Programme Area: Smart Systems and Heat

Project: WP1 Appliance Disaggregation

Title: Final Report

Abstract:

This deliverable presents the outcome of the project by providing a short and high-level description of the algorithms developed, and how they can be run. The details of the algorithms referred to here have been described in the previous deliverables.

Context:

The High Frequency Appliance Disaggregation Analysis (HFADA) project builds upon work undertaken in the Smart Systems and Heat (SSH) programme delivered by the Energy Systems Catapult for the ETI, to refine intelligence and gain detailed smart home energy data. The project analysed in depth data from five homes that trialed the SSH programme's Home Energy Management System (HEMS) to identify which appliances are present within a building and when they are in operation. The main goal of the HFADA project was to detect human behaviour patterns in order to forecast the home energy needs of people in the future. In particular the project delivered a detailed set of data mining algorithms to help identify patterns of building occupancy and energy use within domestic homes from water, gas and electricity data.

Project: HFADA

HIGH FREQUENCY APPLIANCE DISAGGREGATION ANALYSIS

Final Report

Contents

1. History

Date	Issue	Details of Change
	Version 0.0	Initial Version. Authors: Ngoc Canh Duong Waqas Jamil Hamid Bouchachia

2. Documents Referenced

Ref	Document	Title
1	Word document that describes the HEMS data.	Data collection and data format – ELECTRIC, WATER and HEMS-V1 MONITORING
2	Word document that describes the HEMS V1 Mongo data base structure.	HEMS V1 Mongo Data Base Structure
3	Deliverable 1	HFADA_Deliverable 1
4	Deliverable 2	HFADA_Deliverable 2
	Deliverable 3	HFADA_Deliverable 3
5	Deliverable 4	HFADA_Deliverable 4
6	Deliverable 5	HFADA_Deliverable 5
7	Paper	Deep Online Hierarchical Unsupervised Learning for Pattern Mining from Utility Usage Data
8	Paper	Deep Online Hierarchical Dynamic Unsupervised Learning for Pattern Mining from Utility Usage Data
9	Paper	Competitive Normalised Least Squares Regression
10	Paper	Competitive Online Regularised Regression

3. Glossary of Terms

Ref	Description
ETI	Energy Technologies Institute
LDA	Latent Dirichlet Allocation
OGLDA	Online Gaussian Latent Dirichlet Allocation
DBN	Deep Belief network
HMM	Hidden Markov Model
DBN-LDA-HMM	Deep-Hierarchical-Dynamic model
OSLOG	Online Shrinkage via Limit of Gibbs sampling
COIRR	Competitive Online Iterated Ridge Regression
ONLMS	Online Normalised Least Mean Squares Regression

4. Executive Summary

This deliverable presents the outcome of the deliverable 1-5 by providing a short and high-level description of the algorithms developed, how they can be run. The details of the algorithms referred to here have been described in the previous deliverables.

5. Introduction

The basic aim of the HFADA project is to develop pattern mining from multiple sources of data utility: electricity, water flow, gas flow, and some sensory measurements (mostly humidity and temperature) with the ultimate goal to identify useful occupancy and energy usage patterns.

In doing so, HFADA was quite ambitious in developing many new algorithms to

1- Extract features from raw data after reading the hard disks which was done by Deliverable 1.
2- Summarise the utility data in the form of clusters/topics/components which are assumed to provide a representation of the consumption patterns or more precisely appliance usage. This was the target of Deliverable 2.
3- Capture the temporal dynamics hidden in the data in order to understand the relationship between the topics (implicitly representing appliances usage), that is, sequences of appliance usages, which could hopefully and ideally correspond to the daily human activity. This was the target of Deliverable 3.
4- address the problem of prediction of hot water usage and gas using online regression models as a way of understanding and predicting occupancy. While there is no access to the quantity of hot water consumed over time, the inference about the usage of hot water can be made using the temperature of the domestic hot water flow and the central heating flow (see Deliverable 4).

The proposed algorithms are all online, so data is processed sequentially in (pseudo) real-time and the produced algorithmic models are self-adaptive (Deliverable 5). In addition, a parallel/distributed version of the algorithms is proposed which allow to handle big data efficiently. Only the algorithm proposed in deliverable 3, DBN-LDA-HMM, is not distributed, but is online.

In the following we highlight the algorithms and provide their implementation details.

6. Algorithms

6.1 Online Gaussian Latent Dirichlet Allocation (OGLDA)

This algorithm represents the outcome of D1 and D2 and is implemented in Python. It reads utility usage data from CSV files that were created through feature extraction presented in Deliverable 1.

Requirements

- Python (preferably Python 2.7.12) should be installed.
- The following packages are used: numpy(1.13.3), scipy(1.0.0), ray(0.4.0), csv, glob datetime, , time, re, random, argparse and sys.
- Copy data files to "./data" directory. The files should be in CSV format.

