavior and the methods capture this pattern with various degrees of accuracy. This is the case of machine 6, 7, 8 and 10. We have illustrated examples of this machines in Figures 3.41 and 3.40. For the second group of machines the methods are only able to accurately capture the moments in which the machine has a low activity, struggling to accurately predict the moments of activity. Such is the case of machines 1, 3, 4 (refer to Figure 3.37), and 9. The final group of machines appears to have no established pattern of behavior, which is the case of machines 2 and 5. An example of this machines is illustrated in Figure 3.43.
**Figure 3.39:** Snapshot of machine 8 in a random day: Markov Chains and Gaussian models - real data
Figure 3.40: Snapshot of machine 8 in a random day: MMSE and Heuristics - real data
Figure 3.41: Snapshot of machine 10 in a random day: Markov Chains and Gaussian models - real data

Figure 3.42: Snapshot of machine 10 in a random day: MMSE and Heuristics - real data
**Figure 3.43:** Snapshot of machine 5 in a random day: Markov Chains and Gaussian models - real data
3.3.1 Alternative initializations for the Markov Chains methods

So far, we have analyzed the results of all the methods in each of the three datasets according to the metrics established in the beginning of this chapter. Since the Markov chains methods require an initializations and until now the same initialization was used: the most probable path. In other words, we choose the most probable state for the first time slot and afterwards we choose the most probable transitions until we reach the end of the day (which corresponds to the most observed initial state and transitions). While this initialization might seem a good place to start there are other interesting alternatives worth exploring. We explored two alternatives to the initialization used so far. Two of them use the estimations of other methods as a starting point for the Markov Chains methods. We chose the MMSE method’s estimation and the Gaussian model (block size 1) method’s estimation as initializations. In other words, after estimating the consumptions of every machine on a given day, using the MMSE method and the Gaussian model (block size 1) method, for each machine, in each slot we found the closest state for that estimation.

Analysis of the results of the different initializations We begin analyzing the results of these alternative initializations for the real data by looking at the average absolute errors, illustrated in Figure 3.44. It is important to note that the label states the method used and in front the initialization. For the Decoupled Markov Chains method the difference in error between initializations is only noticeable in some machines (namely 7, 8, 9 and 10) while in the rest, even though the errors are different, no initialization is consistently better. For the Coupled Markov Chains method in half the machines (namely 2, 8, 9 and 10) the average error of the new initializations is considerably larger when compared with the initialization used by default. The error is also bigger in machines 3 and 4, however the difference is no longer considerable. In machine 6 the situation is reversed and the estimations resulting from the new initializations have a considerably lower average error. In the remaining machines (namely 1, 5 and 7) the difference in errors between initializations is minimal.

We now take a look at histograms of the relative errors for three machines. The histogram for machine 6 is illustrated in Figure 3.45 and we can see that after the 20% mark, all the methods are similar. However, the alternative initializations of the Decoupled Markov Chains method (green and red) only manage to estimate around 60% of the outputs with a relative error below the 10% mark while the remaining initialization (most probable path illustrated in blue) manages to output 90% of the data with an error less or equal to 10%. The initialization illustrated by the red curve (the initialization provided by the Gaussian model (block size 1)) is overlapped by the results of the initialization illustrated by the green curve (the initialization provided by the MMSE method) making it hard to distinguish them. For this machine the Coupled Markov Chain method behave similarly regardless of the initialization, achieving an error inferior to 10% for more than 85% of the data.

Machine 8’s results, illustrated in Figure 3.46 tells a different story. While both methods manage to estimate more than 35% of the data with an error below the 10% mark, the alternative initializations allow the Decoupled Markov Chain method to slightly improve it’s estimations, whereas the Coupled Markov chain method not only does not improve it’s estimations but also looses accuracy when taking
The results of machine 10, illustrated in Figure 3.47, provide a more general picture of what happens in the remaining machines (1 through 5 plus machine 7). We can see that while the alternative initializations for the Coupled Markov Chains method (illustrated in cyan and magenta) are significantly less accurate than the default initialization (illustrated in black), the alternative initializations of the Decoupled Markov chain methods (illustrated in green and red) only achieve the same levels of accuracy of the default initialization (illustrated in blue) when considering errors above the 15% mark.

In general, the alternative initializations only achieve the same level of accuracy for higher error marks, or they never achieve them at all. In the best case, these alternative initializations only achieve the same level of accuracy of the default initializations (the exception being machine 8). For this case neither the confusion matrices nor the snapshots of a machine for a random day are presented because the different initializations produce similar results for this types of analysis.

**Figure 3.44:** Different Markov Chains initializations: Average absolute errors - real data
Figure 3.45: Different Markov Chains initializations: Histogram of relative errors for machine 6 - real data

Figure 3.46: Different Markov Chains initializations: Histogram of relative errors for machine 8 - real data
Figure 3.47: Different Markov Chains initializations: Histogram of relative errors for machine 10 - real data
3.4 Side information

In Chapter 2, section 2.8, we explained how to incorporate side information in some of the models in order to ascertain which machines provide valuable information to guess the other machines’ states. We now briefly analyze some results resulting from the additional information provided by each machine.

