Residential Household Non-Intrusive Load Monitoring Using Multi-Label Classification Methods

by

Ding Li

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering & Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Smart grid provides a sustainable and environment-friendly vision of the future power systems by incorporating smart meter, remote control and communication technologies into the network of electricity generation, distribution and consumption. Residential Demand-Side Management (RDSM) is an important part of smart grid since a large part of electricity is consumed by residential sector. It refers to programs implemented at the residential customers' side and can be utilized to reduce the overall load demand, most importantly, the peak demand. Since the grid is designed for the peak demand instead of the average demand, the grid is under-utilized for most of the time. By reducing the peak demand or shifting load to off-peak time, the peak-to-average ratio is reduced, grid reliability can be improved, energy can be saved coupled with carbon dioxide and other greenhouse gases emission can be minimized.

However, currently it is impossible for residential customers to identify ways to save electricity. On the one hand, residents have no idea of each share of the total electricity consumed by individual appliances since they only receive an aggregate monthly bill. On the other hand, there are more and more electrical appliances in a household nowadays. In order to solve this problem, Non-Intrusive Appliance Load Monitoring (NIALM)was proposed to disaggregate energy consumption to appliance level or circuit level, specifically, it refers to inferring what appliances are operating and how much energy

they consuming in a household at a given time solely from house-level aggregate measurements from the main panel. It is one approach to residential demand-side management strategies in the Smart Grid and is now commonly implemented via machine learning. However, by and large, the learning-based NIALM algorithms to date have been single-label approaches; each feature vector is associated with a scalar, categorical value (which means there is only a single appliance associates with an instance of aggregate power). This, however, is a poor match to the NIALM problem, in which multiple independent concepts (active appliances) may simultaneously hold true. To model this characteristic, multilabel classification algorithms (which associate a vector of categorical variables to each feature vector) have been employed for NIALM in this thesis. Furthermore, those learning algorithms generally require a ground-truth classification of the observed fluctuations into the set of appliances then operating; data which is not ordinarily available. As a compromise, we examine semi-supervised learning algorithms, which only need a ground truth from a small sample (e.g from an initial "registration" period). Thus this thesis presents multi-label algorithms for NIALM, either supervised or semi-supervised, to recognize the operational states of appliances simultaneously, and based on the states information, appliance energy consumption can be calculated. Extensive experiments have been conducted on five public datasets and comparisons have been made against other methods in the-state-of-the-art literature, which prove that graph-based manifold semi-supervised multi-label classification might be a promising approach to NIALM.

In the end, a reliance weighting strategy is proposed to improve the performance of graph-based manifold multi-label classification. And extensive experiments have been carried out on another four general multi-label datasets.

Preface

- Chapter 2 and chapter 3 have been submitted as "Li, D., & Dick, S. Disaggregating household appliance loads using multi-label classication methods. International Transactions on Electrical Energy Systems." A short version has been published as "Li, D., & Dick, S. (2016, July). Whole-house non-intrusive appliance load monitoring via multi-label classification. In Neural Networks (IJCNN), 2016 International Joint Conference on, Vancouver, BC, 2016, pp. 2749-2755."

- Chapter 4 has been accepted for publication by IEEE Transactions on Smart Grid as "Li, D., & Dick, S., Residential Household Non-Intrusive Load Monitoring via Graph-Based Multi-Label Semi-Supervised Learning. IEEE Transactions on Smart Grid." A short version has been published as "Li, D., & Dick, S. (2017, July). A Graph-Based Semi-Supervised Learning Approach Towards Household Energy Disaggregation. In 2017 IEEE international conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy."

- Chapter 5 has been submitted as "Li, D., & Dick, S. Semi-Supervised Multi-Label Classification Using Graph-Based Manifold Regularization. IEEE Transactions on Pattern Analysis and Machine Intelligence."

- "Li, D., Sawyer, K., & Dick, S. (2015, August). Disaggregating household loads via semi-supervised multi-label classification. In Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American, Redmond, WA, 2015, pp. 1-5." is a preliminary work on semi-supervised learning.

Acknowledgements

I would like to thank China scholarship council for providing me the scholarship. Thanks to the scholarship, I can go abroad, study here and live here. It will be an unforgettable experience in my life. I would like to thank University of Alberta for providing such a good and safe environment. Thanks for all the programs that make our living and studying here much easier.

I would like to express my sincere gratitude to my supervisor, Dr. Scott Dick, for all his efforts and suggestions in helping as well as supervising me for my Ph.D. research. With his guidance, I have learned how to conduct research and write academic articles. I would like thank Dr. Petr Musilek, Dr. Scott Dick, Dr. Witold Pedrycz, Dr. Lukasz Kurgan, Dr. Venkata Dinavahi and Dr. Yasser Abdel-Rady I. Mohamed for the lessons I have learned during the classes. I would like to thank my committee members, Dr. Wilsun Xu, Dr. Petr Musilek, for their comments and suggestions for helping me preparing for the final defense of my Ph.D. degree.

At last, I would like to thank all my friends and family members, especially my parents, my two elder sisters, my baby and my husband for their love and support! My parents have always been working hard and doing their best to support me without reservation since I came to the world, so that I can have all the opportunities in my life and the chance to pursue my Ph.D. degree. I am forever grateful for your love and support. My two elder sisters, have always been there for me and taken good care of me. Meanwhile, I would like to express my special gratitude to my beloved husband for his love, support and friendship! We together built our family, we together saw our family grow with our healthy, sweet, handsome and cute baby, Michael. I could not have

done all of this without you by my side.

I have gained and grown up a lot during the past few years, Thanks again for all your kindness, help and encouragement.

Contents

List of Tables

List of Figures

List of Symbols

$\mathcal{L}$	the number of household appliances
χ_t	the aggregated power load at time instant t
$N(x)$	the set of k nearest neighbors of x
$\overrightarrow{y_x}$	a label vector for x
$\overrightarrow{C_x}$	the label counting vector of x
H_l^1	the event that x belongs to label l
H_l^0	the event that x not having label l
E_l^j	the event that there are j instances among x's neighbors assigned label l
U_l	the set of training instances with the label l
σ_d	the smoothing parameter of Gaussian kernel
μ	a scaling factor
ω	a $(\mathcal{L}+1)$-dimensional vector
R	the observable time measurement
Φ_n	the reconstructed vector
hloss(h)	Hamming loss
Δ	the symmetric difference between two sets
rloss(f)	ranking Loss
Y_i	the true label set for instance $\mathbf{x}_i$
$\overline{Y}_i$	the complementary set of Y_i
one_error(f)	one error
γ	the complete label set for the training dataset

coverage(f)	coverage
avgprec(f)	average precision
tp	the number of true positives
tn	the number of true negatives
fp	the number of false positives
fn	the number of false negatives
$\mathcal{X}$	feature space (aggregated power)
$\mathcal{Y}$	label space (operating appliances)
$\mathcal{L}$	the number of labels (appliances)
$\mathbb{D}, \mathbb{D}_l, \mathbb{D}_u$	complete, labeled, and unlabeled training datasets
$\mathbb{D}_t$	dataset of future observations
N, l, u	the numbers of total, labeled and unlabeled samples
$\mathbf{x}_i, \mathbf{y}_i$	the ith aggregated power sample and label vector
$\mathbf{X}, \mathbf{Y}$	feature and label matrices
$\mathbf{F}$	predicted label matrix
$\mathcal{F}$	domain of $\mathbf{F}$
G, V, E	adjacency graph, nodes, and edges
$\mathbf{W}, \mathbf{U}, \mathbf{H}$	similarity weight, distance, and alternative distance matrices
$\mathbf{L}, \mathbf{\Delta}$	unnormalized and normalized graph Laplacians
$\Gamma(\mathbf{F})$	cost function
$\Gamma_f(\mathbf{F})$	empirical cost function
$\Gamma_s(\mathbf{F})$	graph smoothness cost function
$\hat{\mathbf{Y}}$	estimated label matrix
$\mathbf{K}$	matrix obtained from the kernel function $K(\cdot)$ with respect to all training samples $\mathbf{X}$

List of Acronyms

AMPds	Almanac of Minutely Power Dataset
ANE	Average Normalized Error
BLUED	Building-Level fUlly labeled Electricity Disaggregation dataset
BP	Belief Propagation
BR	Binary-Relevance
DRED	Dutch Residential Energy Dataset
DSM	Demand-Side Management
ECO	Electricity Consumption & Occupancy
FHMM	Factorial Hidden Markov Model
GFHF	Gaussian Field and Harmonic Functions
HMM	Hidden Markov Models
LapRLS	Laplacian Regularized Least Squares
LGC	Local and Global Consistency
LP	Label-Powerset
ML	Multi-Label
ML-BPNN	Multi-Label Back-Propagation Neural Network
ML-FSKSC	Fixed-Size Multi-Label regularized Kernel Spectral Clustering
ML-GFHF	Multi-Label Gaussian Fields and Harmonic Functions
ML-KNN	Multi-Label k Nearest Neighbor
ML-LGC	Multi-Label Local and Global Consistency

ML-MR	Multi-Label Manifold Regularization
ML-MRRW	ML-MR with the Reliance Weighting strategy
ML-RBF	Multi-Label Radial Basis Function
ML-SVM	Multi-Label Support Vector Machine
MR	Manifold Regularization
NIALM	Non-Intrusive Appliance Load Monitoring
RAkEL	RAndom k labELsets
RBF	Radial Basis Function
REDD	Reference Energy Disaggregation Dataset
RKHS	Reproducing Kernel Hilbert Space
RLMPs	Residential Load Management Programs
RMS	Root Mean Square
RW	Reliance Weighting
SSL	Semi-Supervised Learning
SSML	Semi-Supervised learning for Multiple Labels
SVM	Support Vector Machine
TPR	True Positive Rate
TRECVID	TREC VIDeo retrieval evaluation

Chapter 1

Introduction

1.1 Background

Electricity demand usually varies significantly with the time of day and human activities. As a consequence, utilities must devote extensive resources to providing reserve power generation to handle the peak demand in a day. Literally billions of dollars worth of boilers, turbines and generators thus sit idle - or worse, are operating and producing greenhouse gases without being connected to the transmission grid - awaiting the moment when demand is high enough to bring them on-stream. Thus, even minor reductions in peak demand may imply significant energy savings for both consumers and utilities [114, 127, 150]. In order to reduce this waste of resources, Demand-Side Management (DSM) has been proposed to shape customer demand patterns (especially peak shaving) to allow more efficient utilization of power system assets, improve grid reliability, and reduce emissions [117]. DSM has been practiced since the 1980s [57, 93], and continues to be a critical element of the Smart Grid. Within the DSM umbrella, Residential Load Management Programs (RLMPs) focus on residential rather than industrial customers; this class accounts for over 30% of total power consumption [1], and so must be a part of any practical DSM solutions.

A significant reduction of energy wastage in the residential sector is possible via providing fine-grained energy consumption details [36]. By providing customers with itemized, appliance-specific tariffs instead of an aggregated monthly bill, we can raise awareness of which appliances consume the most electricity. Older units may be repaired or replaced with more efficient ones. Consumers can time portions of their energy usage to take advantage of lower

off-peak pricing when it is offered. Energy management systems could devise load scheduling schemes to optimize energy generation and utilization. Moreover, a recommender system could be produced to help customers modify their behaviors to cut down electricity bills and conserve energy. All of these behaviors are reasonable responses to a more detailed price signal. Plainly, however, *obtaining* those detailed tariffs first requires us to identify the times when an appliance is operating, and how much power was consumed during those times. The large-scale deployment of smart meters in the UK and the USA, for instance, is driven by the vision of time-of-use pricing on a national scale [36]. Modern networked appliances can automatically communicate this information to a utility; older devices, however, cannot. Thus, a mechanism for obtaining the needed consumption data is required.

One straightforward method for gathering the necessary data is to install sensors on each appliance within a household, and transmit that data to the utility [129]. Needless to say, this is a significant invasion of privacy (hence untenable for most North American customers) [48], and would impose significant operating costs on the utility as well. An alternative is thus needed for households unable or unwilling to be so connected; Hart's Non-Intrusive Appliance Load Monitoring (NIALM) approach [62] is a well-known candidate. The set of appliances operating in a household at a given time is inferred from fluctuations on the main power feeder only; this is an instance of the general blind-source separation problem in signal processing [67].

The task of Non-Intrusive Appliance Load Monitoring (NIALM) is to disaggregate the total power consumption at the meter into the individual appliance draws based on their energy consumption features (*load signatures*). The fundamental assumption are that appliances have characteristic power demand patterns, which are usually discriminative, and they can be identified in the aggregate household power signal. In practice, the power signal (it usually means aggregate current and voltage on each phase) will be monitored by a sensor at the house-level power meter (like the nation-wide deployment of smart meters in the UK and US [36]). This information is then sent to the utility in real-time. No sensors need to be placed on appliances within the home. Data volumes are relatively small (only a single sensor per household), which also minimizes networking and data storage costs. (Note that recent proposals suggest that for practical NIALM, a one-time training data collec-

tion process for each individual appliance may be needed [45].) Subsequently, the utility will need to disaggregate appliance-specific power draws from the aggregated power signal; machine learning and data mining techniques are frequently used for this purpose. Surveys in [150, 168, 20] and most recently [134] have noted the wide variety of learning algorithms that have been used in NIALM. An open source toolkit integrating many popular algorithms was developed in [14].

1.2 Problems and Motivations

Non-intrusive appliance load monitoring is a technique to help power companies monitor and analyze residential energy usage. Aggregated power load measurements for a household (i.e. the signal on the main powerline) are disaggregated into individual appliance loads by examining the appliance-specific power consumption characteristics. This data can then be used to modify consumer behaviors via detailed billing and/or demand-pricing tariffs. A number of advances in the field have been reported in the past two decades, many of which apply machine learning algorithms. However, a systematic review in [134] found that, by and large, the learning-based NIALM algorithms to date have been single-label approaches; each feature vector is associated with a scalar, categorical value. This, however, is a poor match to the NIALM problem, in which multiple independent concepts (active appliances) may simultaneously hold true. To model this characteristic, multi-label classification algorithms (which associate a *vector* of categorical variables to each feature vector) have been introduced for NIALM in [12, 13, 134]. However, this research work is still in its early stages. At this time, most of the results for multi-label classification come from implementations of the *label powerset* approach. This is a meta-classification technique, in which each element of the power set of labels for a dataset is associated with a new, scalar label. Existing single-label classifiers can then be used to model the dataset. There has been very little work exploring bespoke multi-label algorithms for NIALM [12, 13, 86]. Furthermore, only a limited set of high-power appliances are disaggregated.

A large number of statistical and machine-learning algorithms have been applied to the NIALM problem (see [134] for a recent survey), but all encounter significant problems in real-world deployment. Supervised learning

algorithms tend to be very accurate, even for large numbers of appliances in a house, but require a *ground truth* (a general term for the actual state of the world, e.g., exactly which appliances are operating at any given sample instant, in addition to the observed signal on the main feeder). Unsupervised learning algorithms do not require this information, and the class of Factorial Hidden Markov Models (FHMMs) can in theory be very accurate. However, unsupervised solutions to blind-source separation can only be accurate up to permutation [65]; an appliance may be detected, but there is no way for an FHMM (for example) to decide if this is a washing machine or television. Additional ground-truth information is needed to make that determination. Even then, the performance of FHMM algorithms degrades as the number of appliances rises [149]. Furthermore, the NIALM problem does not map well to standard classification algorithms where the dependent variable is a scalar [13, 134]. While some appliances often operate together, the overall household is best considered a collection of independent loads, and thus the output of NIALM should be a vector. Finally, current smart metering infrastructures (which have recently been rolled out at considerable expense in e.g. Europe, North America and Australia) have relatively long sampling intervals, on the order of 10-60 seconds [67]. Taken together, these requirements indicate that a practical NIALM algorithm must minimize the ground-truth data required, remain accurate even for large numbers of appliances, and model the simultaneous operations of an arbitrary number of appliances in a household, accomplishing all of this with a data sampling rate no faster than tens of seconds.

Considering the above requirements, we are proposing a *low-rate* NIALM algorithm based on Semi-Supervised Multi-Label (SSML) classification algorithms. *Semi-supervised* algorithms learn from a small number of *labeled examples* (those having a classification label from the ground truth appended) joined with a large number of unlabeled ones; this is a typical approach to machine learning in domains where labeled examples are difficult or expensive to acquire [166, 161]. *Multi-label classifiers* associate a feature vector with a vector of labels, each representing some separate concept; this appears to be a closer match to the NIALM problem. In this paper, we develop three new SSML graph signal processing algorithms, and apply them to NIALM. Experiments on five benchmark NIALM datasets show that our approach can

outperform state-of-the-art results reported in the literature for each one.

1.3 Objectives and Contributions

The goal of this dissertation is to develop multi-label non-intrusive load monitoring approaches for residential households, either supervised or semi-supervised, to identify real time operational states of individual appliances from house-level measurements (aggregate data) from the main panel. And calculate each share of electricity consumed by individual appliances. The main focus is to develop a low cost and applicable installation of NIALM for residential households.

The major contributions in this thesis that distinguish it from other work are summarized as follows:

- The first empirical evaluation of bespoke multi-label classifiers for NIALM across multiple benchmark datasets, in a supervised way.
- A demonstration that the "best" of these algorithms is competitive with or superior to existing results on those datasets.
- A semi-supervised multi-label disaggregation framework for NIALM, including graph-based Local and Global Consistency (LGC), Gaussian Field and Harmonic Functions (GFHF) and Manifold Regularization (MR).
- Integration of the Multi-Label k Nearest Neighbor (ML-KNN) algorithm in LGC and GFHF to extend the methods from transductive learning to inductive learning.
- Extended manifold regularization to multi-label classification with a reliance weighting strategy to improve the classification performance.
- Extensive experiments on public datasets to compare the performances using different methods and demonstrate the efficiency of the proposed methods.

