Non-Intrusive Load Monitoring (NILM) Techniques

Machine Learning Engineer Nanodegree Capstone Project

Abdulhakeem Abdulrahman

September 27, 2018

I. Definition

1. Project Overview

Up to 60% of the world's electricity consumption is contributed by both residential and commercial buildings according to the United Nations Environment Programmes Sustainable Building and Climate Initiative (UNEP-SBCI) [1]. However, the conventional energy monitoring only captures the aggregate of all the electrical loads in a building. An accurate and real-time monitoring of the appliance-level consumptions offers actionable feedback and insights into effective energy saving behaviour, significantly stimulating the reduction of the overall energy demand. The current approach deploys sensors that monitor the individual appliance consumption in buildings. This approach is both cumbersome and costly among other limitations and has therefore, together with the introduction of smart energy meters, led to the consideration of energy disaggregation techniques [2].

Energy disaggregation (also known as nonintrusive load monitoring or NILM) is a computational technique that breaks down the single point source data, recognizes the load appliances running in a building, and estimates their individual power consumption. Simply, NILM takes a building energy signal from a single monitoring point and separates it into its component appliances power. George W. Hart started the research into residential energy disaggregation at MIT in the 1980s, proposing the first NILM event-based approach using the state transition edges (corresponding to on/off events) in appliances power signals to cluster them [3]. Several NILM methods have been proposed which include combinatorial optimisation (CO), factorial hidden Markov models (FHMM), non-negative matrix factorization (NMF), and deep neural networks [4]. In addition, recent approaches have explored different features and hyper-features of the alternating current waveforms and used complex device models with multiple states [5].

This project explores the NILM techniques using the NILM Tool Kit (NILMTK) [6] on a publicly available Reference Energy Disaggregation Dataset (REDD) [5] and seeks to discover the effects of the data preprocessing as well as parameters fine-tuning on the performance of two benchmark algorithms: combinatorial optimisation and factorial hidden Markov models (FHMM using exact inference).

2. Problem Statement

The increasing need for energy saving, as well as the accelerating computing power and the advancement in machine learning, has led to the popularity in research on NILM using machine learning techniques. However the promising value of energy disaggregation and the popularity of this field, none of the approaches has proven to successfully solve the real-life disaggregation problem; there is no precise, reliable and general method that can practically be deployed in a household, hence, the need for performance improvements in the algorithms. While it may be too ambitious to develop a new algorithm that solves this problem in this project, exploring and comparing the performance of the NILM algorithms by changing the pre-processing (such as sampling rate, normalization using voltage, and using reactive power) and fine-tuning the hyper-parameters, seeking to get better performance for practical solution proves to be worthwhile.

3. Metrics

For empirical evaluation of NILM algorithms and fair performance comparison between different approaches, it is important to define the evaluation standards. Although energy disaggregation algorithms are evaluated on a number of metrics such as accuracy, F-Score, fraction of energy correctly assigned, mean normalized error power, and rms error power, this project uses Root Mean Square Error (RMSE) power to evaluate the learning models on each of the appliances. RMSE measures the differences between values predicted by a model or an estimator and the observed values or ground truth. Therefore, it is suitable since NILM is not merely determining the presence of an appliance or uptime but more importantly estimating its numerical contribution to the aggregate value. The RMSE is calculated using the following equation:

$$RMS\ Error = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - O_i)^2} \qquad \text{Eqn. (1)}$$

Where n is the number of samples, P_i is the predicted value at index i, and O_i is the observed value at index i.

II. Analysis

1. Data Exploration

This project uses the Reference Energy Disaggregation Dataset (REDD) (available:http://redd.csail.mit.edu), a publicly available dataset containing power usage information for several homes. This reference dataset is aimed to be the benchmark dataset to energy and sustainability domain such as the MNIST digit recognition in machine vision domain, specifically targeting the task of energy disaggregation.

REDD contains measurements of both whole-home and appliance-level electricity consumptions of some real houses over a period of several months. Each house measurements include the whole-home electricity signals (both current and voltage) recorded at a high frequency of 15 kHz; 24 individual appliances signals at 0.5 Hz, each labeled with its category; and 20 plug-level measurements recorded at 1 Hz, to cater for appliances that are grouped and powered from the same wall socket outlet. Overall, there are 10 monitored homes and total of 119 days of data across the homes.

Because of the large size of the high frequency dataset (which contains more physical quantities described previously), this project uses the low frequency version of the dataset. This version contains whole-house apparent power measurement sampled at 1 second and appliance-level active power measurement sampled at of 3 or 4 seconds. The appliances in one of the buildings are: dishwasher, waste disposal unit, microwave, fridge, light, sockets, washer dryer, and electric stove. *Sampling period (measured in seconds) is the inverse of sampling rate (or frequency measured in hertz or Hz): the higher the sampling period, the lower the sampling rate and vice versa.*

2. Exploratory Visualization

Figure 1 shows the plot of a whole-house mains power measured by two meters called site or mains meters while figure 2 shows the plot of the sub-metered power contributions from each of the appliances. Figure 3 shows a light power profile. These figures provide the summary of the characteristics and attributes of the data considered in this project.