We begin by looking at the average absolute error for each machine when we include the information of machine 8, for the Decoupled Markov Chains method. We illustrate this result in Figure 3.48. The blue bars represent the average error for a given machine when we add the observation of one of them. For the observed machine this error is zero. The red bars represent the average absolute error when no machine is observed. We can see that for some machines the average error decreases (machines 9 and 10) while for others it slightly increases (machine 2) or remains mainly the same.

![Figure 3.48: Side information: Average absolute errors for the Decoupled Markov Chains method when machine 8 is observed](image)

We illustrate another example for the other two methods (Gaussian model (block size 1) method and the MMSE method) in Figures 3.49 and 3.50. In Figure 3.49 we illustrate the average absolute errors when we observe machine 7 for the Gaussian model (block size 1) method, while in Figure 3.50 we illustrate the average absolute errors when machine 8 is observed for the acMMSE method. We can see the change in average errors depends on the machine, for some it slightly decreases while for others it slightly increases or remains mainly the same. These figures are illustrative of the overall results.

Figures 3.51, 3.52 and 3.53 show the average error across all machines when a given machine is observed. We compare these results with the average error across all machines when no side information is included in the methods (displayed in position 0, of the figures). We can see that
the machines whose information has a bigger impact on the average errors across all machines are machine 7 and 8. It is interesting to note that machine 8 is the group of equipments whose consumption is not directly measured. It is also interesting to see that it is these two machines that are estimated with the highest average errors, and the results of Figures 3.48, 3.49 and 3.50 seem to indicate that this is mainly due to the fact that the observed machine no longer contributes to the estimation errors. However this metric is insufficient to determine if machines 7 and 8 really are the most informative because these machine’s estimations have the biggest average error, an error the methods do not incur if they do not have to estimate the machine.
Figure 3.50: Side information: Average absolute errors for the MMSE method when machine 8 is observed

Figure 3.51: Side information: Average absolute errors for the Decoupled Markov Chains method
Figure 3.52: Side information: Average absolute errors for the Gaussian model (block size 1) method

Figure 3.53: Side information: Average absolute errors for the MMSE method
Conclusions
We have developed and analyzed the performance of a suite of methods for a dataset of ten real groups of equipment, whose consumptions were measured. We could see that certain machines were more accurately estimated than other, however, none of the methods was consistently better across all machines. In some cases, all of the methods were extremely inaccurate while in other cases all methods showed a good accuracy on the estimations. It was curious to note that for some machines, our methods managed, with varying degrees of precision, to capture the machine’s overall behavior (for example machine 6, which corresponds to the illumination, or, more curiously, machine 8, which is the group of equipments which weren’t directly measured), while in others no method managed to do so, or only managed to capture the moments of inactivity. This might be a result of the slow acquisition rate (one sample every 15 minutes). We also analyzed several alternative initializations of the Markov Chains methods, however, these initializations did not produce a large difference in accuracy, when compared to the default initialization.

Finally we briefly analyzed the results of incorporating side information in some of the methods and while these results seem to indicate that the most informative machines are machines 7 and 8, machine 8 was not directly measured due to impractical reasons leaving only machine 7 as a possible target for a sensor. However, this brief analysis of the inclusion of the side information is not enough to conclude if machine 7 is indeed the best option for the placement of a sensor. While no method has showed to be more accurate for a generic machine we have seen that these methods are capable of modeling the behavior of some of the machines and therefor these methods can be useful in a wide array of problems regarding disaggregation of quantities.
Bibliography


Derivation of the majorization function
For a generic quadratic function:

\[ g(z) = a + b^\top z + \frac{1}{2} z^\top C z, \]

the Mean-value theorem states that

\[
g(z) = g(z_0) + \nabla g(z_0)^\top (z - z_0) + \frac{1}{2} (z - z_0)^\top \nabla^2 g(w) (z - z_0)
\]
\[
= g(z_0) + (C z_0 + b)^\top (z - z_0) + \frac{1}{2} (z - z_0)^\top C(z - z_0).
\]

If we apply the Mean-value theorem to the first term of the objective function (2.12), given by:

\[
(y(t) - c^\top e(t))^2 = y(t)^2 - 2y(t)c^\top e(t) + c^\top ee^\top c(t),
\]

we have

\[
(y(t) - c^\top \tau(t))^2 + (-2y(t)c + 2cc^\top \tau(t))^\top (e(t) - \tau(t)) + \frac{1}{2} (e(t) - \tau(t))^\top 2cc^\top (e(t) - \tau(t))
\]
\[
= (y(t) - c^\top \tau(t))^2 - 2(y(t) - c^\top \tau(t))c^\top (e(t) - \tau(t)) + [(e(t) - \tau(t))^\top c]^2.
\]

We know that

\[
[(e - \tau(t))^\top c]^2 \leq \|c\|^2 \|e(t) - \tau(t)\|^2
\]

since

\[
(u^\top v)^2 \leq \|u\|^2 \|v\|^2,
\]

and so we get

\[
(y(t) - c^\top e(t))^2 \leq (y(t) - c^\top \tau(t))^2 - 2(y(t) - c^\top \tau(t))c^\top (e(t) - \tau(t)) + \|c\|^2 \|e(t) - \tau(t)\|^2,
\]