1.4 Thesis Outline

The remainder of the thesis is organized as follows:

Chapter 2 presents an overview on non-intrusive appliance load monitoring, including a brief overview, the basics of NIALM which contains household appliance types and general steps of the NIALM, literature review of NIALM which contains features used for NIALM and load disaggregation algorithms, public data sets for NIALM are introduced in the end.

Chapter 3 presents load disaggregation via multi-label classification, including an overview, problem formulation, four bespoke multi-label classification methods which contains four bespoke multi-label algorithms(ML-KNN, ML-RBF, ML-BPNN and ML-SVM), performance metrics, experimental design which contains filling small gaps, labeling, downsampling and delay embedding, experimental results and summaries are given at last.

Chapter 4 presents semi-supervised learning for non-intrusive load monitoring, it firstly gives a brief overview and presents the problem formulation, including load disaggregation via semi-supervised multi-label classifiers and framework of the method, then it introduces graph-based semi-supervised learning for multiple labels, including graph construction, graph Laplacian regularization and manifold regularization, next, it presents the experimental methodology, including data preparation, experimental design and performance metrics, then, it demonstrates the experimental results and discussions, including five case studies, summaries are given at the end.

Chapter 5 presents manifold regularization incorporated with a reliance weighting strategy for multi-labels. Including a brief overview, the background, preliminaries on manifold regularization which contains basics and notations, regularization in Reproducing Kernel Hilbert Space and Manifold regularization, the proposed method which includes graph construction, manifold regularization with multiple labels and reliance weighting strategy for performance improvement, experimental design which includes datasets, experimental setup and performance metrics and experimental results which demonstrates four case studies and discussion. Summaries are given in the end.

Chapter 6 concludes the research work and gives the directions for future work.

Chapter 2

Review on the Non-Intrusive Appliance Load Monitoring

2.1 Overview

Non-intrusive appliance load monitoring, refers to a technique using artificial intelligence to infer the operational states as well as the energy consumption of individual appliances from a single or limited measurements of the households. This technique will provide a lot of benefits (environment sustainability, energy sustainability, grid stability and so on) to all kinds of respects for the whole society. In this chapter, general information including the basics, the household appliance types and the steps of Non-intrusive appliance load monitoring is given. Then a comprehensive literature review of features and algorithms used in NIALM is introduced. Lastly, several public benchmark data sets are presented.

2.2 Basics of the NIALM

Non-Intrusive Appliance Load Monitoring (NIALM), which is also called energy disaggregation, was introduced by Hart in the early 1980s [62, 60, 61]. The goal of NIALM is to infer energy consumption and operational states of individual appliances from an aggregated power signal. The fundamental assumptions in NIALM are that appliances have unique characteristic power demand patterns, which is also called appliance signature, and that the aggregate household power demand is a mixture of them. The objective of NIALM is thus to train models or use inference techniques to recognize which combi-

nations of appliances are operating at a given instant, given the time history of the aggregate power. This history might consist of transient features (e.g. current and voltage harmonics) and/or steady-state features (e.g. active and reactive power, total harmonic distortion) [134, 168].

2.2.1 Household Appliance Types

In a modern household, there exists a large number of electrical appliances with different functions. Like kitchen stoves for cooking, bulbs for lighting and washing machines for cleaning as well as more and more electronic products (phone, computer) for entertaining. Different appliances show different electrical characteristics. Hart categorized appliances into 4 categories, namely, two-state appliance, multi-state appliance, continuously-varying appliance and always-on appliance.

- Two-state appliance: appliances that only have two states, either on or off, and they consume a constant power when they are run and no power when in off state, e.g. bulbs and kettle.
- Multi-state appliance: appliances that have finite states, and they consume different more or less constant power when in different modes, like washing machine that consumes differently in different procedure of water-fill, immerse, rinse, spin and dry operations.
- Continuously-varying appliance: appliances like refrigerator whose power consumption varies continuously during operation.
- Always-on appliance: appliances that are continuously run and consuming the same amount of power, like alarm monitoring.

Fig. 2.1 gives examples of the power characteristics of the first three categories of appliances. In Hart's technical report, two-state appliances were disaggregated quite well with high accuracy, the difficulty lay in how to disaggregate multi-state appliances [61]. Later on, other researcher found out that continuously-varying appliance and always-on appliance are difficult to disaggregate as well.

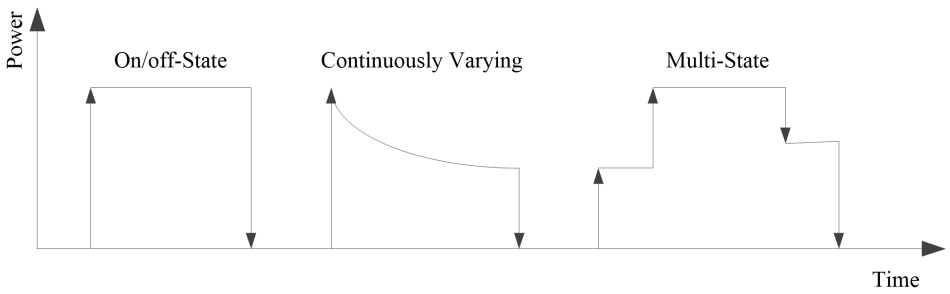

Figure 2.1: Power curves of three types of appliances.

2.2.2 General Steps of the NIALM

The objective of NIALM is to automatically identify the operational states and energy consumption of individual appliances from a single measurement (smart meter) with data mining techniques. There are three steps to achieve this goal, including data acquisition, feature extraction, and inference and disaggregation.

- Data acquisition: aggregated power data is measured and recorded from the main panel with specific sampling rate. The electrical quantities and the sampling rate are determined by features used in the second step. Usually, the aggregated current, voltage, real power and reactive power are measured on each phase. Steady state features (active power, reactive power,) can be extracted with low sampling rate. Transient features (harmonics, current wave, start-up ad turn-off transients) can be obtained with high sampling rate.

- Feature extraction: from the beginning, P-Q plane of active power and reactive power was used to distinguish two-state appliances which have resistive, capacitive and inductive characteristics. Later on, more and more other features are used to characterize the signatures of appliances. e.g. steady state features like peak and root mean square values of voltage, transient features like start-up transients of current,active power and reactive power, and non-electrical features like seasons, time of the day and on duration distribution.

- Inference and disaggregation: the core of NIALM is using machine learning techniques to detect signatures of appliances in the aggregated data thus to identify appliances. The algorithms can be supervised like pat-

tern recognition methods (K nearest neighbor algorithm, support vector machine), or unsupervised like Factorial hidden Markov model, or semi-supervised algorithm like graph-based manifold regularization.

2.3 Literature Review of the NIALM

This subsection presents a literature review of the NIALM from two perspectives, namely, the features used for the NIALM and the load disaggregation algorithms.

2.3.1 Features used for the NIALM

The fundamental assumptions in NIALM are that appliances have characteristic power demand patterns, and that the aggregate household power demand is a mixture of them. The objective of NIALM is thus to train models that recognize which combinations of appliances are operating at a given instant, given the time history of the aggregate power. This history might consist of transient features (e.g. current and voltage harmonics) and/or steady-state features (e.g. active and reactive power, total harmonic distortion) [134, 168]. The steady state signatures include steady active power, reactive power, voltage, current, total harmonic distortion, power factor and voltage-current trajectory and so on. Usually the steady state signatures take with low frequency and over long period. Transient signatures include start-up or turn-off transient of active power, reactive power, current and voltage, transient duration, harmonics and so on. These features require high frequency sampling rate up to kHz or even higher, thus are difficult and expensive to gather. But they can be incorporated to identify appliances with the similar steady-state signatures. Besides, electric irrelevant signatures such as the weather, season, occupancy and temporal information have also been used for load disaggregation [168].

In [53, 116, 106], peak and Root Mean Square (RMS) current and voltage were extracted as signatures to distinguish appliances. RMS was found to be more discriminative than peak values. In [106, 128], researchers have combined harmonics with real and reactive power to improve disaggregation performance. In [85, 83], V-I trajectory with normalized current and voltage values was creatively proposed to cluster appliances into 8 groups and then

sub-partition within each group, this approach was proved to be more effective than many other existing approaches using power measurements. Transient signatures to date also develops fast. Researcher in [30] showed that energy calculated during the starting-up transient could be an effective signature to discriminate appliances. Power spikes and overshoots were tried as features in [34] to decompose loads. A very untraditional and novel method was presented in [145] to discriminate appliances without pre-training and supervision, the author used two basic units, namely rectangles and triangles, as features to identify appliances. Triangle unit was described by starting time, peak time, peak value and ending time while rectangle was described by starting time, peak time, peak value, steady time and steady power. The identification accuracy was 80%. Context-based information like the location of the house residents, house temperature, duration and time of appliance usage have been utilized in [78]. Although the disaggregation performance of NIALM can be improved by these additional features, however, it incurs extra cost and installation of these sensors.

2.3.2 Load Disaggregation Algorithms

Currently, machine learning approaches, whether supervised or unsupervised, dominate the NIALM literature [134]. There are three types of supervised learning in NIALM, all of which require a ground truth associated with each observation made: 1) The optimization based methods construct a database of appliance power draw characteristics for a house, and then identify a subset of operating appliances that minimizes the residual between the observed and expected aggregate power draw [91]. 2) The second type builds appliance models for the target house and disaggregates the corresponding aggregate power with these models [91, 3, 128, 20, 86, 87]. 3) The last type builds a database comprising of general models of different appliances and uses the general models to disaggregate power for unknown households [3, 13].

In contrast, unsupervised NIALM models require no ground truth for individual observations (but see below for a caveat). In this case, NIALM closely resembles the classic Blind-Source Separation (BSS) problem from signal processing. An m-dimensional vector of signals $\mathbf{S}$ are mixed together as $\mathbf{X} = \mathbf{AS}$, where $\mathbf{A}$ is a nonsingular square m-dimensional matrix. The BSS problem is to reconstruct $\mathbf{S}$ given only the mixed signals $\mathbf{X}$. NIALM is specifically

the under-determined case, where $\mathbf{X}$ is a scalar, but $\mathbf{S}$ remains a higher-dimensional vector [67]. A large number of solutions to the BSS problem have been proposed, with Independent Components Analysis prominent amongst them. A key limitation to note is that BSS can only be solved (the vector $\mathbf{S}$ recovered) up to permutation; it is not possible to recover the ordering of components in $\mathbf{S}$ [66]. In NIALM this corresponds to not knowing what physical machine matches with which disaggregated component.

The Factorial Hidden Markov model (FHMM) is currently the most successful and popular unsupervised approach in NIALM [50]. Hidden Markov Models (HMM) can be expressed as a graphical model with two layers. One layer is composed of the hidden state sequences $(s_t)_{t=1}^T$ and the sequences form a Markov chain. The other layer consists of the observations $(o_t)_{t=1}^T$. States change according to the transition probabilities $\pi_{ij} = p(s_{t+1} = i|s_t = j)$ and $s_t \in \{1, 2, \cdots, n\}$. $\pi = (\pi_{i,j})_{i,j=1}^n$ for hidden states i, j. HMMs assume that transitions are completely independent of all other variables except for the current observation. In general, the probability of observing o_t at state s_t is $p(o_t|s_t) = p(s_1)p(o_1|s_1)\prod_{t=2}^T p(s_t|s_{t-1})p(o_t|s_t))$. Factorial hidden Markov models (FHMM) are an extension of HMM where multiple HMMs evolve independently and in parallel, with the observation being the joint function of all the hidden states. Four variants of FHMM were adapted for NIALM in [78]. In addition to the observed aggregate power, exogenous variables including time of day and the dwell time for each appliance state were incorporated in the models. Additive FHMM and differential FHMM [79] are combined to model the aggregate power and the difference between the successive outputs. The observation is the sum of all the HMMs, with the constraint that only one appliance can change state at a time. A hierarchical HMM based approach were proposed in [109], where each appliance was modeled as an HMM and then aggregated by another HMM. Many other FHMM based NIALM techniques have been proposed in e.g. [110, 97]. However, as above, the separate appliances detected by an HMM must still be mapped to actual appliances in the home - which requires at least some ground-truth information. Furthermore, the accuracy of HMMs has been observed to degrade as the number of appliances being modeled increases [149].

Ground-truth information in NIALM is scarce, but a small amount can likely be collected for each house. A registration period, during which res-

idents operate individual appliances for a brief time, was proposed in [55]. Based on the review in [36], consumers seem willing to put forward a small amount of effort if provided with a modest inducement (e.g. a discounted tarrif). Thus, NIALM data for a house potentially consists of a large volume of unlabeled data, augmented by a small number of labeled observations, leading to our proposed use of semi-supervised learning. To the best of our knowledge, there is little existing literature using semi-supervised algorithms for NIALM. References [90] and [88] present our preliminary results of applying semi-supervised algorithms in NIALM. Note also that our labeled examples include no appliance mixtures; in field use we do not expect the consumer to be willing to operate numerous appliance combinations during this registration period.

2.4 Public Datasets for the NIALM

The development and public release of benchmark NIALM datasets, gathered from field measurements in real-world households, has spurred further work in the area. Table 2.1 summarizes the basic information of each dataset.

Table 2.1: Description of public datasets

Dataset	Location	Num. of appliances/circuits	Aggregate sampling period	Appliance sampling period	Duration
House1	MA, USA	20	1 sec	3 sec	36 days
House3	MA, USA	24	1 sec	3 sec	45 days
Blued	PA, USA	43	1/12,000 sec	-	1 week
Smart*	MA, USA	26	1 sec	1 sec	3 months
AMPds	BC, Canada	21	1 min	1 min	1 year
ECO	Swiss	6-10	1 Hz	1Hz	8 months
DRED	Netherlands	12	1 Hz	1Hz	2 months

The Reference Energy Disaggregation Dataset (REDD) [80] measures both house-level and circuit/appliance level data from 6 US households over various durations, ranging from a few weeks to a few months. The data from two houses, specifically house 1 and house 3, have been selected for experiments in this study, because the two datasets covered longer time periods compared with those from other houses; we focus on analyzing the low frequency data

for each house. The Circuit/device level data allows us to label the power waveform with the appliances in use at each sample instant.

The Building-Level fUlly labeled Electricity Disaggregation dataset (BLUED) [4] records house-level data of a single US household for a whole week. Current and voltage measurements of the mains are sampled at high frequency (12 kHz). A time-stamped list of every operational state transition of the individual appliances in the home is kept during the measurement period, forming the ground truth for this dataset.

The UMASS Smart* Home Data Set (Smart*) [7] gathers a variety of electrical and environmental data as well as operational data of switches for three households in Western Massachusetts. Experiments are only carried out on household A since it contains both aggregate and sub-metered electrical data and it is the most thoroughly instrumented home. The average active and apparent power of the mains and individual circuits are sampled at intervals of one second, and the active power of each plug load is sampled every few seconds. Moreover, on-off events of all of the house's wall switches are recorded. There are 26 individual circuits from the electrical panel.

The Almanac of Minutely Power Dataset (AMPds) [98] records data of the mains and 21 individual circuits from a household in the great Vancouver area, BC, Canada, over a period of one year. Each reading includes measurements of voltage, current, active power, reactive power, apparent power, power factor and frequency sampled at 1 minute intervals.

The Electricity Consumption & Occupancy (ECO) [16] data set collected aggregate data as well as 6-10 selected plug-level data for ground truth from 6 households in Swiss for 8 months. For each household, the aggregate data (voltage,current and phase shift between voltage and current for each phase) and the plug-level data were sampled at 1 Hz frequency. Besides, the occupancy information was manually entered into a tablet by residents and monitored by a passive infrared sensor.

Dutch Residential Energy Dataset (DRED)[142] provides house level and appliance level electricity consumption data at 1 Hz frequency for a single house in the Netherlands. Besides, it collects ambient information (room level indoor temperature, outside temperature, wind speed, precipitation and humidity), occupancy data (room-level information of occupants) as well as household information (house layout, number of residents, appliance-location

mapping). This dataset is useful for algorithms using context-based features (occupancy,temperature, number of residents) for NIALM.

Many other public datasets are available as well, like COOLL [112], BERD-S [94] and Tracebase [115].

Chapter 3

Load Disaggregation via Multi-Label Classification

3.1 Overview

This section explores the applications of bespoke multi-label classification algorithms in NIALM and conduct experiments to compare their performances. Four existing bespoke multi-label classification algorithms (ML-KNN, ML-RBF, ML-BPNN, ML-SVM) have been applied to disaggregate several public data sets. The power signal of each individual household is disaggregated into all of the sub-metered appliances that are available, and the experimental results are compared to the existing literature on these datasets. Two of the algorithms are found consistently superior to the other two, with the consistently "best" generally outperforming all other approaches that have been attempted on these data sets. The remainder of this section is organized as follows. In Section 3.2, the problem of load disaggregation is formulated, and machine-learning approaches to NIALM are reviewed. Multi-label classification algorithms are then reviewed in Section 3.3. We discuss the performance metrics and our experimental design in Sections 3.4 and 3.5. Experimental results are presented in Section 3.6. A summary is offered in the last subsection.

3.2 Problem Formulation

The objective of NIALM is to disaggregate the total power demand of a household into its major constituents, and identify the household appliances in

use. The general problem of NIALM can be briefly described as follows. Given a labeled database of household power loads, a mathematical model representing the relations between features and appliances is trained first. Then, the unknown aggregated power load is broken down into a set of constituents that are used to identify appliances in use based on the trained model. According to [62, 91], the general NIALM problem can be formulated as

$$\min_{\{\hat{a}_{i,t}:i=1,\cdots,\mathcal{L}\}} \epsilon = h_t\left(\sum_{i=1}^{\mathcal{L}}(x_i \times \hat{a}_{i,t}), \chi_t\right) \tag{3.1}$$

where $\mathcal{L}$ denotes the number of household appliances, χ_t is the aggregated power load at time instant t, x_i is the load signature of the ith appliance, $\hat{a}_{i,t}$ is the estimated operating status of the ith appliance at time t, formulated as

$$\hat{a}_{i,t} = \begin{cases} 1, & \text{if the } i\text{th appliance is on at } t \\ 0, & \text{if the } i\text{th appliance is off at } t \end{cases} \tag{3.2}$$

and h_t is a loss function.