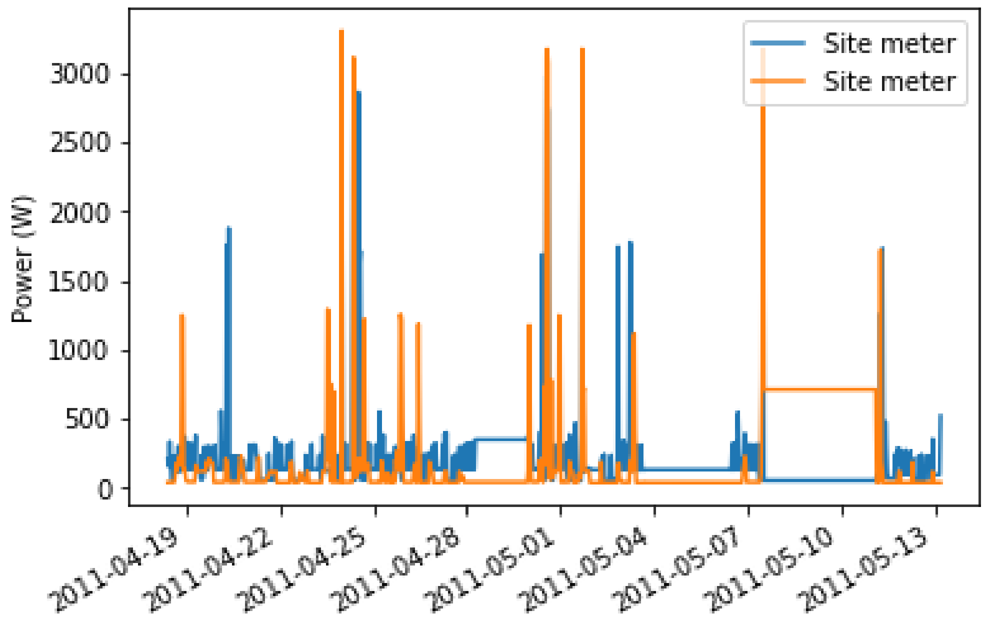

Figure 1. Mains power measurements

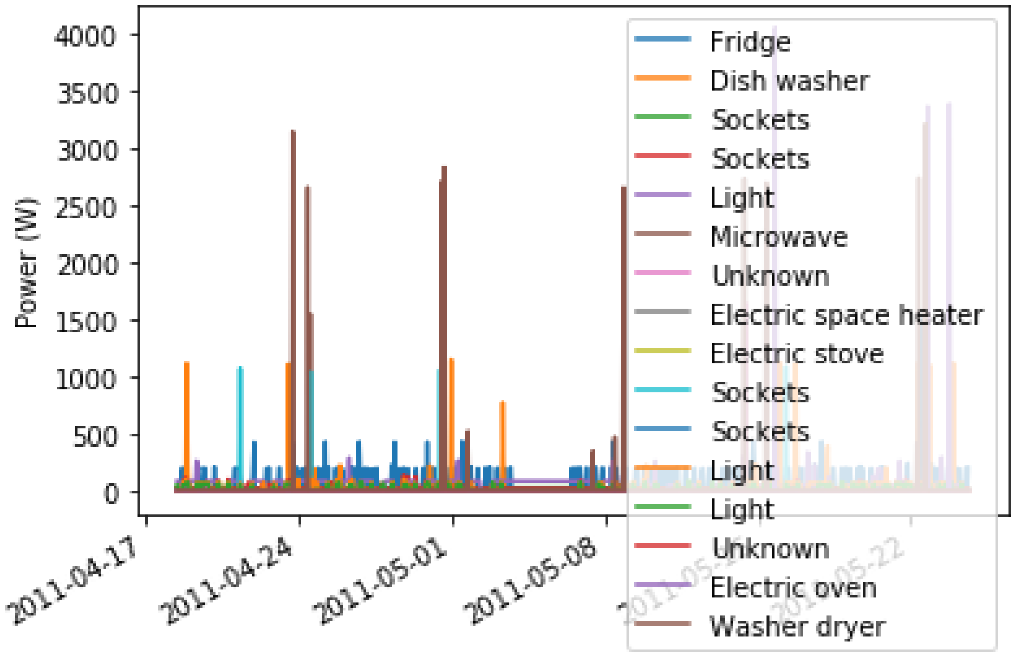

Figure 2. Submeters power measurements

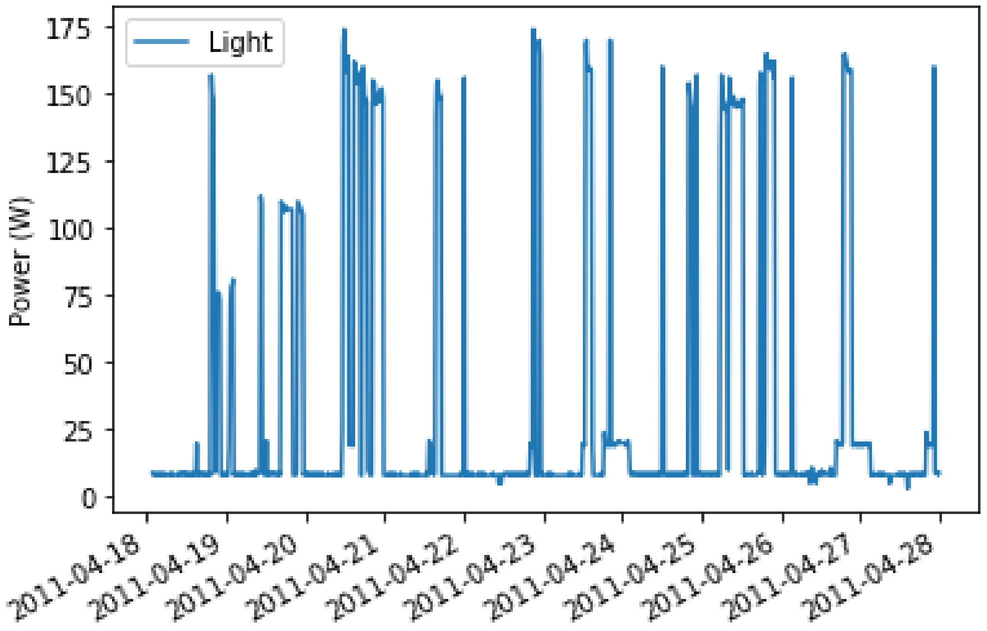

Figure 3. Light power measurements

3. Algorithms and Techniques

Two NILM algorithms are used in this project: combinatorial optimisation (CO) and factorial hidden Markov model (FHMM using exact inference). These are parts of the benchmark algorithms implemented in NILMTK and are relatively computationally inexpensive to train on our dataset. The disaggregation algorithms learn a model of the appliances energy consumption from existing buildings, therefore they generally require appliance-level data from one or more buildings. Each algorithm requires a group of mains and sub meters data, iterates over chunks of the data, and returns a learned model stored in memory which can be used to disaggregate seen or unseen mains meters data into its component appliances power.

The model learned by each of the algorithms separates a site meters or mains' load into individual appliances loads. Like in the training, the model iterates over chunks of aggregate data and returns a panda dataframe, with columns corresponding to individual appliances and rows represent time instants. The returned dataframe indexes matches that of the parameter dataframe used in training the model.

A. Factorial Hidden Markov Model (FHMM):

The factorial hidden Markov model is an extension of hidden Markov model; it has multiple independent hidden state sequences upon which each observation depends [7]. The following equations describe the observable and hidden sequences relations:

$$S = \left\{ S^{(1)}, S^{(2)}, S^{(3)}, \ldots\ldots S^{(M)} \right\} \qquad \text{Eqn. (2)}$$

$$S^{(i)} = \left\{ S_1^{(i)}, S_2^{(i)}, S_3^{(i)}, \ldots\ldots S_T^{(i)} \right\} \qquad \text{Eqn. (3)}$$