Running the code

There are two runnable scripts, *sequential_OGLDA.py* and *parallel_OGLDA.py*. *sequential_OGLDA.py* takes no argument and invokes the OGLDA model to process data in sequential order. Here is an example command to run the script:

```
python sequential_OGLDA.py
```

*parallel_OGLDA.py t*akes --num-workers as an argument, 4 by default. This code can be run on a single machine to achieve efficient multiprocessing, or it can be used on a cluster for large computations (for using Ray on a large cluster).

When using the script, several relevant processes are started. These include a local scheduler, a global scheduler, an object store and manager, a parameter-server process, and a number of worker processes. Here is an example command to run the script:

```
python parallel_OGLDA.py --num-workers 4
```

Sample of pattern mining results

A set of extracted regular and coherent patterns are provided in "./results" directory of the code. The perplexity measured values are also included in "Gpreplexity_sequential.txt" and "Gpreplexity_parallel.txt" within the code. Note that these results are produced from 1000 csv data files (data_full0.csv … data_full999.csv).

6.2 Online Deep Hierarchical Dynamic Model (DBN-LDA-HMM)

This represents the outcome of D1 and D3. Only the sequential (online) version is provided. It reads utility usage data from CSV files that were created through feature extraction presented in Deliverable 1 and applies deep learning to learn new features.

Requirements

- Python (preferably Python 2.7.12) should be installed.
- The following packages are used: numpy(1.13.3), scipy(1.0.0), csv, glob, datetime, matplotlib, time, re, random, argparse and sys.

- Copy extracted feature files to "./Features" directory. The files should be in CSV format. The code to generate feature data is placed in "./Deep Learning" directory.
- Copy post-processed data files to "./Data" directory. The files should be in CSV format.

Running the code

The runnable script is named as *run_HDM.py*. The script only takes one optional argument --data-dir with to specify the extracted feature directory, default value is "./Features". Here is an example command:

```
python run_HDM.py --data-dir ./Features
```

and

```
python plot_global.py
```

The other runnable script is named as *LeastSquareRegression.py* allowing to show how the algorithm can be used to predict electricity consumption. The script only takes one optional argument --data-dir with to specify the data directory, default value is "./Data". Here is an example command:

```
python LeastSquareRegression.py --data-dir ./Data
```

Sample results

The outcome are the learned components by DBN-LDA-HMM correspond to clusters in DBN's output space and patterns of energy consumption activities (Figure 1 in the "./results" directory) and the computed energy consumption and estimated energy consumption (Figure 2 a & b in the "./results" directory).

6.3 Online Regression Algorithms (OSLOG, COIRR, ONLMS)

The regression algorithms stem from deliverable 4, which deal with hot water usage and gas prediction.

Requirements

- Python (preferably Python 2.7.12) should be installed.
- The following packages are used: numpy(1.13.3), scipy(1.0.0), ray(0.4.0), csv, glob, datetime, matplotlib, time, re, random, argparse and sys.
- Copy data files to "./data" directory. The files should be in CSV format.

Running the code

There are three scripts for running OSLOG, COIRR and ONLMS algorithms in sequential mode: *coirr.py*, *onlmsr.py* and *oslog.py*. Each of these sequential scripts require three arguments. The first argument is the input, that is fed to predict the output. After the prediction is made, the algorithm requires the ground-truth output and the tuning parameter ("self.*a*") which must be strictly positive (>0) for COIRR and OSLOG. Here are example commands to run the sequential version of the algorithms:

```
python coirr.py --data-file data/gas.txt --tuning-parameter 0.5

python oslog.py --data-file data/temp1.txt --tuning-parameter 0.5

python onlmsr.py --data-file data/temp2.txt --tuning-parameter 0.5
```

Each of the aforementioned algorithms can be ran in parallel mode. The script *run_parallel.py* takes 4 arguments as follows:

```
--algorithm (required)
    The name of the algorithm.
    Possible values: "coirr", "oslog", "onlmsr".

--tuning-parameter (optional)
    The value of the tuning parameter.
    Default value = 0.6.

--data-file (required)
    Path of the data file.

--num-workers (optional)
    Number of the workers to use.
```

This code can be run on a single machine to achieve efficient multiprocessing, or it can be used on a cluster for large computations (for using Ray on a large cluster).

When using the script, several relevant processes are started. These include a local scheduler, a global scheduler, an object store and manager, a parameter-server process, and several worker processes. Here is an example command to run the script:

```
python run_parallel.py --algorithm coirr --num-workers 2 --data-file data/gas.txt
```

Sample regression results

COIRR is implemented on Gas data. OSLOG is implemented for Temp1 data and ONLMSR is implemented for Temp2 data. For details on data preparation please see Deliverable 4. Once the user runs *{algorithm_name}.py,* the user will be presented with similar plots as those included in Deliverable 4.

7. Conclusion

In this report, we briefly summarised the tasks of the project. We provided an overview of a set of algorithms developed in the context of HFADA and show how they can be applied. More details are provided in the code itself. It is worthwhile to stress the fact these algorithms are original and are implemented using the same coding approach in terms of distribution. We wished to be able to do more by developing alternative solutions and comparing them, but due to limitation of resources and the short duration of the project with many demanding deliverables, HFADA proved to be challenging.