Thus the NIALM problem is to identify the status $\hat{a}_{i,t}$ of appliances from the unknown dataset χ_t based on the known features x_i. Figure 3.1 presents an example of aggregated power load of four different appliances. In this case, the basic NIALM problem is to determine $\hat{a}_{1,t}, \hat{a}_{2,t}, \hat{a}_{3,t}$, and $\hat{a}_{4,t}$ from the observed power signal, using the known load signatures x_1, x_2, x_3, and x_4. In the machine-learning variant of NIALM, we do not have access to the load signatures x_i. Instead, we extract features from the observed power signal, use them to inductively learn appliance load models, and then use those models to predict the operating statuses $\hat{a}_{i,t}$.

One broad strategy for solving the NIALM problem is to derive a set of descriptive features from the power waveform at the meter, and use machine learning to relate these features to a subset of the appliances in a household that are operating at that sample instant. The review in [134] discovered literally hundreds of approaches following this pattern. The features ranged from the active power signal [45, 160, 113], to current harmonics [96, 101, 128, 146], and the V-I trajectory [63, 64]. Other studies employ transformations of the power signal including the discrete wavelet transform [51, 124], the Stockwell transform [24, 74, 92], and the short-time Fourier transform [70, 73, 72]. Side channels (correlates of appliance usage) are used in [2, 136, 137, 148]. There are a number of other feature sets, but they do not appear to be frequently used.

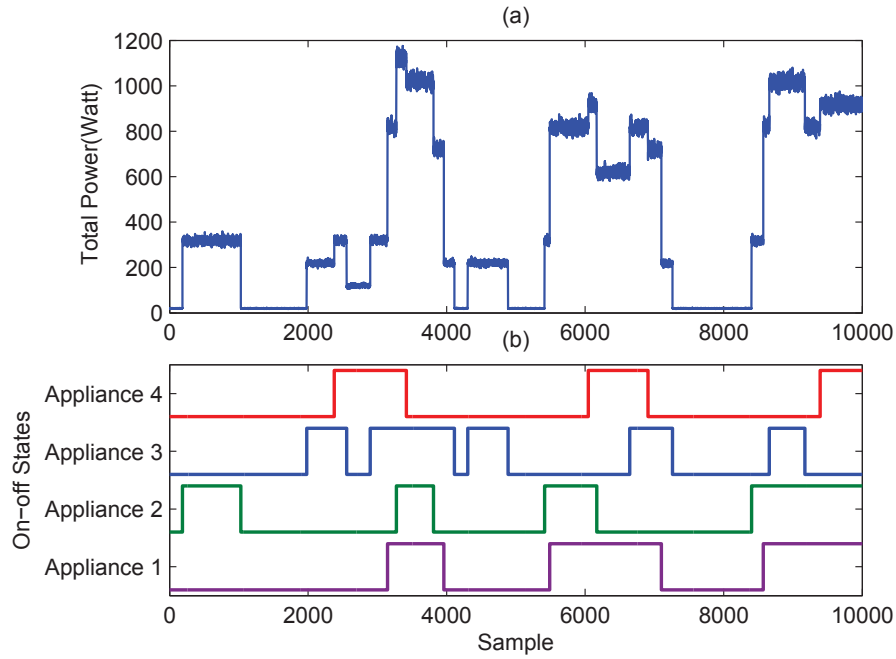

Figure 3.1: (a)Aggregated power load measured from the meter at the entry point; (b)The on-off states of four unique household appliances.

The learning algorithms in use can broadly be classified into supervised, unsupervised, and semi-supervised approaches, which are differentiated by whether they require that a ground truth be known for each observation. Supervised approaches (those that require ground truth for every input example) include parametric statistical approaches (e.g. [81]), and (much more commonly) non-parametric approaches (i.e. those that do not assume a particular probability distribution for the data). Commonly-used approaches include the K-Nearest-Neighbor algorithm [13, 15, 52], neural networks [28, 29], recently including deep learning algorithms [77, 100], and support vector machines [73, 74, 82]. A number of researchers attempt to inductively build an appliance signature database, e.g. [84, 101]. Unsupervised approaches, on the other hand, do not need a ground truth for any examples. Clustering [40, 44], and certain types of Bayesian inference [98, 147, 160], are unsupervised methods, but the most frequently used unsupervised approach seems to be hidden Markov models (particularly the Factorial Hidden Markov Model (FHMM)), and their variants [79, 2, 46, 59]. FHMM models, in particular, have proven to be very accurate in NIALM. Finally, semi-supervised methods only require a ground truth for a small subset of all the examples in a training set [16, 90].

Side channel information can sometimes be used as a proxy for the ground truth [148].

Most studies on NIALM are event based. The techniques work by detecting on/off transitions, namely the edges shown in Figure 3.1-(a). This requires the assumption that events do not occur simultaneously. However, in reality, multiple appliances may operate simultaneously, such as shown in Figure 3.1. By contrast, the non-event based techniques only detect whether appliances are on over a specific time period [13]. Thus, this requires the classifier to have the ability to identify multiple appliances in use simultaneously. A class of algorithms that possesses this property, known as multi-label classifiers, can be formulated as follows [154].

Let $\mathbf{x}$ denote a vector of aggregated power samples in a d-dimensional space χ and $\gamma = \{1, 2, \cdots, \mathcal{L}\}$ be the appliances, namely, the multiple labels. Given a training dataset $D = \{(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2), \cdots, (\mathbf{x}_n, Y_n)\}(\mathbf{x}_i \in \chi, Y_i \in \gamma)$, the goal of the learning system is to produce a multi-label classifier $h : \chi \to 2^{\gamma}$ by optimizing a specific performance metric. In other cases, the learning system will output a real-valued function $f : \chi \times \gamma \to \mathcal{R}$. The assumption is that, given an instance $(\mathbf{x}_i, Y_i)$, a successful learning system will output larger values for labels in Y_i than those not in Y_i, namely, $f(\mathbf{x}_i, y_i) > f(\mathbf{x}_i, y_j), \forall y_i \in Y_i \wedge y_j \notin Y_i (i + j = \mathcal{L})$.

The real-valued function $f : \chi \times \gamma \to \mathcal{R}$ can also be transformed to a ranking function Rank(f) by mapping the outputs of $f(\mathbf{x}_i, y_i)$ to $\{1, 2, \cdots, \mathcal{L}\}$. It is assumed that if $f(\mathbf{x}_i, y_i) > f(\mathbf{x}_i, y_j)$, then Rank($f(\mathbf{x}_i, y_i)$)< Rank($f(\mathbf{x}_i, y_j)$). The multi-label classifier $h(\cdot)$ can also be obtained through $f(\mathbf{x}_i, y_i)$, i.e., $h(\mathbf{x}_i) = \{Y_i | f(\mathbf{x}_i, y_i) > t(\mathbf{x}_i)\}$, where $t(\cdot)$ is a threshold which is usually set to zero.

3.3 Multi-Label Classification Methods

In the Multi-Label (ML) classification problem, we assume that a given learning problem contains multiple separate concepts that must simultaneously be modeled. These concepts, while distinct, might not be independent in the statistical sense. For instance, the microwave and the stove in a kitchen are separate appliances, with no causal link between the operation of either one. However, their *usage* might well not be independent, as many people cook meals using the stove and microwave simultaneously (using each for the

dishes they are best suited for). From the machine-learning point of view, each appliance is a concept, which we assume means it is associated with a (preferably convex) region of the feature space for the problem. The particular difficulty of the ML problem is that the feature-space regions for different concepts may overlap heavily. Hence the need for ML algorithms instead of single-label classification; the latter implicitly assumes that overlaps between classes are fairly minor [155], [126].

Many ML algorithms have been proposed in the literature. These methods can be categorized into two groups, namely problem transformation and algorithm adaptation [126]. Problem transformation is an indirect solution to the ML task, as it transforms the learning problem into one or more single label classification problem. The Binary-Relevance (BR) and Label-Powerset (LP) methods are problem transformation approaches [126]. As previously discussed, LP methods create a new multi-class single-label problem by assigning a new label to each element of the power set of appliance labels. Accordingly, this problem can then be solved using any multi-class learning algorithm [13]. The RAndom k labELsets (RAkEL) algorithm is a typical LP method, which has been integrated into a multi-label learning tool named MULAN [139, 140]. The method is believed to improve substantially over the basic label powerset approach, especially with a large number of labels, and performs better than other multilabel learning methods [139]. However, a disadvantage of LP methods is the relatively high computational demands [13]. The BR approach creates a single binary classifier for each label of the ML problem, and then combines them all into an ensemble [155, 126]. The BR method has been used as the building block of many state-of-the-art multi-label learning algorithms [155]. However, it has also been criticized for its implicit assumption of label independence, which might not hold in a given dataset [126]. Both the LP and BR methods have been used to solve the multi-label load disaggregation problem in [13], while an LP method (RAkEL) was also employed in [134]. Algorithm adaptation is a direct solution to the ML task, in which bespoke ML algorithms are created (often by modifying known single-label algorithms) [155].

In [10], the BR and LP methods (both using decision trees as the base classifier; the LP method is RAkEL), plus a second BR instance using boosting, were compared against each other, ML-KNN, and a single-label meta-classifier

for NIALM, using features of the active power signal sampled every 10 minutes. In testing on three high-power appliances across 100 houses from the IRISE database [12], the LP method was slightly superior to the others. [9] expands on this study, testing eight appliances and the sum of all lighting in each house, adding side-channel information, but reducing the data sampling rate to once per hour. Again, the LP approach was slightly superior. In [11], LP and BR classifiers are used to forecast *future* usage of four high-power appliances, using features extracted from the active power signal sampled at 1-hour intervals. Both approaches performed approximately equally well. [12] divides those 100 houses into three categories, with different numbers of high-power appliances disaggregated in each (between three and six). BR and LP are explored, with decision trees and support vector machines as the underlying classifiers. These are comapred against ML-KNN, and Hidden Markov Models. The "best" approach seemed to vary between categories of houses.

3.3.1 ML-KNN

The ML-KNN [154] is adapted from the well-known single-label k-nearest neighbor algorithm. The rationale behind this algorithm is that a new instance's labels should be strongly correlated with the labels assigned to its neighboring data points. Given an unclassified instance x, its k nearest neighbors $N(x)$ in the training dataset are first identified, and the number of neighbors having each label is counted. Let $\overrightarrow{y_x}$ denotes a label vector for x. $\overrightarrow{C_x}$ represents the label counting vector of x, and its lth element $C_x(l)$ sums up the number of x's neighbors having label l. Then the label set of the instance is determined through maximizing *a posteriori* principle $\overrightarrow{y_x}(l) = \arg\max_{b\in\{0,1\}} P\left(H_l^b | E_l^{\overrightarrow{C_x}(l)}\right)$. where H_l^1 is the event that x belongs to label l and H_l^0 is the event that x not having label l. $E_l^j (j \in 0, 1, \cdots, K)$ denotes the event that there are j instances among x's neighbors assigned label l.

3.3.2 ML-RBF

ML-RBF is a three-layer neural network [152], much like the familiar Radial Basis Function Network (RBFN). The input layer of identity neurons is fully connected to the hidden layer of RBF neurons, which in turn is fully connected to the output layer, where each output neuron corresponds to a la-

bel. There are two stages for training. K-means clustering is performed in the first layer; $U_l = \{(\mathbf{x}_i, Y_i), l \in Y_i\}$ denotes the set of training instances with the label $l \in \{1, 2, \cdots, \mathcal{L}\}$. $k_l = \beta \times |U_l|$ clusters are formed for each label, with $0 < \beta < 1$ limiting the number of clusters for that label to less than the number of examples. A total of $K = \sum_{l=1}^{\mathcal{L}} k_l$ clusters are formed, each corresponding to one neuron in the hidden layer. Weights are learned in the second step by minimizing the sum of squared error $E = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{\mathcal{L}}\left((y_l(\mathbf{x}_i) - t_l^i\right)^2$, where t_l^i is the true value of the ith label, $y_l(\mathbf{x}_i) = \sum_{d=0}^{K} w_{dl}\phi_d(\mathbf{x}_i)$ is the activation of the ith output neuron (label). The radial basis neurons use the Gaussian activation function $\phi_d(\mathbf{x}_i) = \exp\left(-\frac{\text{dist}(\mathbf{x}_i, c_d)^2}{2\sigma_d^2}\right)$, where $\text{dist}(\mathbf{x}_i, c_d)$ is the Euclidean distance between $\mathbf{x}_i$ and the dth prototype vector c_d. The smoothing parameter σ_d ($d = 1, 2, \cdots, K$) is given by $\sigma = \mu \times \left(\frac{\sum_{p=1}^{K-1}\sum_{q=p+1}^{K}\text{dist}(c_p, c_q)}{K(K+1)/2}\right)$, μ a scaling factor. Differentiating the sum-of-squares error function with respect to weights and setting the derivative to zero, $(\phi^T\phi)W = \phi^T T$ is obtained. Weights are trained by solving this equation via singular value decomposition. ϕ is the matrix $[\phi_{id}]_{n\times(K+1)}$ with elements $\phi_{id} = \phi_d(\mathbf{x}_i)$, W is the matrix $[w_{dl}]_{(K+1)\times\mathcal{L}}$, and T is the matrix $[t_{il}]_{n\times\mathcal{L}}$ with elements $t_{il} = t_l^i$.

3.3.3 ML-BPNN

As with ML-RBF, ML-BPNN [153] is a three-layer neural network, having an input layer of identity neurons, and a hidden and output layer. Again, each neuron in the output layer corresponds to one appliance label. The layers are fully connected to one another, and both the hidden and output layer employ the $tanh()$ activation function. The network is trained (using the error back propagation algorithm) to minimize the global error $E = \sum_{i=1}^{n} E_i = \sum_{i=1}^{n} \frac{1}{|Y_i||\overline{Y}_i|}\sum_{(l,\lambda)\in Y_i\times\overline{Y}_i}\exp\left(-(y_i^l - y_i^\lambda)\right)$, where E_i is the error on the training instance $\mathbf{x}_i$, Y_i are the true labels of instance $\mathbf{x}_i$, $\overline{Y}_i$ is the complementary set of Y_i, $|\cdot|$ measures the cardinality of a set, $\exp\left(-(y_i^l - y_i^\lambda)\right)$ measures the difference between labels ($l \in Y_i$) belonging to $\mathbf{x}_i$ and labels ($\lambda \in \overline{Y}_i$) not belonging to $\mathbf{x}_i$. Minimizing the global error tends to assign higher values to labels that belong to an instance than those not belonging to it. A threshold function $t(x)$ is determined through training to finally predict labels for unknown instances, i.e. $Y = \{j | y_j > t(x), j \in \gamma\}$. $t(x)$ is modelled as a linear function $t(x) = w^T \cdot C(x) + b$. Given training data $(\mathbf{x}_i, Y_i)(1 \le i \le n)$, $C_i(x) = (c_1^i, c_2^i, \cdots, c_{\mathcal{L}}^i)$ is the predicted label set of instance i and the target

values of $t(x_i)$ are $t(\mathbf{x}_i) = \arg\min(|\{l|l \in Y_i, c_l^i \leq t\}| + |\{j|j \in \overline{Y}_i, c_j^i > t\}|)$. When the minimum is not unique and the optimal values are an interval, the middle value of the interval is chosen. Parameters (w, b) are learned by solving a matrix equation $\Phi \cdot \omega = t$ via linear least squares. Φ is a $n \times (\mathcal{L}+1)$ matrix and the ith row is $(c_1^i, c_2^i, \cdots, c_{\mathcal{L}}^i, 1)$, ω is a $(\mathcal{L}+1)$-dimensional vector (w, b), and t is $(t(x_1), t(x_2), , t(x_n))$ of n-dimensions.

3.3.4 ML-SVM

A SVM-based ranking approach is proposed by [47]. Given an instance x, suppose there exists k labels for x. A ranking function can be defined as $R_j(x) =< \omega_j, x > +b_j$ and j is in the label set if $R_j(x)$ is among the largest k elements of $(R_1(x), R_2(x), \cdots, R_{\mathcal{L}}(x))$. For a system that ranks the values of $< \omega_j, x > +b_j$, the decision boundary for x is defined by the hyperplanes $< \omega_j - \omega_\lambda, x > +b_j - b_\lambda = 0$, where j is in the label set and λ is not in the label set. The margin of (x, Y) is $\min_{j \in Y, \lambda \in \overline{Y}} \frac{<\omega_j - \omega_\lambda, x> + b_j - b_\lambda}{||\omega_j - \omega_\lambda||}$. Maximizing the margin on the whole training data set D and using the method of Lagrange multipliers, the optimization problem is transformed to $\min_{\omega_l, l=1,2,\cdots,\mathcal{L}} \sum_{j=1}^{\mathcal{L}} ||\omega_j||^2 + \varphi \sum_{i=1}^{n} \frac{1}{|Y_i||\overline{Y}_i|} \sum_{(j,\lambda) \in Y_i \times \overline{Y}_i} \varepsilon_{ij\lambda}$ and subject to $< \omega_j - \omega_\lambda, \mathbf{x}_i > +b_j - b_\lambda \geq 1 - \varepsilon_{ij\lambda}, (j, \lambda \in Y_i \times \overline{Y}_i)$ and $\varepsilon_{ij\lambda} \geq 0$. $\varepsilon_{ij\lambda}$ is a relaxing constraint term, φ is a trade-off parameter. When predicting labels for unknown instances, the method is the same as that used in BPNN-Based ML Classification.