$$Y = \{ y_1, y_2, y_3, \ldots\ldots, y_T \} \qquad \text{Eqn. (4)}$$

Where S represents the set of hidden state sequences, $S^{(i)}$ is the hidden state sequence of the chain shown in figure 4, and Y is considered the observable sequences.

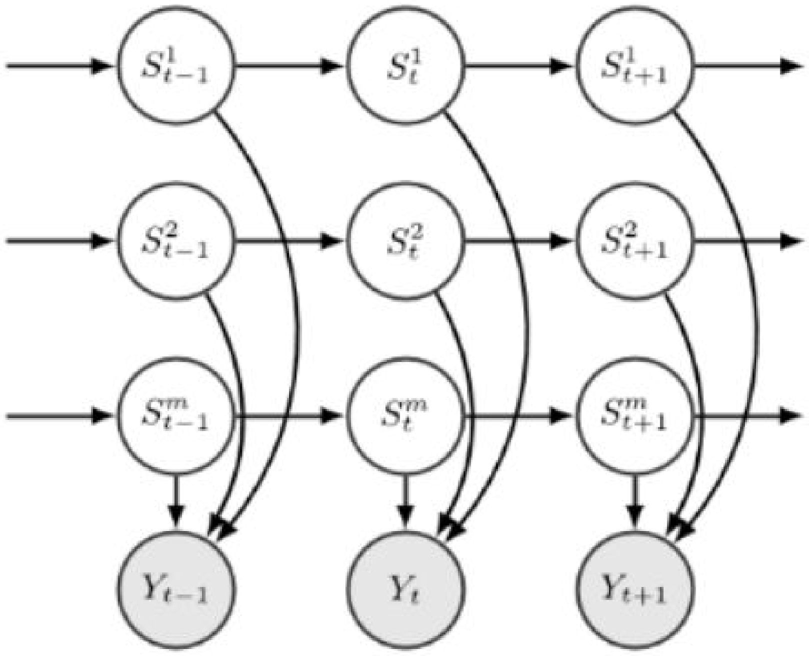

Figure 4. Graphical representation of the factorial hidden Markov model [1]

B. Combinatorial Optimization (CO):

NILM as a combinatorial optimization problem, in which events detection are not necessary, assumes that the whole-house measurement are made up of and can be decomposed into individual components shown in Equation 5. This formulation is presented by Hart [3], and it seeks to find the optimal combination of appliances' power consumption to minimize residual energy [6]. CO’s power draw is not related in time and it is similar to subset-sum problem and thus NP-complete.

$$P(t) = \sum_{i=1}^{n} x_i(t) P_i + \varepsilon \qquad \text{Eqn. (5)}$$

Where $P(t)$ is the measured variable (in our case, apparent power) of the whole-house at instant of time t, $x_i(t) \in \{0, 1\}$ is a boolean that determines the power state status i at time t, and P_i represents the target power states which the NILM CO aims to decompose into.

4. Benchmark

The NILMTK compares the two benchmark algorithms by downsampling the low frequency REDD data set to a resolution of 120 seconds and selecting the top-k appliances in terms of energy consumption and use them for training the models. The RMSE power for each of the appliances from the predictions by each of the models are summarized in table 1. Both training and disaggregation run-times are also an important performance parameter, although these values depend on largely on the available computation resources but they must be low enough (with prediction time possibly around a few seconds) for real-time load disaggregation.