3.4 Performance Metrics

The evaluation of multi-label classifiers requires different metrics than those used in single label classification. A unified categorization of the state-of-the-art multi-label classification criteria can be found in [155, 126]. Seven popular evaluation measures are provided in this report, including the Hamming loss, ranking loss, one error, coverage and average precision. Moreover, the widely used micro $F1$ and macro $F1$, which are based on label averaging methods, are also adopted [90, 108] for evaluation. Average precision, $F1_{micro}$ and $F1_{macro}$ are commonly used in NIALM studies, while the other metrics commonly appear in multi-label classification research. These performance metrics are a combination of metrics commonly used in NIALM and metrics

widely used in multi-label classification problems.

The Hamming Loss evaluates how many times an instance-label pair is misclassified, i.e., a label not belonging to the instance is predicted or a label belonging to the instance is not predicted. The smaller the value of hloss(h), the better the performance.

$$\text{hloss}(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\mathcal{L}}\left|h(\mathbf{x}_i)\Delta Y_i\right| \tag{3.3}$$

where Δ stands for the symmetric difference between two sets. $h(\cdot)$ is the multi-label classifier, and Y_i is the true label set for instance $\mathbf{x}_i$.

Ranking Loss measures the average fraction of label pairs that are reversely ordered for instance $\mathbf{x}_i$. The smaller the value of rloss(f), the better the performance.

$$\begin{aligned}\text{rloss}(f) \quad = \quad & \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i||\overline{Y}_i|}|\{(y_j,y_k)|f(\mathbf{x}_i,y_j)\leq f(\mathbf{x}_i,y_k),\\ & (y_j,y_k)\in Y_i\times\overline{Y}_i\}|\end{aligned} \tag{3.4}$$

where Y_i is the true label set of instance $\mathbf{x}_i$, $\overline{Y}_i$ is the complementary set of Y_i. y_j and y_k are the predicted labels.

One Error evaluates how many times the top-ranked label is not in the predicted label set of $\mathbf{x}_i$. The smaller the value of one_error(f), the better the performance.

$$\text{one\_error}(f) = \frac{1}{n}\sum_{i=1}^{n}\left[\left[\arg\max_{y\in\gamma} f(\mathbf{x}_i,y)\right]\notin Y_i\right] \tag{3.5}$$

where γ is the complete label set for the training dataset.

A list of class labels are sorted in descending order based on $f(\mathbf{x}_i,y)$. Coverage measures how far it is needed, to go down this label list so that all the true labels of $\mathbf{x}_i$ are covered. The smaller the value of coverage(f), the better the performance.

$$\text{coverage}(f) = \frac{1}{n}\sum_{i=1}^{n}\max_{y\in Y_i}\text{rank}_f(\mathbf{x}_i,y) - 1 \tag{3.6}$$

where $\text{rank}_f(\mathbf{x}_i,y)$ denotes the rank of each label $y\in\gamma$.

Average Precision calculates the average fraction of labels ranked above a particular label $y\in Y$ that are predicted. The larger the value of avgprec(f),

the better the performance.

$$\text{avgprec}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i|}\sum_{y\in Y_i} \frac{|\{y'|\text{rank}_f(\mathbf{x}_i, y') \leq \text{rank}_f(\mathbf{x}_i, y), y' \in Y_i\}|}{\text{rank}_f(\mathbf{x}_i, y)} \tag{3.7}$$

The $F1$ metric is the harmonic mean of precision and recall and it is widely adopted for single-label classification.

$$F1 = \frac{2 \times tp}{2 \times tp + fp + fn} \tag{3.8}$$

Where tp is the number of true positives, tn is the number of true negatives, fp is the number of false positives, and fn is the number of false negatives.

Macro $F1$ and micro $F1$ are multi-label classifier metrics derived by computing the $F1$ measure across the label set; either after summing true and false positives and false negatives across all labels, or by averaging the $F1$ measure for each label.

$$F1_{micro} = F1(\Sigma_{\lambda=1}^{M} tp_\lambda, \Sigma_{\lambda=1}^{M} fp_\lambda, \Sigma_{\lambda=1}^{M} fn_\lambda) \tag{3.9}$$

$$F1_{macro} = \frac{1}{M}\Sigma_{\lambda=1}^{M} F1(tp_\lambda, fp_\lambda, fn_\lambda) \tag{3.10}$$

where there are M labels in the dataset, tp_λ is the number of true positives, fp_λ is the number of false positives, and fn_λ is the number of false negatives of label λ after being evaluated by binary evaluation of $F1$.

3.5 Experimental Design

This section describes the datasets used in our experiments and the preprocessing methods employed. The multi-label classifiers were applied to four publicly available datasets, including the Reference Energy Disaggregation Dataset (REDD) [80], the Building-Level fUlly labeled Electricity Disaggregation dataset (BLUED) [4], the UMASS Smart* Home Data Set (Smart*) [7], and the Almanac of Minutely Power Dataset (AMPds) [98]. Each data set is treated as a time series (containing observations of the aggregate power in one house) in these experiments. The data set is first cleaned (missing values replaced), labeled, and downsampled. Features are then extracted from the time series using the method of delay embedding.

3.5.1 Filling Small Gaps

Missing values in each data set appear as small gaps in the time series, usually less than 10 seconds. We replace the missing gaps with the last-known value of the time series (i.e. a zero-order hold).

3.5.2 Labeling

The algorithm proposed in [43] is adapted to recognize events in the power series of individual appliances, by breaking the time series of sub-metered power into continuous small segments with the slope method and considering each segment as an event. Then the power of each segment is averaged, and labeled with the appropriate appliance. The process is composed of the following steps [43]:

1. Calculate the slopes of all data points in the power series. Since the time interval between data points is constant, the slope can be calculated as $\Delta P_n = P_{n+1} - P_n$.

2. From a starting point, joining the subsequent points with small slopes as a segment. If the slope value is beyond a threshold, or the difference of the steady points after the point and before the point is beyond a threshold, stop joining the point into the current segment. And the point is called as an interruption point. In order to prevent cutting off a segment due to occasional noisy points, an additional check is applied to the points behind the noisy data, namely, if the value of the steady points behind the noisy data is close to the value of steady points before the noisy data, then the noisy data is "ignored" and the following points will continue to be added to the segment.

3. Stating a new segment from the interruption point, add points into the segment with the method described in step 2.

4. Steps 2 and 3 iterate until all points have been added to segments. Thus the power series have been broken into continuous small segments.

5. Averaging the power of each segment.

6. Labeling for each segment.

The above labeling method is applied to the REDD, Smart* and AMPds dataset. The Blued dataset provides an event list which includes timestamps, events (every state transition of each appliance) and which phase it is on instead of power consumption data of individual appliance. Thus a different labeling method is proposed. The event list is matched against the aggregate main power. The difference between the average of the main power before an event and after the event is calculated (the average is taken over 60 data points). If there is an increase of main power, then it is a turn on event, otherwise it is a turn off event. For multi-state appliances, subsequent increases or decreases may be found in the main power when subsequent events happen for the same appliance. The event is considered an on event if there is a subsequent increase, or the last event is labeled off if there exist subsequent decreases for the same appliance. For all appliances, state only changes due to an event, and is otherwise constant.

3.5.3 Downsampling

The sampling rate of the data sets varies from 1 minute to 12 kHz. 12 kHz is a high sampling rate, considering that a 10-minute sampling frequency was still adequate for NIALM in [12] and [13]. Moreover, the size of the data set is too large to process. Downsampling is applied to decrease the sampling rate by keeping every jth sample starting with the first sample. For example, the raw data is down sampled to take every $100th$ data point starting with the first one in Smart*.

3.5.4 Delay Embedding

A NIALM dataset such as Smart* is an example of a time series, which is entirely distinct from the feature-space representation assumed in machine learning algorithms. It is thus necessary to extract features from this time series, so that a learning algorithm can be trained. One common method for doing so is to use delay embeddings (also known as *lagged* inputs). Embedding vectors are formed from prior observations from the time series concatenated together. According to Takens' embedding theorem [135], the underlying phase space of a physical system is equivalent to the space defined by the

embedding vectors. Given a single time series

$$R = \{R_N, R_{N-1}, \cdots, R_2, R_1\} \tag{3.11}$$

embedding vectors are given by

$$\Phi_n = \{R_{N-(m-1)\tau}, R_{N-(m-2)\tau}, \cdots, R_N\} \tag{3.12}$$

where R is the observable time measurement. Φ_n is the reconstructed vector and each of the vector is a set of features. The number of reconstructed vectors is $N-(m-1)\tau$. m is the embedding dimension, τ is generally referred to as the delay or lag, and allows us to select each sequential observation, every second observation, every third, etc. into the embedding vector. The parameters m and τ are critical to a proper embedding; too small of a dimensionality will leave some of the subspaces of the original state space unexpanded, while τ allows us to adjust the relative importance of temporal and spatial correlations in the delay vectors. There is no definitive means of determining appropriate values of m and τ, so the field of nonlinear time series analysis has instead developed heuristics. In this report, the time-delayed mutual information and false nearest neighbor heuristics are respectively adopted to find the best embedding delay time τ and embedding dimension m [76].

3.6 Experimental Results

Experiments with the multi-label classifiers in section III are implemented in Matlab and are carried out following a ten-fold cross validation design. Parameter exploration is conducted beforehand with five-fold cross-validation, and the best set of parameters is used for training in the final ten-fold design. Table 3.1 summarizes the final experimental results as means and standard deviations over the ten folds using the ML-KNN approach with the number of nearest neighbors set as 10. According to the metrics calculated in Table 3.1, it is obvious that ML-KNN achieves impressive scores over all the datasets. Both $F1_{micro}$ and $F1_{macro}$ are very high, indicating that ML-KNN disaggregates well for each monitored appliance.

Table 3.2 summarizes the experimental results using the ML-RBF approach over the ten folds. The fraction parameter β is set to be 0.1 and the scaling factor μ is 1. According to Table 3.2, ML-RBF performs well over

Table 3.1: Ten-fold cross validation results (mean±std) with ML-KNN over five datasets.

	House 1	House 3	Blued	Smart*	AMPds
Hamming Loss	0.02 (0.0008)	0.03 (0.002)	0.03 (6.98e-4)	0.15 (6.93e-4)	0.03 (2.25e-4)
Ranking Loss	0.01 (0.0006)	0.02 (0.001)	0.01 (6.33e-4)	0.08 (7.99e-4)	0.006 (1.36e-4)
One Error	0.05 (0.003)	0.05 (0.004)	0.01 (0.001)	1.69e-4 (1.5e-4)	0.002 (2.5e-4)
Coverage	0.12 (0.007)	0.37 (0.03)	4.13 (0.03)	9.09 (0.01)	2.92 (0.007)
Average Precision	0.97 (0.002)	0.97 (0.002)	0.98 (0.79e-4)	0.96 (4.64e-4)	0.99 (2.53e-4)
$F1_{micro}$	0.92 (0.003)	0.9 (0.004)	0.93 (0.001)	0.89 (5.67e-4)	0.94 (3.78e-4)
$F1_{macro}$	0.82 (0.016)	0.89 (0.006)	0.91 (0.004)	0.86 (0.001)	0.83 (0.01)

all the datasets in terms of Hamming loss, ranking loss, one error, average precision and $F1_{micro}$. The scores of $F1_{macro}$ are not that good over all the datasets except for Smart*, indicating some appliances can not be disaggregated well using ML-RBF.

Table 3.2: Ten-fold cross validation results (mean±std) with ML-RBF over five datesets.

	House 1	House 3	Blued	Smart*	AMPds
Hamming Loss	0.04 (0.001)	0.07 (0.002)	0.13 (0.002)	0.17 (0.001)	0.08 (7e-4)
Ranking Loss	0.03 (0.003)	0.08 (0.002)	0.09 (0.002)	0.1 (0.001)	0.03 (3.85e-4)
One Error	0.13 (0.004)	0.17 (0.006)	0.09 (0.004)	0.01 (0.001)	0.03 (0.001)
Coverage	0.26 (0.02)	0.6 (0.036)	5.58 (0.05)	9.26 (0.016)	2.62 (0.02)
Average Precision	0.92 (0.003)	0.89 (0.003)	0.87 (0.0026)	0.96 (5.54e-4)	0.95 (5.2e-4)
$F1_{micro}$	0.83 (0.005)	0.8 (0.004)	0.74 (0.003)	0.88 (0.001)	0.83 (0.001)
$F1_{macro}$	0.67 (0.01)	0.71 (0.009)	0.68 (0.006)	0.84 (0.002)	0.67 (0.007)

Table 3.3 summarizes the final experimental results using the ML-BPNN approach over the ten folds. The number of hidden neurons, the learning rate, and the training epochs are set to be 120, 0.05, and 50, respectively. According to Table 3.3, ML-BPNN performs poorly over all the datasets except for Smart*.

Table 3.3: Ten-fold cross validation results (mean±std) with ML-BPNN over five datesets.

	House 1	House 3	Blued	Smart*	AMPds
Hamming Loss	0.19 (0.052)	0.38 (0.016)	0.35 (0.04)	0.32 (0.003)	0.36 (0.044)
Ranking Loss	0.14 (0.057)	0.22 (0.026)	0.39 (0.051)	0.25 (0.031)	0.14 (0.063)
One Error	0.56 (0.245)	0.57 (0.046)	0.76 (0.069)	0.0003 (0.0004)	0.37 (0.335)
Coverage	0.95 (0.371)	1.12 (0.106)	9.79 (0.651)	11.31 (0.348)	7.78 (1.119)
Average Precision	0.66 (0.125)	0.63 (0.027)	0.47 (0.053)	0.9 (0.012)	0.78 (0.109)
$F1_{micro}$	0.4 (0.206)	0.48 (0.011)	0.51 (0.009)	0.81 (0.005)	0.67 (0.028)
$F1_{macro}$	0.15 (0.058)	0.28 (0.006)	0.45 (0.024)	0.77 (0.008)	0.42 (0.015)

Table 3.4 summarizes the experimental results using the ML-SVM approach over the ten folds. The radial basis function is used and the kernel is $\exp(-\Gamma \times |(x(i) - x(j)|^2))$. $x(i)$ and $x(j)$ are the input instances. Γ is set to be 1. According to Table 3.4, ML-SVM performs poorly over all the datasets except for Smart*.

From Tables 3.1, 3.2, 3.3 and 3.4, it can be concluded that ML-KNN and ML-RBF perform significantly better than ML-BPNN and ML-SVM with all the metrics over all the datasets, while ML-KNN also outperforms ML-RBF, and is generally effective on all datasets. We therefore compare the ML-KNN algorithm with existing results in the literature for each dataset. The comparison results on REDD are summarized in Table 3.5. ML-KNN is superior to the other approaches except for semi-Markov models and Dynamic Time Warping; note however, that results for the latter are superior to ML-KNN as measured by $F1_{macro}$, but inferior when measured by accuracy. This appears to be a statistical tie.

Table 3.4: Ten-fold cross validation results (mean±std) with ML-SVM over five datesets.

	House 1	House 3	Blued	Smart*	AMPds
Hamming Loss	0.23 (0.0007)	0.23 (0.002)	0.35 (0.002)	0.22 (0.02)	0.14 (0.001)
Ranking Loss	0.16 (0.001)	0.14 (0.002)	0.22 (0.004)	0.14 (0.015)	0.07 (0.004)
One Error	0.71 (0.006)	0.16 (0.005)	0.18 (0.005)	0.001 (0.004)	9.55e-5 (0.0001)
Coverage	1.08 (0.012)	2.01 (0.012)	1.79 (0.015)	9.69 (0.156)	7.01 (0.191)
Average Precision	0.56 (0.003)	0.86 (0.003)	0.85 (0.003)	0.94 (0.006)	0.92 (0.003)
$F1_{micro}$	0.49 (0.003)	0.62 (0.004)	0.45 (0.002)	0.85 (0.01)	0.78 (0.001)
$F1_{macro}$	0.16 (0.001)	0.23 (0.001)	0.17 (0.001)	0.74 (0.037)	0.26 (0.0001)

Table 3.5: Comparison results on REDD.

Method	CO[14]	FHMM[14]	FHMM1[23]	Watzzup[114]
Average accuracy	-	-	0.827	0.88
F_{macro}	0.31	0.31	0.7129	-
Method	AFAMAP[79]	SMF[79]	Bayesian[149]	Semi-MM[149]
Average accuracy	-	-	-	-
F_{macro}	0.52	0.22	0.83	0.9
Method	DTW[23]	ML-KNN		
Average accuracy	0.9124	0.97		
F_{macro}	0.8616	0.82		

The comparison results on Blued (phase B) are summarized in Table 3.6. ML-KNN outperforms all the other state-of-the-art algorithms in terms of ture positive rate (the most commonly used metric for this dataset).

Table 3.6: Comparison results on Blued (TPR denotes the true positive rate).

Method	GLR[5]	UAED[8]	Fast Shapelets[111]	ML-KNN
TPR	0.44	0.8896	0.8226	0.9343

The comparison results on AMPds are summarized in Table 3.7. ML-KNN is superior to the other algorithms in literature in terms of $F1_{macro}$.

Table 3.7: Comparison results on AMPds.

Method	CO[14]	FHMM[14]	ML-KNN
$F1_{macro}$	0.55	0.71	0.83

The comparison results on Smart* are summarized in Table 3.8. ML-KNN outperforms the other algorithms in terms of $F1_{macro}$.

Table 3.8: Comparison results on Smart*.

Method	CO[14]	FHMM[14]	ML-KNN
$F1_{macro}$	0.53	0.61	0.86

3.7 Summary

Fine-grained power monitoring technique is quite critical to help end users with the daily management of household appliances and achieve significant reduction of energy consumption. Since the NIALM, or known as energy disaggregation, was firstly proposed by Hart, increasing attentions have been attracted to this research area and a variety of NIALM techniques have been proposed in the past two decades. A detailed literature survey is given in this section to summarize the existing algorithms and results. However, most existing studies assume that simultaneous events do not occur and thus are only able to recognize a single appliance while simultaneous utilization of multiple appliances are quite common in reality. In view of this, this section explores the utilization of different multi-label classification algorithms in household appliance disaggregation and implements experiments using some public data sets. The experimental results demonstrate the effectiveness and applicability of the multi-label classification in household appliance disaggregation. The algorithms exploited in this section, such as the ML-KNN, ML-RBF, ML-SVM, and ML-BP outperform the methods from the references. In further, the ML-KNN achieves the best performance score compared with all the other algorithms.

Chapter 4

Semi-Supervised Learning for Non-Intrusive Load Monitoring

4.1 Overview

For real applications of NIALM, you might be able to collect few labeled data from households by persuading residents to take samples for each appliances during a short period of time, however, it seems impossible to sub-metering the households all the time because of privacy concern for being intrusive. Thus a few labeled data as well as a large number of unlabeled data (the house-level aggregate power data) could be gathered for NIALM. To solve NIALM in this case, a graph-based multi-label semi-supervised learning is proposed in this work. To the best of our knowledge, there is little research work in the existing literature using semi-supervised algorithms for NIALM. References [90] and [88] present our preliminary results of applying semi-supervised algorithms in NIALM. In contrast to supervised NIALM techniques which require sub-metering to gather the ground truth for training and unsupervised NIALM techniques which make no use of prior information, the semi-supervised learning method stands in the middle. Brief samples of individual appliances are taken during a temporary registration period. During the process, individual appliance is turned on for a short time period so its load signatures are captured. Thus a small number of labeled instances covering all the electric appliances inside the household are gathered. By incorporating massive unlabeled training data, this section disaggregates residential household aggregate power non-intrusively in a semi-supervised way. The same idea has been studied in other areas for binary labeled case [166, 161]. However, binary

labeling is a poor match to NIALM since multiple appliances can be operating simultaneously. Each appliance is regarded as a concept, thus multiple concepts exist for a single instance. The remainder of the section is organized as follows: Section II formulates NIALM in a semi-supervised multi-label way and introduces the proposed framework. Section III presents graph-based semi-supervised learning for multi-labels, including graph construction, graph Laplacian regularization, extension from transduction to induction, and manifold regularization. Section IV illustrates experimental design, including data preparation, experiment design and evaluation metrics. Section V gives experimental results and discussions, followed by conclusions in Section VI.

4.2 Problem Formulation

This subsection formulates the NIALM problem in the framework of semi-supervised learning, and introduces graph-based regularization methods for semi-supervised learning.

4.2.1 Load Disaggregation via Semi-Supervised Multi-Label Classifiers

Ground-truth information in NIALM is scarce, but a small amount can perhaps be collected for each house. A registration period, during which residents operate individual appliances for a brief time, was proposed in [55]. Based on the review in [36], consumers seem willing to put forward a small amount of effort if provided with a modest inducement (e.g. a discounted tarrif). Thus, NIALM data for a house potentially consists of a large volume of unlabeled data, augmented by a small number of labeled observations, leading to our proposed use of semi-supervised learning. To the best of our knowledge, there is little existing literature using semi-supervised algorithms for NIALM. References [90] and [88] present our preliminary results of applying semi-supervised algorithms in NIALM. Note also that our labeled examples include no appliance mixtures; in field use we do not expect the consumer to be willing to operate numerous appliance combinations during this registration period.

Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, \cdots, \mathcal{L}\}$ denote the feature space (aggregated power) and the label space (currently operating appliances) of an ML classifier,

respectively. In the framework of semi-supervised learning, the multi-label training dataset $\mathbb{D}$ is a combination of labeled and unlabeled data, namely, $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$, where $\mathbb{D}_l = \{(\mathbf{x}_i, \mathbf{y}_i) : \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \{-1, 1\}^{\mathcal{L}}, i = 1, 2, \cdots, l\}$ is the labeled set and $\mathbb{D}_u = \{\mathbf{x}_j \in \mathcal{X}, j = l+1, l+2, \cdots, l+u\}$ is the unlabeled set. l and u are the numbers of labeled and unlabeled samples respectively. $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^T$ denotes the ith aggregated power sample in a d-dimensional feature space $\mathcal{X}$. $\mathbf{y}_i = [y_{i1}, y_{i2}, \cdots, y_{i\mathcal{L}}]^T$ is the corresponding ith label vector. For labeled instances, $y_{ij} = 1$ indicates that the appliance $j \in \mathcal{Y}$ is active whereas $y_{ij} = -1$ denotes that the appliance j is inactive, while $y_{ij} = 0, \forall j \in \mathcal{Y}$ indicates that the ith instance $\mathbf{x}_i, i = l + 1, \cdots, l + u$ is unlabeled. The matrices of features and labels are denoted by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T$, where $N = l + u$.

The objective of this study is to learn a semi-supervised multi-label classifier $f : \mathcal{X} \to 2^{\mathcal{Y}}$ from the training dataset $\mathbb{D}$, so that future observations $\mathbb{D}_t = \{\mathbf{x}_i^t \in \mathcal{X} : i = 1, 2, \cdots, t\}$ can be correctly mapped to the appliances then active. We will accomplish this using low-rate NIALM data; observations from five public NIALM datasets will be down-sampled, yielding effective sampling rates between 6 and 60 seconds. This section furthermore explores classifying future instances either transductively or inductively (the latter allowing for generalization beyond the training set).

4.2.2 Framework of the Method

Graph-based learning is among the most active areas of Semi-Supervised Learning (SSL). We represent instances as nodes V of the graph and their relations as edges E connecting nodes. Graph-based methods involve two stages: first, a graph is constructed over the whole training data, including both the labeled set and the unlabeled set; then, labels for the unlabeled training instances are estimated through label propagation. Given a training dataset $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$, an adjacency graph $G = (V, E)$ is firstly constructed with nodes V and edges E. Usually the edge is weighted by a similarity measurement w_{ij} between node i and node j. The principal objective in this stage is to obtain a weight matrix $\mathbf{W}$ that faithfully represents the similarities between instances. This is normally assumed to mean that the labels are smooth with respect to the graph [151, 167]. In other words, given two instances with a close connection on the graph, their labels tend to be the same.

Mathematically, given labels of the labeled training set $\mathbb{D}_l$ and the similarity weight matrix $\mathbf{W}$ on $\mathbb{D}_l \cup \mathbb{D}_u$, the label set $\mathbf{F}_i = [F_{i1}, F_{i2}, \cdots, F_{i\mathcal{L}}]$ of an unlabeled instance $\mathbf{x}_i \in \mathcal{X}, i = l+1, l+2, \cdots, l+u$ is predicted by minimizing some loss function with respect to $\mathbf{W}$. This can be achieved through graph regularization methods, e.g. graph Laplacian regularization and manifold regularization [131, 167, 48, 162]. The graph Laplacian regularization learns a transductive classifier f, which can only predict labels for unlabeled training samples in $\mathbb{D}_u$. Methods have been proposed to extend such transductive methods to produce inductive solutions, so that the labels for out-of-sample data can be predicted [31, 37]. By contrast, the manifold regularization is intrinsically inductive. It learns a classifier f defined on the whole training set $\mathbb{D}$ over feature space $\mathcal{X}$ [19]. In the next section, we define novel extensions of these methods to the multi-label case, and ultimately apply them to NIALM.

It is noteworthy that [67], [157] also employ graph signal processing. Those works differ from ours in the following aspects: firstly [67] is a supervised learning algorithm, and [157] is an unsupervised one. Secondly, neither employs multi-label classification. Finally, both use a simpler variation regularization, instead of the more complex manifold regularization that was the best option in our work.

4.3 Graph-Based Semi-Supervised Learning for Multiple Labels

This subsection extends graph-based SSML classification algorithms to the multi-label case. Three graph regularization methods are studied: Gaussian Fields and Harmonic Functions (GFHF), Local and Global Consistency (LGC), and Manifold Regularization (MR). In subsection A, we first review the methods available for constructing the "graph" portion of this algorithm from a dataset with real-valued independent and categorical dependent variables. We discuss the design alternatives available for graph construction, and our rationale for the choices we have made. In subsection B, we then extend two algorithms (Gaussian Fields and Harmonic Functions, and Local and Global Consistency) that employ graph Laplacian regularization for label induction on unlabeled examples. These algorithms are exclusively transductive, so we also need to extend them to permit out-of-sample predictions. In subsection

C, we extend a third alternative (manifold regularization) to the multi-label case.

4.3.1 Graph Construction

Graph-based SSML learning begins with the construction of an adjacency graph, followed by determining the weight matrix $\mathbf{W}$. The idea of graph construction is as follows: Both labeled and unlabeled samples from a dataset are treated as nodes in a graph and pairwise edges are built to connect labeled nodes to unlabeled nodes. Edge weights represent the similarity between two samples. Then labels for unlabeled nodes can be predicted based on their connections to labeled nodes.

There are three types of commonly used graph, the fully connected graph, the k-Nearest Neighbor (kNN) graph, and ε-radius graph [131, 167]. For improved efficiency, better accuracy, and robustness to noise, the graph construction method in [144] is exploited here. Given the feature matrix $\mathbf{X}$, a full $N \times N$ distance matrix $\mathbf{U}$ is calculated with each element given by a Gaussian kernel

$$u_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right), \tag{4.1}$$

where σ is the bandwidth, calculated from each data set. A wide variety of kernel forms and distance measures may be substituted into eqn. (1); we empirically explored the use of Gaussian, linear and polynomial kernels, with the Euclidean and Cosine distances, before settling on the Gaussian kernel and Euclidean distance above.

At this point, we must also note that, since eqn. (1) requires the creation of an $N \times N$ distance matrix, the time complexity for graph construction is necessarily $O(N^2)$. This is a common limitation in graph-based learning, as well as in manifold- and kernel-based learning in general. In Section IV of this section we will discuss how we used undersampling to manage this time complexity; we later discuss how our results in Section V show that this strategy was successful.

Once $\mathbf{U}$ has been calculated, the derived matrix $\mathbf{H}$ with each element given by $h_{ij} = \sqrt{u_{ii} + u_{jj} - 2u_{ij}}$ is then used to determine $\mathbf{W}$ through graph sparsification and edge reweighting. There are two primary graph sparsification approaches: neighborhood search algorithms and matching algorithms (i.e., b-matching). The b-matching algorithm produces more bal-

anced graphs (i.e. each vertex in the graph has exactly the same number of edges) than neighborhood-based algorithms. This advantage plays a key role in conducting label propagation on unevenly and non-uniformly distributed samples [71, 144], and we therefore adopt it in this section. Mathematically, b-matching finds the binary matrix $\mathbf{B} = [B_{ij}]_{N\times N}$ by minimizing the following optimization problem

$$\begin{aligned} &\min_{\mathbf{B}\in\{0,1\}^{N\times N}} \sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}H_{ij}, \\ &s.t. \sum_{j=1}^{N} B_{ij} = b, B_{ii} = 0, B_{ij} = B_{ji}, \forall i, j = 1, \cdots, N. \end{aligned} \tag{4.2}$$

The above optimization problem can be solved using polynomial algorithms, which however have high computational complexity. As an alternative, loopy belief propagation is a much faster solution for the problem [69]. Belief Propagation (BP) is a classical dynamic programming approach. It computes conditional probabilities on the rest of the graph based on some given subset of the graph. The authors in [69] extended this method by setting up the b-matching optimization as BP on a loopy graphical model. The resulting algorithm circumvents the combinatorial message updates and thus is on average much faster than polynomial algorithms. With $\mathbf{B} = [B_{ij}]_{N\times N}$ computed, the graph edge between node i and node j can be removed if $B_{ij} = 0$.

Following this sparsification, the remaining edges must be reweighted. There are two popular reweighting methods, including the binary weighting and Gaussian kernel weighting. Using binary weighting, the weight matrix is obtained simply as $\mathbf{W} = \mathbf{B}$. Using Gaussian kernel weighting, the weight matrix $\mathbf{W}$ is calculated with each element given by

$$w_{ij} = B_{ij}u_{ij}. \tag{4.3}$$

Ultimately, the adjacency graph is represented via the sparse symmetric matrix $\mathbf{W}$. Then, the unnormalized graph Laplacian is

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{4.4}$$

where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} W_{ij}$. The normalized graph Laplacian is

$$\mathbf{\Delta} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}. \tag{4.5}$$

Both the unnormalized and normalized graph Laplacian matrices are commonly used in regularization for graph smoothness.

4.3.2 Graph Laplacian Regularization

We now extend label induction algorithms from the single-label to the multi-label case; generally, this means that the dependent variable becomes a categorical vector instead of a scalar. The work below parallels the development in [151] and [161], but is our own. Assuming $\mathbf{F}$ is the matrix of predicted labels for the whole training data $\mathbb{D}_l \cup \mathbb{D}_u$, the optimal labels $\mathbf{F}^*$ can usually be estimated through graph Laplacian regularization [144] by optimizing a certain objective function

$$\mathbf{F}^* = \arg\min_{\mathbf{F}\in\mathcal{F}} \Gamma(\mathbf{F}), \tag{4.6}$$

where $\mathcal{F}$ denotes the domain of $\mathbf{F}$. $\Gamma(\mathbf{F})$ is a cost function given by

$$\Gamma(\mathbf{F}) = \Gamma_f(\mathbf{F}) + \Gamma_s(\mathbf{F}), \tag{4.7}$$

where $\Gamma_f(\mathbf{F})$ is the empirical cost over labeled training set $\mathbb{D}$ and $\Gamma_s(\mathbf{F})$ ensures graph smoothness. The optimal solution $\mathbf{F}^*$ can be obtained using Gaussian Fields and Harmonic Functions (GFHF) [151] or Local and Global Consistency (LGC) [161], which have different forms for $\Gamma_f(\mathbf{F})$ and $\Gamma_s(\mathbf{F})$.

Gaussian Fields and Harmonic Functions

In the GFHF framework [151], the empirical cost and the regularization term in eqn. (4.7) for multiple labels are given by [151]

$$\Gamma_f(\mathbf{F}) = \mathrm{tr}\left((\mathbf{F}-\mathbf{Y})^T\mathbf{\Psi}(\mathbf{F}-\mathbf{Y})\right), \tag{4.8}$$

$$\Gamma_s(\mathbf{F}) = \mathrm{tr}\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right), \tag{4.9}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix, and $\mathbf{\Psi}$ is a diagonal matrix with each element given by

$$\Psi_{ii} = \begin{cases} 1 & \text{for} \quad i \le l, \\ 0 & \text{for} \quad l < i \le l+u. \end{cases} \tag{4.10}$$

The optimal solution $\mathbf{F}^*$ is obtained by differentiating $\Gamma(\mathbf{F})$ in eqn. (4.7) with respect to $\mathbf{F}$, [151]:

$$\left.\frac{\partial\Gamma(\mathbf{F})}{\partial\mathbf{F}}\right|_{\mathbf{F}=\mathbf{F}^*} = 2\mathbf{\Psi}(\mathbf{F}^*-\mathbf{Y}) + 2\mathbf{L}\mathbf{F}^* = 0, \tag{4.11}$$

which is rewritten as

$$(\mathbf{\Psi} + \mathbf{L})\mathbf{F}^* = \mathbf{\Psi}\mathbf{Y}. \tag{4.12}$$

Partitioning $\mathbf{L}$ and $\mathbf{F}^*$ into blocks corresponding to the labeled and unlabeled sets, we have $\mathbf{L} = \begin{bmatrix} \mathbf{L}_{ll} & \mathbf{L}_{lu} \\ \mathbf{L}_{ul} & \mathbf{L}_{uu} \end{bmatrix}$ and $\mathbf{F}^* = \begin{bmatrix} \mathbf{F}_l^* \\ \mathbf{F}_u^* \end{bmatrix}$. Since $\Psi_{ii} = 0, l < i \leq l+u$, there is

$$\mathbf{L}_{ul}\mathbf{F}_l^* + \mathbf{L}_{uu}\mathbf{F}_u^* = 0. \tag{4.13}$$

As a result, the optimal solution for multiple labels is given by [151]:

$$\begin{cases} \mathbf{F}_l^* = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_l]^T, \\ \mathbf{F}_u^* = -\mathbf{L}_{uu}^{-1}\mathbf{L}_{ul}\mathbf{F}_l^*. \end{cases} \tag{4.14}$$

Accordingly, the estimated label matrix $\hat{\mathbf{Y}}$ is obtained as: $\hat{y}_{ij} = 1$, if $F_{ij} \geq 0$; otherwise, $\hat{y}_{ij} = 0$, where F_{ij} is an element of $\mathbf{F}$.

Local and Global Consistency

In the LGC framework [161], the empirical cost and the regularization term in eqn. (4.7) for multiple labels are given by [144, 151]

$$\Gamma_f(\mathbf{F}) = \mu \operatorname{tr}\left((\mathbf{F} - \mathbf{Y})^T(\mathbf{F} - \mathbf{Y})\right), \tag{4.15}$$

$$\Gamma_s(\mathbf{F}) = \operatorname{tr}\left(\mathbf{F}^T\mathbf{\Delta}\mathbf{F}\right). \tag{4.16}$$

By differentiating $\Gamma(\mathbf{F})$ in eqn. (4.7) with respect to $\mathbf{F}$, i.e.,

$$\left.\frac{\partial\Gamma(\mathbf{F})}{\partial\mathbf{F}}\right|_{\mathbf{F}=\mathbf{F}^*} = 2\mu(\mathbf{F}^* - \mathbf{Y}) + 2\mathbf{\Delta}\mathbf{F}^* = 0, \tag{4.17}$$

the optimal $\mathbf{F}^*$ is eventually obtained as

$$\mathbf{F}^* = \left(\frac{\mathbf{\Delta}}{\mu} + \mathbf{I}\right)^{-1}\mathbf{Y}, \tag{4.18}$$

where $\mathbf{I}$ is a $N \times N$ identity matrix, and $\mu > 0$ is the regularization parameter [161]. Analogous to the GFHF method, the estimated label matrix $\hat{\mathbf{Y}}$ is obtained by comparing $\mathbf{F}^*$ with 0. However, unlike GFHF, LGC also relabels some labeled examples, under the assumption that some given labels might be wrong. In other words, LGC assumes that the true, correct mapping from observations to labels is smooth, and if some of the labels provided as ground truth violate graph smoothness, those are noise or errors. LGC will therefore correct those erroneous labels to ensure overall smoothness.