Table 1. NILMTK RMSE benchmark in Watts (W)

	CO	**FHMM**
Dish washer	222.076767	183.716326
Fridge	189.811752	165.582688
Light	122.587723	92.281793
Microwave	192.521194	177.342418
Sockets	28.782600	47.876566

III. Methodology

1. Data Preprocessing

A. Sampling Rates and Downsampling:

Data preprocessing is the main focus of this project and it involves exploring sampling rates and downsampling the power measurements from the mains (or site) meters and the appliance-level sub meters. Downsampling (or subsampling) is the process of reducing the sampling rate of a signal, usually done to reduce the data rate or the size of the data. REDD dataset, specifically the low frequency version, has appliance-level data sampled every 3 to 4 seconds and the mains data sampled every 1 second. Downsampling is therefore necessary since the disaggregation algorithms needs both the two classes of data sets, processed the same way, to train each of the appliances as well as to disaggregate the site meters data into appliances components. It also reduces the data size since some trains of measurements have the same values. Tables 2 and 3 show the sampling periods of both the mains meter and sub meters of a building in the dataset.

Table 2. Mains meter power sampled at 1 second period

Physical_quantity	**Apparent Power in Watts (W)**
2011-04-17 19:18:27-04:00	306.809998
2011-04-17 19:18:28-04:00	306.319977
2011-04-17 19:18:29-04:00	306.400024

Table 3. Fridge submeter power sampled at 3 or 4 seconds period

Physical_quantity	**Active Power in Watts(W)**
2011-04-18 01:31:40-04:00	6.0
2011-04-18 01:31:44-04:00	6.0
2011-04-18 01:31:47-04:00	6.0

B. Good Sections and Dropout Rate:

When we plot the raw power data, we will observe there are gaps where, supposedly, the metering system was not working or the data are lost along the channel especially in wireless sensors. The sections with continuous data are the good sections are the useful part for training and disaggregation purposes. Figures 5 and 6 show the plots of raw data and good sections of fridge sub meter power in one of the buildings in the dataset.

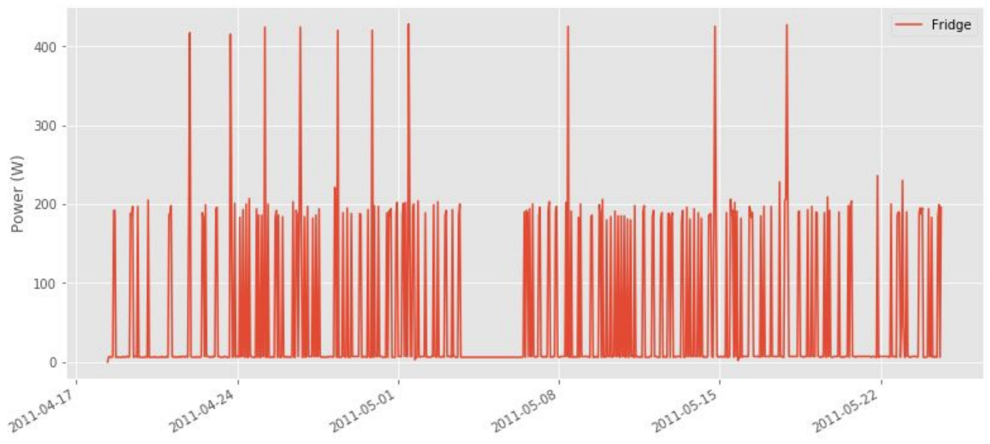

Figure 5. Plot of fridge submeter raw power

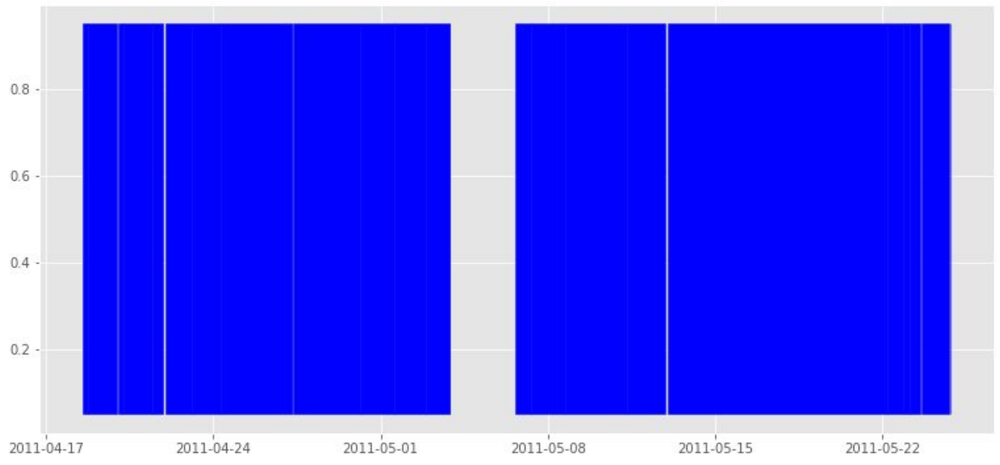

Figure 6. Plot of fridge submeter good sections

Dropout rate is a number between 0 and 1 which specifies the proportion of missing samples, or measurements intervals larger than the sampling period. A value of 0 means no samples are missing while a value of 1 means all samples are missing. Downsampling reduces the dropout rate as it widens the sampling period. Overall, the preprocessing involves resampling the data, filtering out erroneous readings, and finding and removing gaps in the data.

2. Implementation

This project uses NILMTK, a toolkit specifically designed for energy disaggregation problems. This toolkit is similar to the various toolkits for generalized machine learning tasks such as the scikit-learn. NILMTK includes the tools such as data set parsers, benchmark algorithms, and evaluation metrics specific for the energy disaggregation domain. This toolkit does not replace the existing general toolkits rather it extends them, similar to the way scikit-learn adds machine learning functionality to the numpy API for Python. Figure 7 highlights the NILMTK pipeline from datasets import to the evaluation of algorithms.

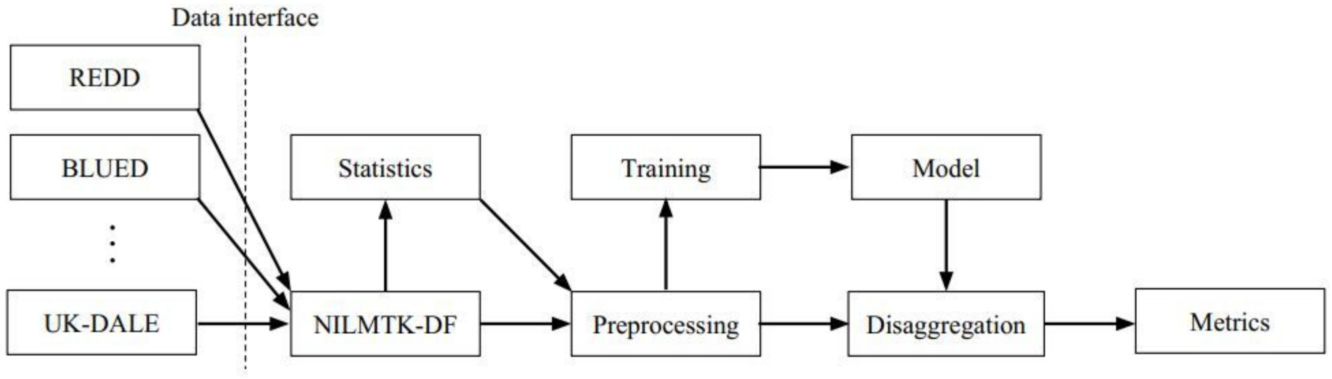

Figure 7. NILMTK Pipeline [6]

A. Dataset Import and Data Format:

The REDD dataset is imported and converted into the NILMTK HDF5 format. The data is then saved in this format and loaded from disk into memory for further operations. It is good to note that at multiple stages in the NILMTK processing pipeline, the data can be saved and loaded to allow other tools to interact with NILMTK. Also, the toolkit allows specification of sampling period when loading the dataset into memory as well as when training a disaggregator model.

B. Dataset Summary:

The loaded dataset is explored, specifically showing certain features from its metadata. These include types of appliances in a sample building, types of attributes (physical quantities measured), how the data are collected, and details of the sampling rates. Insights into the data structure and characteristics is explored and the data summary is presented.

C. Data Splitting and Preprocessing:

A sample building data is selected and split into the training and test sets. The start and end dates of the data measurement are inspected, then two data windows are set by picking a date that corresponds to the end and start dates of the train and test datasets respectively, corresponding to a split ratio of 3:1. Then the top-5 appliances in terms of energy consumption are selected for training the models.

The raw data as well as the good sections are plotted and the dropout rates are calculated for both the site meters and a fridge submeter. The data is subsampled and the gaps and erroneous readings are removed as described in section B under methodology.

D. Training and Disaggregation:

Helper functions which facilitate training, prediction, and validation in iterations of data over different sampling periods are created and are explained as follows:

find_intersection*:* this function captures the intersecting readings in both the ground truth and the predicted data by first converting the measurement times to UTC and then back to the local timezone as described in the metadata.

predict: this function takes a trained classifier model, test dataset, sampling period, and timezone as parameters. It disaggregates chunks of test dataset to its component submeters data, extracts the corresponding submeterer ground truth data, and returns both of them.

compute_rmse: this function takes both the predicted and ground truth data and calculates the RMSE power for each of the appliances.

run_disaggregator: this takes a classifier model, training dataset, test data, sampling period, and timezone and trains the classifier on the train dataset chunks, makes predictions on the test dataset chunks using the trained model, computes the errors, and records the runtimes for both training and disaggregation.

The disaggregation process is first run with FHMM_EXACT classifier and a sampling period of 60 seconds on the dataset. Then performances in terms of both the runtimes and the RMS errors are assessed and present in tables 4 and 5.

Table 4 . FHMM_EXACT runtime at 60 seconds sampling period

FHMM_EXACT runtime	**Runtime in seconds**
prediction	6.810888
training	3.238356

Table 5. FHMM_EXACT RMSE power at 60 seconds sampling period

FHMM_EXACT RMSE	**Errors in Watts (W)**
Dish washer	140.655961
Fridge	140.195129
Light	127.499494
Microwave	58.879628
Sockets	44.612464

A small chunk of the test mains meter data, its real submeters data composition, and the values disaggregated to by the model are assessed. The disaggregation processes are run over sampling periods of 10 seconds through 120 seconds in a step of 10 seconds and the corresponding performance parameters are recorded. It is worth to note that the implementation of Combinatorial Optimisation in the NILMTK does not support multiple calls to `train` method on the same classifier instance, therefore, an object of CO is instantiated in each of the iterations.

E. Performance Evaluation:

The performance parameters - runtimes and RMS errors - are converted to Pandas Series data and then each added to a separate dictionary with the keys in each case being the sampling periods. Two Pandas data frames are made out of these dictionaries of Series data and the transposes are computed to swap the indexes; having the appliances on the columns axis and the sampling periods on the row index.

3. Refinement

This project uses two of the benchmark disaggregation algorithms and focuses on the preprocessing of the data for the techniques. Downsampling reduces the dropout rate, the runtimes, and memory required in running the techniques.