Extension from Transduction to Induction

GFHF and LGC are transductive learning algorithms, and thus can only predict labels for in-sample data. However, the application to NIALM requires an inductive step, so as to predict labels for future out-of-sample data. Studies in [37] and [31] present inductive solutions by simply minimizing a regularization criterion applied to each out-of-sample instance. Here, we extend the graph-based SSL methods to inductive learning by taking the labeled set and the unlabeled set as well as their estimated labels $\hat{\mathbf{Y}}$ to train a multi-label classifier, e.g., ML-KNN [154] (which was found to outperform a group of other ML classifiers on the NIALM problem in [86]).

Specifically, k nearest neighbors for $\mathbf{x}_i, i = 1, 2, \cdots, N$, are determined; let these neighbors be denoted as $\mathcal{N}(\mathbf{x}_i)$. A membership counting vector $\mathbf{C}$ for $\mathbf{x}_i$ is given by [154]:

$$\mathbf{C}_{(\mathbf{x}_i)}(m) = \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} y_{jm}, \tag{4.19}$$

where $m \in \mathcal{Y}$. Then given a test sample $\mathbf{x}$, its mth label is determined as

$$\begin{aligned} y_m &= \arg\max_{b \in \{-1,1\}} P\left(H_m^b | E_m^{\mathbf{C}_\mathbf{x}(l)}\right) \\ &= \arg\max_{b \in \{-1,1\}} \frac{P\left(H_m^b\right) P\left(E_m^{\mathbf{C}_\mathbf{x}(m)} | H_m^b\right)}{P\left(E_m^{\mathbf{C}_\mathbf{x}(m)}\right)} \\ &= \arg\max_{b \in \{-1,1\}} P\left(H_m^b\right) P\left(E_m^{\mathbf{C}_\mathbf{x}(m)} | H_m^b\right), \end{aligned} \tag{4.20}$$

where H_m^1 (H_m^{-1}) is the event that $\mathbf{x}$ has (no) mth label, and $E_m^j (j \in 0, 1, \cdots, k)$ denotes the event that there are j instances among its neighbors having the mth label. The prior probabilities $P(H_m^b)$ and the posterior probabilities $P\left(E_m^{\mathbf{C}_\mathbf{x}(m)} | H_m^b\right)$ are calculated from the whole training dataset. Eventually, the predicted label vector for the test sample $\mathbf{x}$ is $\mathbf{y} = [y_1, y_2, \cdots, y_\mathcal{L}]^T$ [154].

4.3.3 Manifold Regularization

In contrast to the GFHF and LGC regularizations, Manifold Regularization (MR) is intrinsically an inductive method that includes elements of spectral graph theory, manifold learning, and regularization in Reproducing Kernel Hilbert Space (RKHS). Again, this work parallels the developments

in [19, 162], but is our own. We extend MR for multiple labels, learning a mapping function $f : \mathcal{X} \rightarrow \mathbb{R}^{\mathcal{L}}$ in an RKHS K by solving the following optimization problem

$$\min_{f:\mathcal{X}\rightarrow\mathbb{R}^{\mathcal{L}}} \Gamma_f(\mathbf{F}) + \gamma_A \Gamma_k(\mathbf{F}) + \gamma_I \Gamma_s(\mathbf{F}), \tag{4.21}$$

where $\mathbf{F} = f(\mathbf{X})$. $\Gamma_f(\mathbf{F})$ is a loss function term with respect to $\mathbf{F}$ and $\mathbf{Y}$, $\Gamma_k(\mathbf{F})$ is a smoothness penalty term related with respect to the RKHS norm, and $\Gamma_s(\mathbf{F})$ is a smoothness penalty with respect to the graph Laplacian, while γ_A and γ_I are their respective regularizers; the values for γ_A and γ_I must be determined empirically. Depending on the choices of $\Gamma_f(\mathbf{F}, \mathbf{Y})$, γ_A, and γ_I, there are different algorithms to solve the above optimization problem [19]. In this section, the Laplacian Regularized Least Squares (LapRLS) is adopted. Accordingly, the three regularization terms in eqn. (4.21) are

$$\Gamma_f(\mathbf{F}) = \frac{1}{l} \operatorname{tr}\left((\mathbf{F}-\mathbf{Y})^T \mathbf{\Psi} (\mathbf{F}-\mathbf{Y})\right), \tag{4.22}$$

$$\Gamma_k(\mathbf{F}) = \operatorname{tr}\left(||\mathbf{F}||_K^2\right), \tag{4.23}$$

$$\Gamma_s(\mathbf{F}) = \frac{1}{N^2} \operatorname{tr}\left(\mathbf{F}^T \mathbf{L} \mathbf{F}\right), \tag{4.24}$$

where $||\mathbf{F}||_K$ is a norm of $\mathbf{F}$ in the chosen RKHS.

Analogous to the single label case in [19], the solution of the optimization problem in eqn. (4.21) is an expansion of kernel functions over both the labeled and unlabeled instances:

$$\mathbf{F}^* = \mathbf{K}\mathbf{\Lambda}, \tag{4.25}$$

where $\mathbf{\Lambda}$ is a $N \times \mathcal{L}$ coefficient matrix. $\mathbf{K}$ is a $N \times N$ matrix obtained from the kernel function $K(\cdot)$ with respect to all training samples $\mathbf{X}$. Then, $||\mathbf{F}||_K^2 = \mathbf{\Lambda}^T \mathbf{K} \mathbf{\Lambda}$. Taking $K(\cdot)$ as a Gaussian kernel function, the kernel matrix can be obtained as $\mathbf{K} = \mathbf{U}$ using eqn. (4.1). The coefficient matrix $\mathbf{\Lambda}^*$ is obtained though

$$\left.\frac{\partial\left(\Gamma_f(\mathbf{K}\mathbf{\Lambda}) + \gamma_A \Gamma_k(\mathbf{K}\mathbf{\Lambda}) + \gamma_I \Gamma_s(\mathbf{K}\mathbf{\Lambda})\right)}{\partial \mathbf{\Lambda}}\right|_{\mathbf{\Lambda}=\mathbf{\Lambda}^*} = 0. \tag{4.26}$$

Analogous to the result in [131], $\mathbf{\Lambda}^*$ for the case of multiple labels is

$$\mathbf{\Lambda}^* = \left(\mathbf{\Psi}\mathbf{K} + l\gamma_A \mathbf{I} + \frac{l\gamma_I}{N^2}\mathbf{L}\mathbf{K}\right)^{-1} \mathbf{Y}, \tag{4.27}$$

where $\mathbf{I}$ is a $N \times N$ identity matrix.

In the test phase, the output $\mathbf{F}^t$ for a given test sample $\mathbf{x}$ can be obtained as follows: First, a kernel vector $\mathbf{K}^t = [K_1^t, K_2^t, \cdots, K_N^t]$ is calculated with $K_i^t = K(\mathbf{x}, \mathbf{x}_i)$, where $\mathbf{x}_i$ is the ith sample from the training set $\mathbf{X}$. Second, based on eqn. (4.25), the output $\mathbf{F}^t$ for $\mathbf{x}$ is estimated as

$$\mathbf{F}^t = \mathbf{K}^t \mathbf{\Lambda}^*. \tag{4.28}$$

Eventually, the estimated label vector $\mathbf{Y}^t$ for the test sample $\mathbf{x}$ is obtained by comparing $\mathbf{F}^t$ with 0.

4.4 Experimental Methodology

This subsection describes our evaluation methodology, including data preparation, experimental design, and performance metrics, for the proposed NIALM methods.

4.4.1 Data Preparation

Multiple public power disagreggation datasets are used in this section to test our algorithms. The Reference Energy Disaggregation Dataset (REDD) [80] was collected from a number of real households in United States over various periods of a few months each. Specifically, one phase each from House 1 and House 3 (which are widely used in comparative studies) are used in our experiments. The Building-Level fUlly labeled Electricity Disaggregation dataset (BLUED) [4] was collected from a single US household over one week. The UMass Smart* Home Dataset (Smart*) [7] was collected from three households in Western Massachusetts. Specifically, the data from household A is used in our experiments. The Almanac of Minutely Power Dataset (AMPds) [98] was collected from a household in the great Vancouver area, Canada. To distinguish these datasets from each other in the experiments, they are denoted as House 1, House 3, BLUED, Smart*, and AMPds, respectively. For each dataset, we only use active (real) power in our experiments. Some appliances were excluded as they were always active or inactive. The actual numbers of appliances used from House1, House3, Blued, Smart*, and AMPds, are 9, 7, 25, 25, and 13, respectively. The list of appliances (or circuits) modeled for each dataset is presented in the tables of individual disaggregation performance in the next subsection.

These datasets are classic time series, and thus not formatted in the manner our algorithms (or most other learning algorithms) require. We will perform a delay embedding [135] to convert our datasets to collections of feature vectors (each being a sequence of lags relative to the current sample instant). Given a single time series $R = \{R_N, R_{N-1}, \cdots, R_2, R_1\}$, the delay vector for the i-th sample is $\mathbf{x}_i = [R_{i-(m-1)\tau}, R_{i-(m-2)\tau}, \cdots, R_i]$, where $i = N, N-1, \cdots, (m-1)\tau + 1$. τ and m are the embedding delay and dimension, respectively, which can be estimated using the mutual information and false nearest neighbors heuristics [86, 76]. Each of these feature vectors is equivalent to a reconstructed state vector for the household power signal at that time instant, which (assuming that household power consumption is a dynamical system) implies that appliance labels for the $(i+1)$-th sample can be predicted independently for each feature vector. Additional data preprocessing is required to impute missing values, associate labels with observations, and down sample the data. The detailed procedures of our data preprocessing are presented in [86].

The labeling of House 1, House 3, Smart*, and AMPds is performed using the slope algorithm [43], which breaks the time series of sub-metered power into continuous small segments. Then the power of each segment is averaged and labeled with the active appliances. The labeling of the BLUED dataset is different since an event list of state transitions of each appliance is available, making the labeling more straightforward. The slope algorithm in [43] is capable of recognizing significant power variances in the data as noisy points. Whenever such noisy points are detected, they will be filtered out from the dataset. This procedure can reduce the influence of power variances on disaggregation performance.

We down-sample the datasets by keeping every jth sample starting with the first sample. In this study, the first four datasets have been downsampled to take every 10th, 10th, 72000th, and 60th data point in House 1, House 3, Blued, and Smart*, respectively, and AMPds has not been downsampled. As a result, the sampling periods of House 1, House 3, Blued, and Smart* after downsampling are 30 s, 30 s, 6 s, and 60 s, respectively, while AMPds remains at one minute. These sampling periods are compatible with the capabilities of the smart metering infrastructures outlined in the Introduction. While different downsampling rates could be adopted, and may have some impact

on disaggregation performance, the research on low-rate NIALM does seem to adopt sampling periods not dissimilar to what we are using [67]. Our experiments thus assume the sampling rates above, and focus on exploring the ratio of labeled to unlabeled examples in the datasets, as this seems to be the more influential parameter. We select a portion of the training dataset to label, leaving the rest unlabeled; this "labeling rate" varies from 5% to 50%, in increments of five percentage points. Examples to be labeled are chosen randomly, with the restriction that the "on" and "off" states of every appliance must appear in the labeled set.

At this point, we must acknowledge that our experiments (and indeed, much of the experimental work reported in the NIALM literature) are not strictly comparable on a like-for-like basis. Early NIALM work was commonly based on unique datasets collected by a research group, and thus very difficult to replicate; this changed with the release of the REDD dataset and other public datasets [134], but problems of comparability still remain. The datasets are very large, and so most experimental studies must down-sample them, select appliance subsets to model, and determine how training and testing sets will be selected. For example, two of our major contrasts, [67] and [157], employ different appliance subsets and select different test sets even though both are evaluated on the "same" NIALM dataset.

4.4.2 Experimental Design

As discussed, we assume that NIALM data will consist of a small amount of labeled observations, and a large number of unlabeled ones. The relative sizes of the two groups are unknown, so our experiments are designed to explore the performance of our algorithms across a range of size ratios.

The down-sampling described above reduces the datasets to roughly 100,000 observations apiece (save AMPds, which contains roughly 525,000). While a times series of this size is not enormous by today's standards, our graph-based methods require quadratic time. Hence, the experiments we have outlined above would require months of compute time to complete for even the down-sampled data. Instead, we seek to leverage the smoothness property for the graphs we constructed in Section III. Specifically, if the graphs are smooth, then the observed feature vectors should actually map to a lower-dimensional set of latent variables that still predict class labels just as accurately as the

observed ones (note that the methods in this section do not *explicitly* define those latent variables, but they are implicit in the classifier models). Furthermore, it should still be possible to find those same latent variables with only a subset of the full dataset. This then implies that if we assemble a subset of the dataset by drawing without replacement, and then train our models on it, those models should perform roughly as well as if the full dataset was used. Alternatively, if we repeatedly assemble these subsets, randomly drawing a small fraction of the dataset each time, then the resulting models should perform very similarly (i.e. the variance in their performance should be low). This is one of the hypotheses we will examine in our experimental results.

As discussed earlier, we are already randomly drawing examples from the training set to be labeled in our experiments; combining this with the random draws of examples is trivial. Thus, for a given labeling rate, the experiment is replicated 10 times for each algorithm applied to each dataset; in each experiment, only a randomly-selected subset of examples are used. 90% of the drawn examples will form the training set, subdivided into the labeled and unlabeled groups. The remaining 10% forms the test data set. We choose the size of the sample to be 10% of the down-sampled training data for House 1, House 3, BLUED, and Smart*, respectively; for AMPds we draw only 2% of the data, leaving each of the datasets roughly equal in size. In total, there are 10368, 12960, 10080, 12960 and 10512 observations for each dataset, respectively. Modeling datasets of this size is easily accomplished on a personal computer. In our case, the experiments are programmed using MATLAB and conducted on a personal computer with an Intel Core i7-4770 CPU at 3.4GHz, 12 GB RAM, running Windows 10 Home Edition (64-bit), and MATLAB R2014a.

4.4.3 Performance Metrics

The performance of multi-label classification methods can be evaluated using a variety of metrics; state-of-the-art reviews are presented in [155, 126]. It would be fair to say that these are generally based on the usual single-label classification measures (e.g. accuracy, precision, the F-measure, etc.) NIALM algorithms can also be evaluated with a variety of metrics; these include both those same classification measures, and ones based on the total amount of energy correctly assigned to appliances (e.g. root-mean-square error, energy

error, etc.)

We select four popular ML classification metrics, including Hamming loss, average precision, $F1_{macro}$, and $F1_{micro}$, and one energy based metric, namely, the average normalized error. The Hamming loss is the fraction of examples misclassified:

$$\text{hloss}(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\mathcal{L}}\left|h(\mathbf{x}_i)\Delta\mathbf{y}_i\right|, \tag{4.29}$$

where Δ stands for the symmetric difference between two sets. $h(\cdot)$ is the multi-label classifier, and $\mathbf{y}_i$ is the true label set for instance $\mathbf{x}_i$.

Average precision calculates the average fraction of labels (for one instance) ranked above a given label in that instance that are correctly predicted.

$$\begin{aligned}\text{avgprec}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\mathbf{y_i}|}\sum_{y_{ij}\in\mathbf{y_i}} \\ \frac{|\{y_i'|\text{rank}_f(\mathbf{x}_i, y_i') \leq \text{rank}_f(\mathbf{x}_i, y_{ij}), y_i' \in \mathbf{y}_i\}|}{\text{rank}_f(\mathbf{x}_i, y_{ij})},\end{aligned} \tag{4.30}$$

where y_i' is the chosen particular label, y_{ij} is the jth label of instance i. $F1_{macro}$ and $F1_{micro}$ are both based on the well-known F1 measure for single-label classifiers [90, 108]:

$$F1 = \frac{2\times tp}{2\times tp + fp + fn}, \tag{4.31}$$

where tp, fp, and fn are the number of true positives, false positives, and false negatives, respectively. For the multi-label case, the F1 measure must be averaged across labels; the two metrics we use differ in how that averaging is done.

$$F1_{micro} = F1\left(\sum_{\lambda=1}^{\mathcal{L}} tp_\lambda, \sum_{\lambda=1}^{\mathcal{L}} fp_\lambda, \sum_{\lambda=1}^{\mathcal{L}} fn_\lambda\right), \tag{4.32}$$

$$F1_{macro} = \frac{1}{\mathcal{L}}\sum_{\lambda=1}^{\mathcal{L}} F1(tp_\lambda, fp_\lambda, fn_\lambda), \tag{4.33}$$

where tp_λ, fp_λ and fn_λ are the numbers of true positives, false positives and false negatives of label $\lambda \in \mathcal{Y}$ evaluated by F1.

The average normalized error metric [67] is adopted to evaluate the energy disaggregation accuracy. The ANE metric is given by

$$ANE = \frac{\left|\sum_{i=1}^{n} P_i - \sum_{i=1}^{n}\hat{P}_i\right|}{\sum_{i=1}^{n} P_i}, \tag{4.34}$$

where P_i and $\hat{P}_{ij}$ represent the actual and estimated total power consumptions of all known appliances in $\mathcal{Y}$ at sample i, respectively. In this section, $\hat{P}_{ij}$ is a combination of average active/inactive power consumptions of individual appliances in $\mathcal{Y}$ based on their on/off states.

4.5 Experimental Results and Discussions

Our experimental results are now presented for each dataset. Our results are primarily presented in the fifteen plots; the tables below summarize those results.

4.5.1 Case I: REDD House 1

Table 4.1 presents each performance metric for the four methods applied to the REDD House 1 data. This table provides the overall mean and standard deviation of our experiments across all labeling rates; we present the mean of means of each performance metric across the ten replicates of each labeling rate, and the mean of the standard deviations of those same experiments. The mean of the average precision, Micro F1, and Macro F1 measures for each labeling rate are plotted in Figs. 4.1, 4.2 , and 4.3; standard deviations for each labeling rate are presented as error bars. The best performance for each metric is highlighted in bold. The SSML techniques all outperform ML-KNN on all metrics, with ML-MR the overall best. Two of the SSML techniques, namely, ML-GFHF and ML-LGC, outperform ML-KNN slightly, and ML-MR achieves the best performance. In contrast, the current best experimental results on the REDD dataset are an average accuracy of 0.9124 and $F1_{macro}$ of 0.8616 in [23], and an $F1_{macro}$ of 0.9 in [149]. ML-MR acheives similar results with a labeling rate of 10%.

Table 4.2 presents the average F1-measure of each individual appliance in the REDD House 1 data. The results also demonstrate the superior performance of ML-MR over the other methods. We can compare our method to [67] and [157], the other current graph signal processing approaches to NIALM, on this dataset only, and then only for the appliances in common with each of those papers. These are the Refrigerator, Microwave, Kitchen Outlet and Washer-Dryer circuits for [67]; the Macro-F1 measure for that paper is 0.72, while our ML-MR algorithm achieves 0.93 on those appliances. For [157], the

Table 4.1: The mean value and standard deviation (in bracket) of each performance metric for each method for the REDD House 1 data.

Performance Metrics	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Hamming loss	0.0705 (0.0182)	0.0694 (0.0165)	0.0706 (0.0193)	**0.0491** (0.0116)
Average precision	0.9075 (0.0204)	0.9090 (0.0189)	0.9095 (0.0203)	**0.9184** (0.0169)
Micro F1	0.8457 (0.0400)	0.8512 (0.0335)	0.8509 (0.0360)	**0.8955** (0.0240)
Macro F1	0.8443 (0.0637)	0.8483 (0.0506)	0.8464 (0.0522)	**0.9171** (0.0241)
ANE	0.0671 (0.0189)	0.0397 (0.0183)	0.0313 (0.0169)	**0.0096** (0.0052)

Table 4.2: The average F1-measure of each individual appliance in the REDD House 1 data.

Appliance	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
3 Oven	0.8624	0.8290	0.8168	0.9459
5 Refrigerator	0.8803	0.8899	0.8938	0.9170
9 Lighting	0.8663	0.8692	0.8666	0.8887
11 Microwave	0.7672	0.8225	0.8315	0.9269
12 Bathroom GFI	0.8891	0.8947	0.8957	0.9450
13 Electric heat	0.7760	0.7778	0.7752	0.9321
15 Kitchen outlets	0.8476	0.8548	0.8446	0.9377
18 Lighting	0.7621	0.7684	0.7735	0.8068
20 Washer dryer	0.9476	0.9281	0.9195	0.9539

Refrigerator, Microwave, Kitchen Outlet, Washer-Dryer, Oven, Lighting (unknown which one), and Bathroom GFI circuits are common with our work; the Macro-F1 measure for that paper is 0.68, while our ML-MR algorithm achieves 0.86 on those appliances (choosing Lighting-18 instead of Lighting-9, as the former was the weaker result in our experiments).

4.5.2 Case II: REDD House 3

Table 4.3 again summarizes the detailed results from Figure 1 on REDD House 3. Figs. 4.4, 4.5 , and 4.6, present the precision, Micro F1, and Macro F1 under different labeling rates for the four classification algorithms. Again,

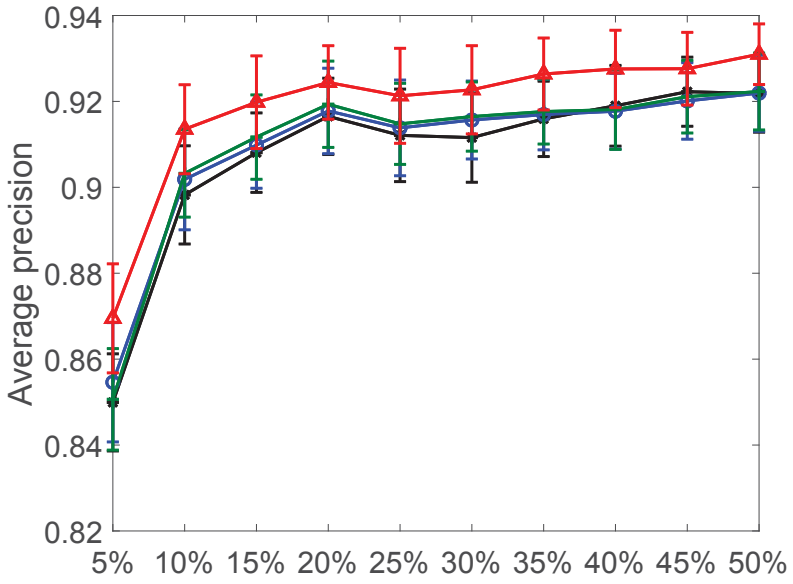

Figure 4.1: Classification precision v.s. labeling rates for classification algorithms applied to REDD House 1. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

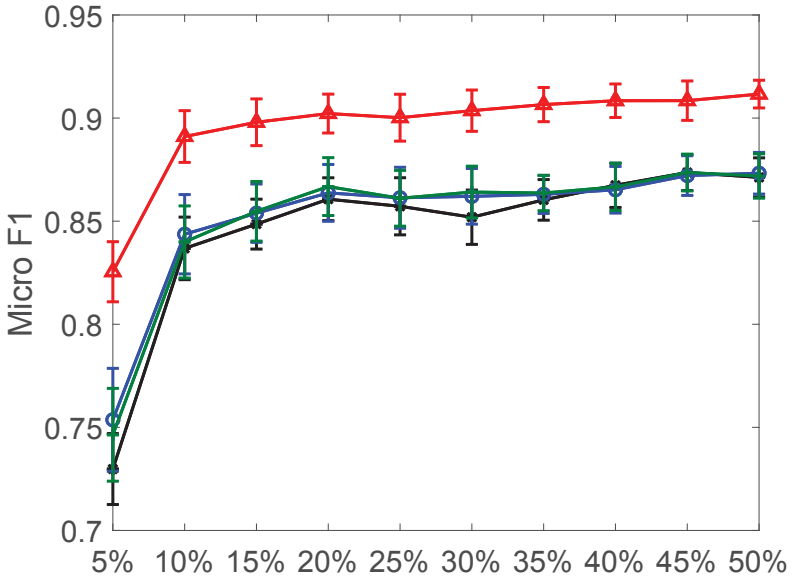

Figure 4.2: Micro F1 v.s. labeling rates for classification algorithms applied to REDD House 1. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

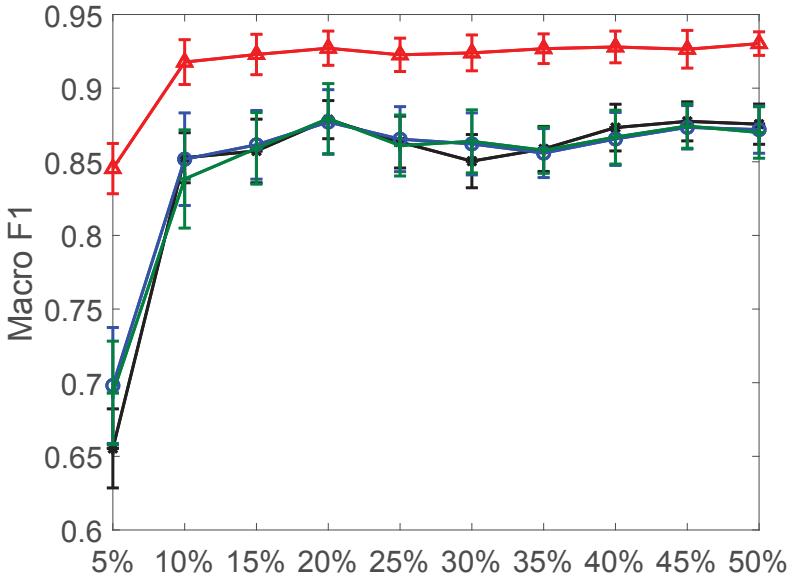

Figure 4.3: Macro F1 v.s. labeling rates for classification algorithms applied to REDD House 1. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

the three SSML techniques outperform ML-KNN on all metrics, with ML-MR again the overall best. In the literature, the best experimental results on this dataset reported as $F1_{macro}$ of 0.492 and $F1_{micro}$ of 0.959 , were presented in [134]. ML-MR achieves a better $F1_{macro}$ even when the labeling rate is just 5%. However, our best result with $F1_{micro}$ is 0.88. Table 4.4 presents the average F1-measure of each individual appliance in the REDD House 3 data. The results also demonstrate the superior performance of ML-MR algorithm over the other methods.

4.5.3 Case III: BLUED

Table 4.5 again summarizes the detailed results from Figure 1 on BLUED. Figs. 4.7, 4.8 , and 4.9, present the precision, Micro F1, and Macro F1 under different labeling rates for the four classification algorithms. It is obvious that the three SSML methods outperform the ML-KNN method. ML-MR performs much better than the other algorithms, especially under lower labeling rates. The best experimental result in the literature is a true-positive rate of 0.8896, presented in [8]. ML-MR achieves a similar performance when the labeling

Table 4.3: The mean value and standard deviation (in bracket) of each performance metric for each method for the REDD House 3 data.

Performance Metrics	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Hamming loss	0.1216 (0.0127)	0.1173 (0.0094)	0.1188 (0.0121)	**0.1099** (0.0115)
Average precision	0.8835 (0.0122)	0.8866 (0.0105)	0.8855 (0.0131)	**0.8922** (0.0096)
Micro F1	0.8509 (0.0160)	0.8560 (0.0124)	0.8552 (0.0146)	**0.8647** (0.0149)
Macro F1	0.8600 (0.0241)	0.8667 (0.0151)	0.8636 (0.0169)	**0.8775** (0.0127)
ANE	0.0356 (0.0127)	0.0315 (0.0144)	**0.0226** (0.0116)	0.0272 (0.0085)

Table 4.4: The average F1-measure of each individual appliance in the REDD House 3 data.

Appliance	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
3 outlet	0.8078	0.8225	0.8236	0.8408
7 Fridge	0.7874	0.8058	0.8050	0.8185
9 Dishwasher	0.8844	0.8985	0.8776	0.9128
11 Lighting	0.8809	0.8811	0.8798	0.8868
13 Washer dryer	0.9744	0.9680	0.9654	0.9803
17 Lighting	0.8691	0.8725	0.8728	0.8788
19 Lighting	0.8163	0.8184	0.8211	0.8248

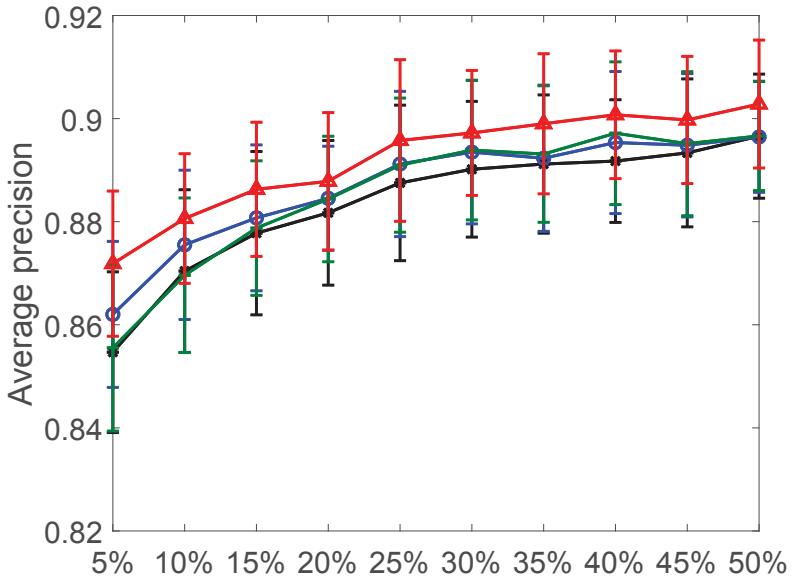

Figure 4.4: Classification precision v.s. labeling rates for classification algorithms applied to REDD House 3. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

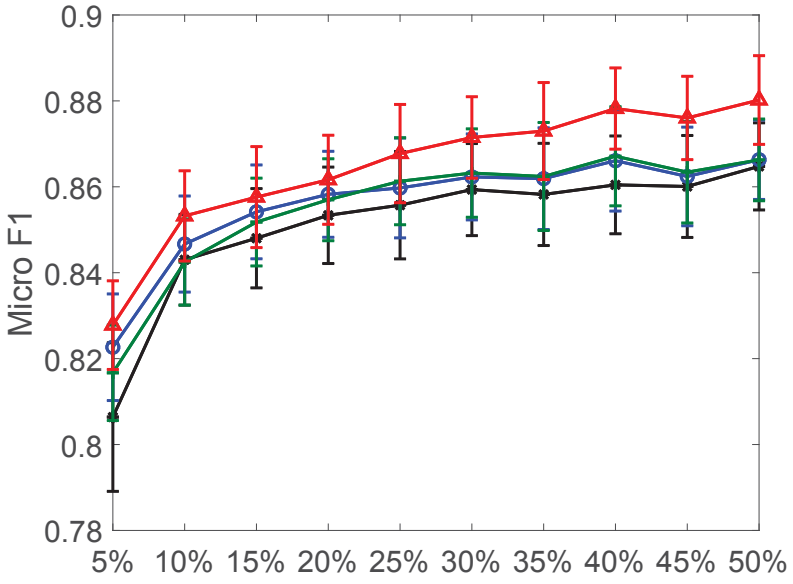

Figure 4.5: Micro F1 v.s. labeling rates for classification algorithms applied to REDD House 3. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

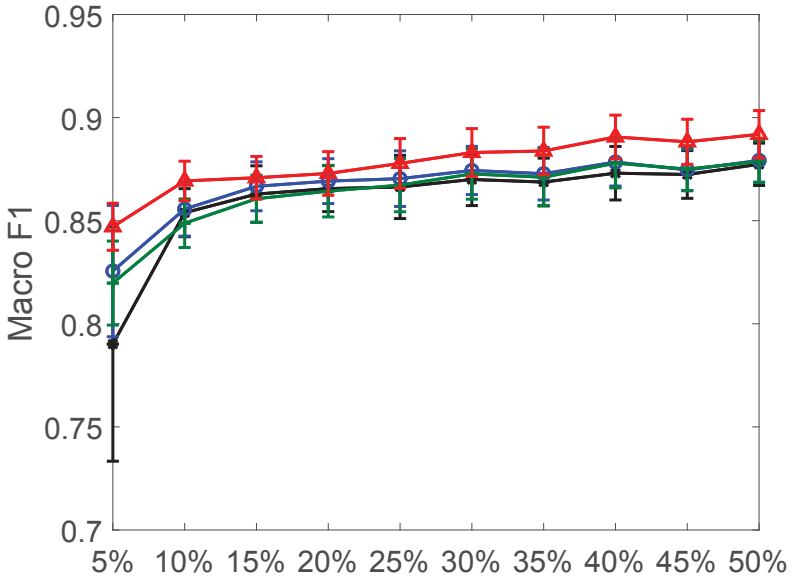

Figure 4.6: Macro F1 v.s. labeling rates for classification algorithms applied to REDD House 3. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

rate is approximately 25%. Table 4.6 presents the average F1-measure of each individual appliance in the BLUED data.

4.5.4 Case IV: Smart*

Table 4.7 again summarizes the detailed results from Figure 1 on Smart*. Figs. 4.10, 4.11 , and 4.12, present the precision, Micro F1, and Macro F1 under different labeling rates for the four classification algorithms. It can be seen that ML-GFHF and ML-LGC show better performances on the average precision and Micro F1 compared to the ML-KNN method, but not on the Macro F1. ML-MR achieves outstanding performance over all the other three methods, especially, on Micro F1 and Macro F1. In the literature, the best experimental result was $F1_{macro} = 0.61$, presented in [14]. ML-MR can achieve the same result with a 5% labeling rate. Table 4.8 presents the average F1-measure of each individual circuit in the Smart* data.

Table 4.5: The mean value and standard deviation (in bracket) of each performance metric for each method for the BLUED data.

Performance Metrics	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Hamming loss	0.1279 (0.0196)	0.1187 (0.0114)	0.1175 (0.0113)	**0.1068** (0.0099)
Average precision	0.8244 (0.0375)	0.8404 (0.0219)	0.8444 (0.0200)	**0.8520** (0.0150)
Micro F1	0.7028 (0.0535)	0.7312 (0.0278)	0.7370 (0.0232)	**0.7659** (0.0219)
Macro F1	0.5996 (0.1304)	0.6307 (0.0807)	0.6531 (0.0570)	**0.7363** (0.0328)
ANE	0.0566 (0.0214)	0.0403 (0.0164)	0.0319 (0.0157)	**0.0179** (0.0123)

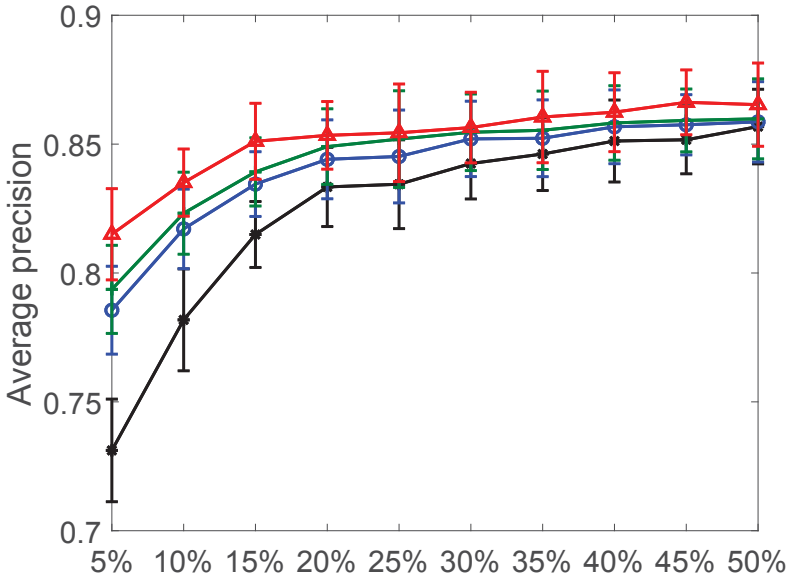

Figure 4.7: Classification precision v.s. labeling rates for classification algorithms applied to Blued. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

Table 4.6: The average F1-measure of each individual appliance in the BLUED data.