IV. Results

1. Models Evaluation and Validation

The recorded performances parameters - runtimes and RMS errors - for both FHHM_EXACT and CO classifiers are presented in figures 7 through 10.

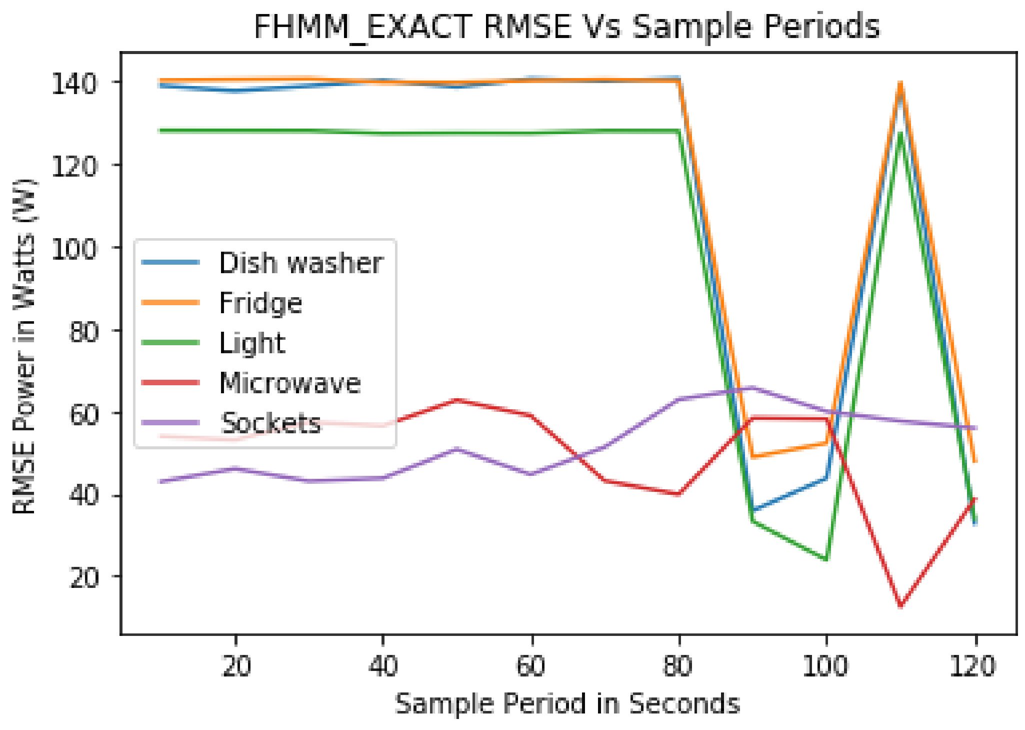

Figure 7. Plot of appliances RMSE power for FHMM_EXACT model

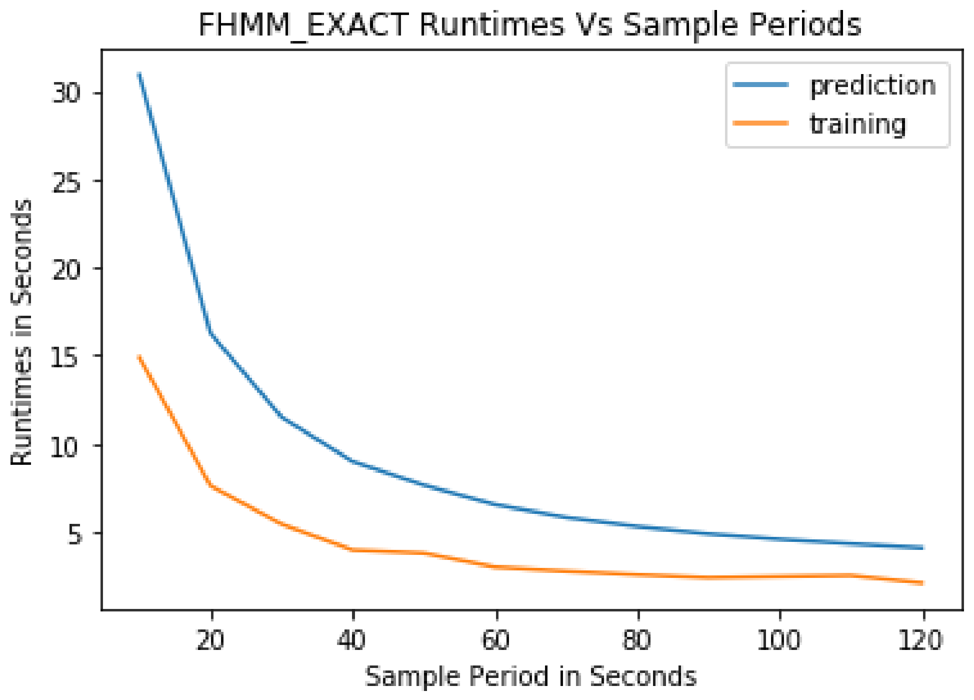

Figure 8. Plot of training and disaggregation runtimes for FHMM_EXACT model

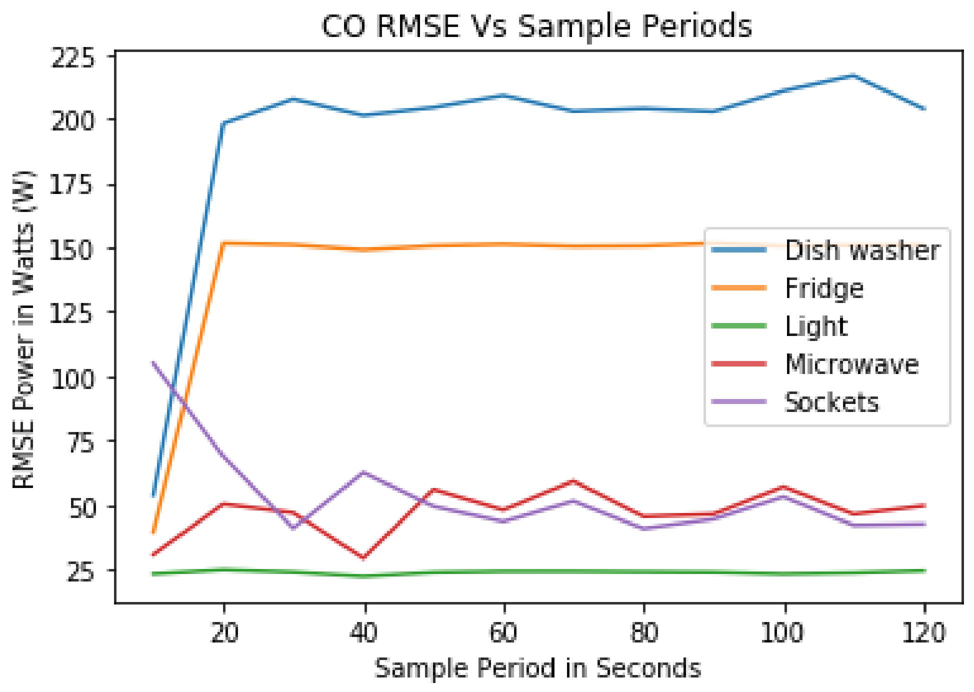

Figure 9. Plot of appliances RMSE power for CO model

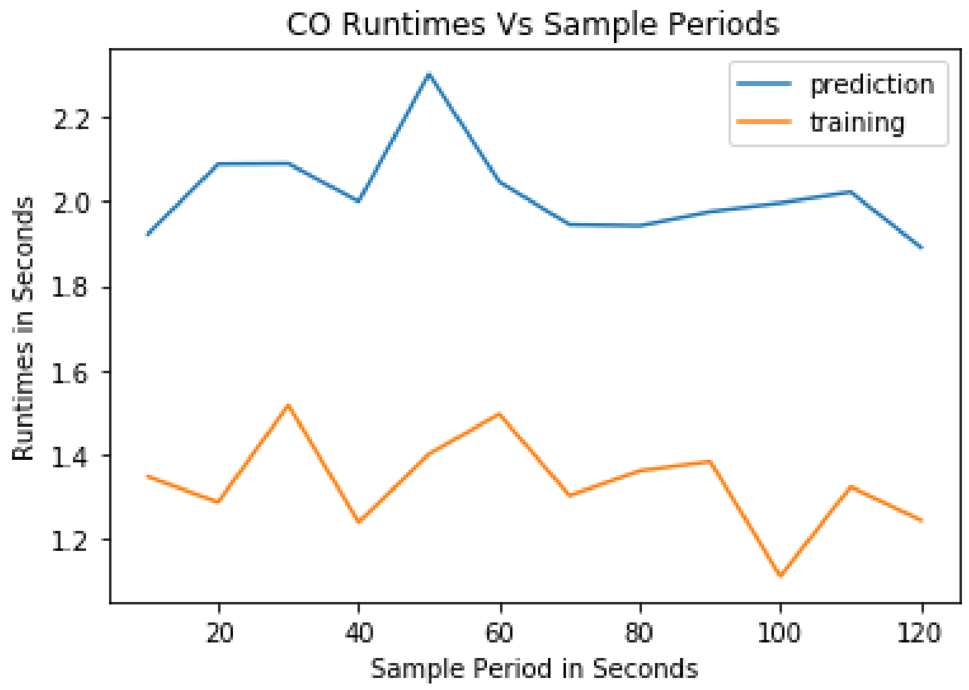

Figure 10. Plot of training and disaggregation runtimes for CO model

2. Justification

Varying the sampling period does not improve the FHMM model, as the RMSE power computed for each of the appliances remains fairly constant until a sampling period of 80 seconds when the values dropped for dishwasher, fridge, and light. However, the runtimes decreases with increasing sampling period (or decreasing sampling rate). This is expected as FHMM models time variation; the higher the sampling rate the longer it takes to train and disaggregate data.

On the other hand, varying the sampling period increases the RMSE power disaggregated by CO until a sampling period of 20 seconds after which the values remain unchanged for all of the 5 appliances. The runtimes remain fairly constant for different sampling periods, suggesting that subsampling the data does not save both training and and disaggregation times.

Overall, the RMSE power obtained at lower sampling periods or higher sampling rates are better than the benchmark figures for both FHMM_EXACT and CO models trained on benchmark sampling period of 120 seconds. This is also true for a period of 120 seconds with data gaps and errors filtered out. The runtimes can not really be compared since the computation resources available highly determine them.

V. Conclusion

1. Free-Form Visualization

Figure 11 compares the fridge submeter real power and power predicted by FHMM_EXACT model. The model successfully captures the appliance uptime and does well in estimating its power consumption over time. However, the model fails to explain or capture its transient state values as shown in the ripples of the true power train. This may be due to the low sampling rates, however, downsampling reduces the data size and hence reduces the runtime since some trains of measurements have the same values, thus the trade-off is worth it.

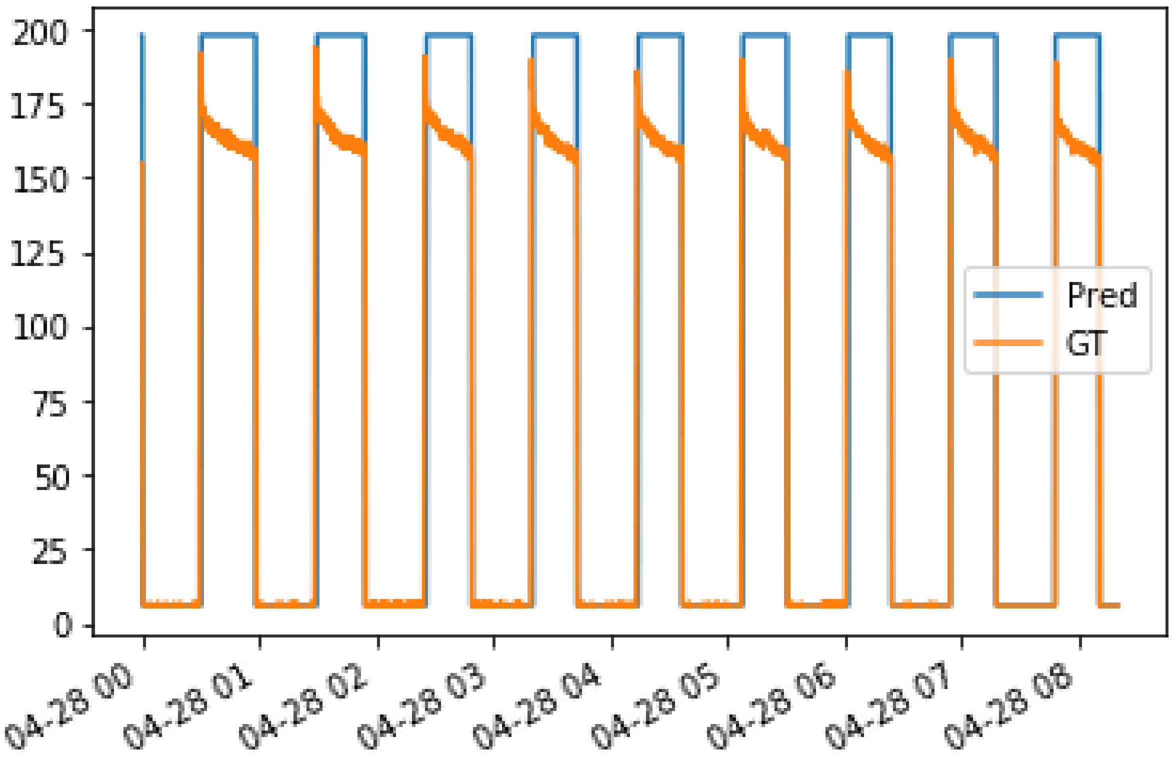

Figure 11. Plot of fridge submeter real and predicted power using FHMM_EXACT model

2. Reflection

This project is able to compare two benchmark NILM disaggregation algorithms and explore the effects of sampling rates and downsampling on the performance parameters such as the RMS errors and runtimes of the algorithms. It is however difficult to have generality across multiple appliances despite that each of them is trained separately. Also, it took me some time to conclude on the disaggregation performance metrics to consider for the empirical comparison and assessment of the techniques.

I chose RMS error power instead of accuracy, F1_Score etc., as it does not only determine the presence or uptime of appliances but also estimates its numerical contribution to the aggregate power values. Also, I consider the training and disaggregation runtimes as a performance metric, especially because the disaggregation time determines the real-timeness of such techniques and applicability in real time load monitoring. Addressing generality and finding consistent performance metrics remain parts of the core challenges in NILM research.

I found it interesting to verify that FHMM outperforms CO across the REDD dataset, but CO is exponentially quicker than FHMM, perhaps because FHMM models time variations.

3. Improvement

Using a high frequency dataset will definitely give more insights into the effects of sample rates and allows subsampling over a wider useful sampling periods. Alos, high sampling rates have the potential of capturing the transient state behaviors of the appliances as displayed in Figure 11, the plot of real and predicted values of fridge submeter power.

Similarly, using a dataset with more more attributes such as voltage and reactive power, and especially normalizing using the voltage will yield better performance. Finally, other machine techniques such as deep learning using deep neural networks will be worth exploring.

References:

[1] Faustine, Anthony & Mvungi, Nerey & Kaijage, Shubi & Kisangiri, Michael. (2017). A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem.

[2] W.H.C. Janssen, Energy disaggregation: Nonintrusive load monitoring: The search for practical methods, Master of Science Thesis, Delft University of Technology, Jan. 2018.

[3] G.W. Hart. Nonintrusive appliance load monitoring, Proceedings of the IEEE, 80(12):1870-1891, 1992.

[4] Kelly, Jack & Knottenbelt, William, Neural NILM: Deep Neural Networks Applied to Energy Disaggregation, Conference: 2nd ACM International Conference on Embedded Systems For Energy-Efficient Built Environments, 2015.

[5] Zico Kolter, J & J Johnson, Matthew. REDD: A Public Data Set for Energy Disaggregation Research. Artif. Intell, Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability, 2011.

[6] Batra, Nipun & Kelly, Jack & Parson, Oliver & Dutta, Haimonti & Knottenbelt, William & Rogers, Alex & Singh, Amarjeet & Srivastava, Mani. (2014). NILMTK: An open source toolkit for non-intrusive load monitoring. e-Energy 2014 - Proceedings of the 5th ACM International Conference on Future Energy Systems. 10.1145/2602044.2602051.

[7] Z. Ghahramani and M I Jordan. Factorial Hidden Markov Models. Machine Learning, 29(2):245–273, 1997.