Appliance	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
1 Desktop lamp	0.6517	0.7019	0.7101	0.7335
2 Tall desk lamp	0.6517	0.7018	0.7102	0.7335
3 Garage door	0.3589	0.3279	0.4430	0.7827
4 Washing machine	0.7968	0.8151	0.8174	0.8371
10 A/V living room	0.6794	0.6975	0.7168	0.7470
12 Computer A	0.8700	0.8709	0.8695	0.8816
13 Laptop B	0.7126	0.7632	0.7611	0.7776
16 DVR, A/V basement	0.3627	0.4222	0.4466	0.5635
20 LCD Monitor A	0.7120	0.7633	0.7613	0.7776
21 TV Basement	0.4727	0.5054	0.5784	0.7973
23 Printer	0.6677	0.6841	0.6824	0.8594
25 Iron	0.6957	0.8318	0.9354	0.8695
26 Living room socket	0.5106	0.5582	0.5873	0.6799
28 Monitor B	0.5875	0.6208	0.6435	0.7087
31 Office lights	0.6125	0.7069	0.7353	0.7814
32 Closet lights	0.4579	0.5492	0.5617	0.5752
33 Upstairs hallway light	0.8363	0.8436	0.8399	0.8442
34 Hallways stairs lights	0.5082	0.6651	0.6707	0.6547
35 kitchen hallway light	0.6475	0.6791	0.6998	0.7194
36 kitchen overhead light	0.4710	0.5737	0.5884	0.6372
38 Dining room light	0.7421	0.7860	0.7849	0.8072
40 Basement light	0.7908	0.8158	0.8197	0.8447
41 Microwave	0.4629	0.4464	0.4919	0.7378
42 Air conditioner	0.5000	0.2469	0.2712	0.7076
43 Dryer	0.2307	0.1914	0.2016	0.3498

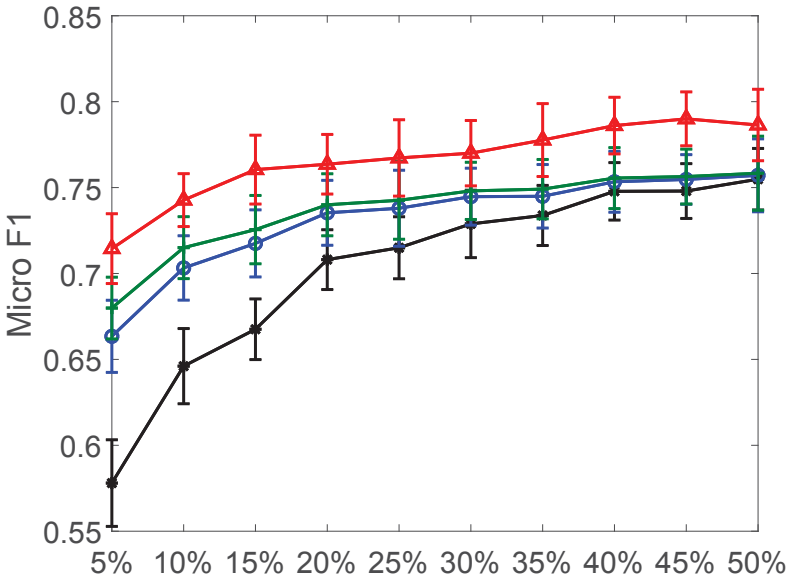

Figure 4.8: Micro F1 v.s. labeling rates for classification algorithms applied to Blued. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

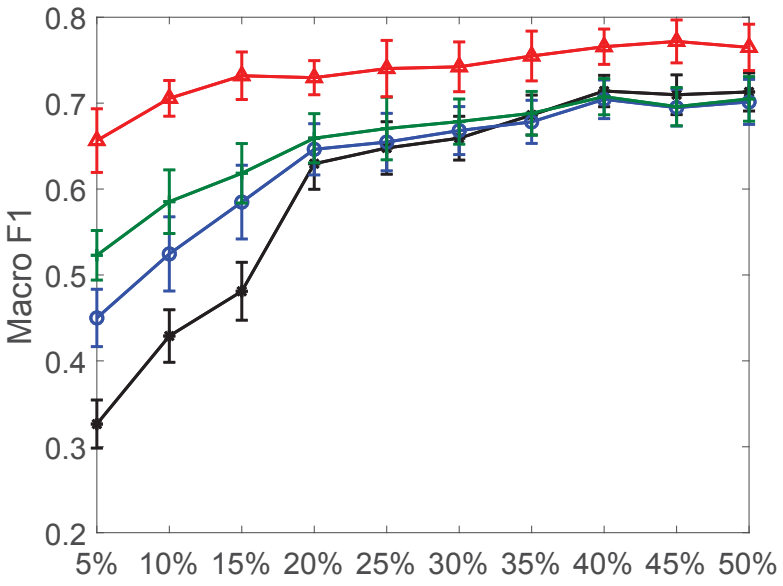

Figure 4.9: Macro F1 v.s. labeling rates for classification algorithms applied to Blued. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

Table 4.7: The mean value and standard deviation (in bracket) of each performance metric for each method for the Smart* data.

Performance Metrics	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Hamming loss	0.1248 (0.0066)	0.1217 (0.0036)	0.1203 (0.0032)	**0.0970** (0.0072)
Average precision	0.9300 (0.0065)	0.9333 (0.0052)	0.9350 (0.0049)	**0.9414** (0.0050)
Micro F1	0.8412 (0.0084)	0.8438 (0.0043)	0.8454 (0.0041)	**0.8762** (0.0091)
Macro F1	0.5028 (0.0424)	0.4860 (0.0335)	0.5173 (0.0112)	**0.7569** (0.0433)
ANE	0.1288 (0.0278)	0.1190 (0.0273)	0.1021 (0.0268)	**0.0370** (0.0162)

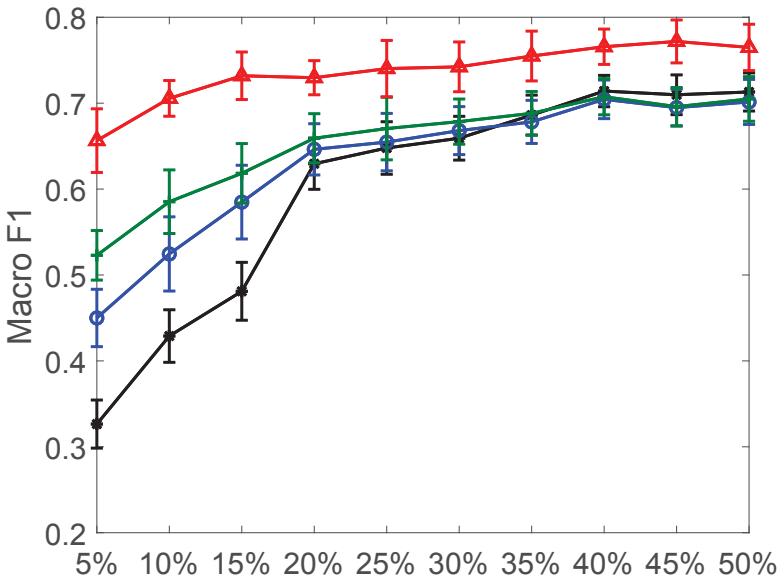

Figure 4.10: Classification precision v.s. labeling rates for classification algorithms applied to Smart*. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

Table 4.8: The average F1-measure of each individual circuit in the Smart* data.

Circuit	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Circuit 2	0.4447	0.3080	0.3247	0.6395
Circuit 3	0.0035	0.0071	0.0327	0.3751
Circuit 4	0.7086	0.7313	0.7291	0.7617
Circuit 5	0.4626	0.2113	0.2602	0.8608
Circuit 6	0.1383	0.1764	0.1838	0.8182
Circuit 7	0.1734	0.1512	0.2357	0.5784
Circuit 8	0.3614	0.2979	0.3794	0.7429
Circuit 9	0.6013	0.6298	0.6427	0.6793
Circuit 10	0.1755	0.1409	0.1877	0.4043
Circuit 11	0.0210	0.0211	0.1071	0.5488
Circuit 12	0.9862	0.9862	0.9867	0.9900
Circuit 13	0.9229	0.9220	0.9198	0.9289
Circuit 14	0.1213	0.0913	0.1376	0.5751
Circuit 15	0.8820	0.8823	0.8780	0.8808
Circuit 16	0.8103	0.8085	0.8129	0.8353
Circuit 17	0.0469	0.0512	0.1157	0.7456
Circuit 18	0.4917	0.4950	0.5562	0.6811
Circuit 19	0.8546	0.8640	0.8669	0.9067
Circuit 20	0.8431	0.8411	0.8380	0.8516
Circuit 21	0.9853	0.9858	0.9859	0.9939
Circuit 22	0.9794	0.9794	0.9796	0.9849
Circuit 23	0.5737	0.5880	0.6249	0.6970
Circuit 24	0.0892	0.0773	0.1545	0.8773
Circuit 25	0.8795	0.8744	0.8725	0.8801
Circuit 26	0.0142	0.0291	0.1201	0.6859

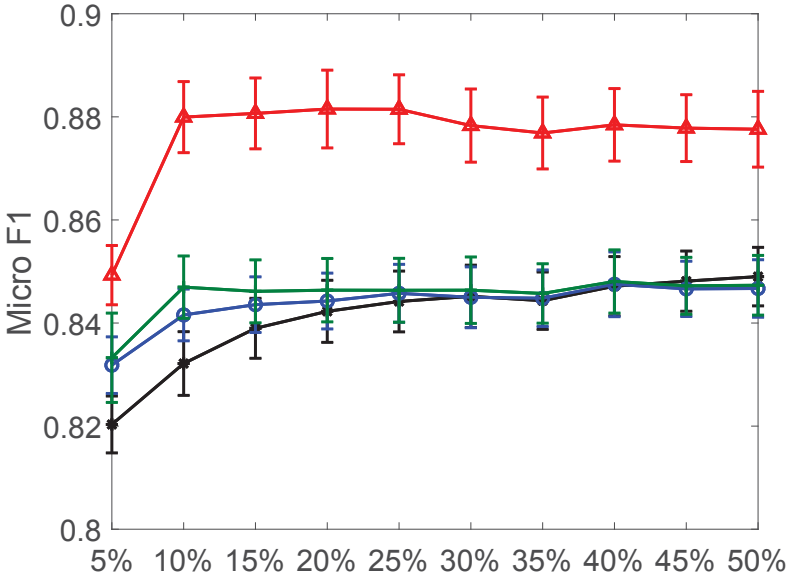

Figure 4.11: Micro F1 v.s. labeling rates for classification algorithms applied to Smart*. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

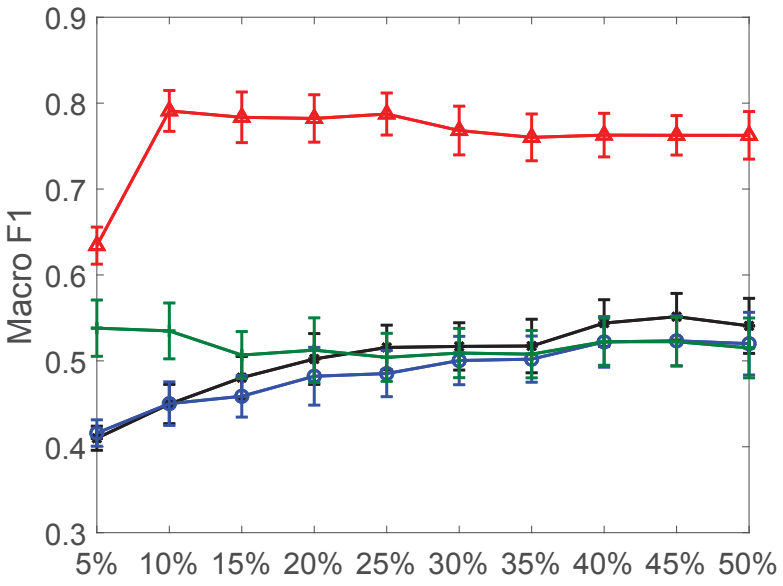

Figure 4.12: Macro F1 v.s. labeling rates for classification algorithms applied to Smart*. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

Table 4.9: The mean value and standard deviation (in bracket) of each performance metric for each method for the AMPds data.

Performance Metrics	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Hamming loss	0.1576 (0.0036)	0.1537 (0.0033)	0.1511 (0.0028)	**0.1162** (0.0107)
Average precision	0.8753 (0.0067)	0.8817 (0.0049)	0.8841 (0.0047)	**0.9000** (0.0075)
Micro F1	0.7050 (0.0070)	0.7170 (0.0106)	0.7269 (0.0048)	**0.8017** (0.0239)
Macro F1	0.2684 (0.0221)	0.2886 (0.0372)	0.3339 (0.0170)	**0.6978** (0.0742)
ANE	0.3476 (0.0271)	0.3167 (0.0185)	0.2808 (0.0199)	**0.1351** (0.0137)

4.5.5 Case V: AMPds

Table 4.9 again summarizes the detailed results from Figure 1 on Smart*. Figs. 4.13, 4.14 , and 4.15, present the precision, Micro F1, and Macro F1 under different labeling rates for the four classification algorithms. It is obvious that the three SSML methods outperform ML-KNN method, with ML-MR again the best. In the literature, the best experimental result was $F1_{macro} = 0.71$, presented in [14]. ML-MR will match that performance with a labeling rate between 10 and 15%. Table 4.10 presents the average F1-measure of each individual appliance in the AMPds data.

When we now consider the totality of the results in Figure 1, we see that the reported error bars all correspond to small standard deviations. This matches our predicted outcome from sub-sampling the low-rate NIALM data. Thus, sub-sampling even low-rate NIALM data seems to lead to highly effective models when using our proposed algorithms, and ML-MR in particular.

We have also tested the actual running times of our algorithms. Table 4.11 presents the mean and standard deviations of the running time for each method. It is obvious that ML-KNN is much faster than the three SSML algorithms. This is reasonable since ML-KNN is a supervised algorithm and thus only labeled instances are used, whereas the SSML algorithms use both labeled and unlabeled instances. Depending on the labeling rate, the size of data exploited by ML-KNN is 5%-50% of that exploited by SSML algorithms.

In summary, our general conclusion is that semi-supervised multi-label

Table 4.10: The average F1-measure of each individual circuit in the AMPds data.

Appliance	Semi-supervised learning			
	ML-KNN	ML-GFHF	ML-LGC	ML-MR
B2E Master/south Br	0.1428	0.1527	0.2504	0.4959
BME Basement	0.0287	0.0592	0.1366	0.4556
CDE Clothes dryer	0.3388	0.2830	0.3404	0.5464
CWE Clothes washer	0.0017	0.0016	0.0357	0.7107
DNE Dining room	0	0.0009	0.0332	0.7003
DWE Dishwasher	0.0052	0.0027	0.0210	0.6447
EBE Electronics	0.7390	0.7507	0.7598	0.7711
FGE Kitchen fridge	0.0732	0.2278	0.2874	0.5137
GRE Garage	0.0004	0.0024	0.0679	0.8611
HPE heat pump	0.9634	0.9634	0.9632	0.9710
HTE Instant hot water	0.1590	0.3067	0.4124	0.6006
OFE home office	0.9570	0.9571	0.9579	0.9742
WOE Wall oven	0.0800	0.0443	0.0750	0.8264

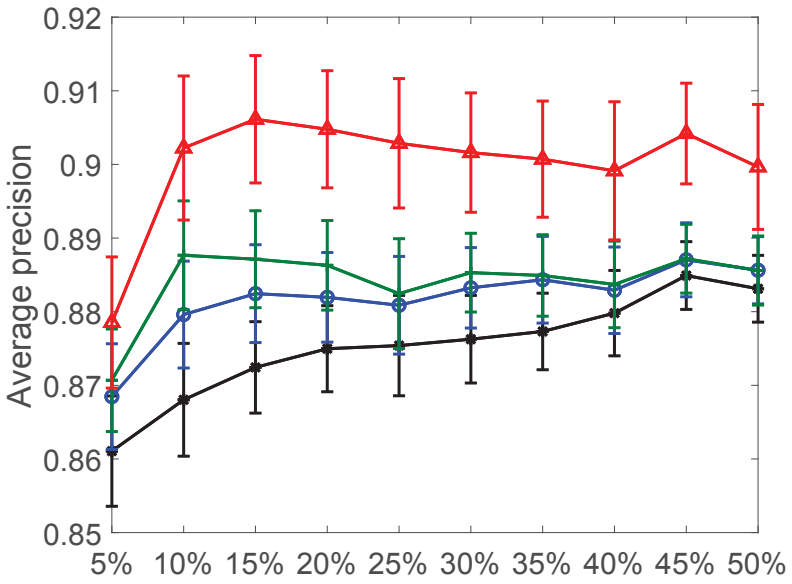

Figure 4.13: Classification precision v.s. labeling rates for classification algorithms applied to AMPds. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

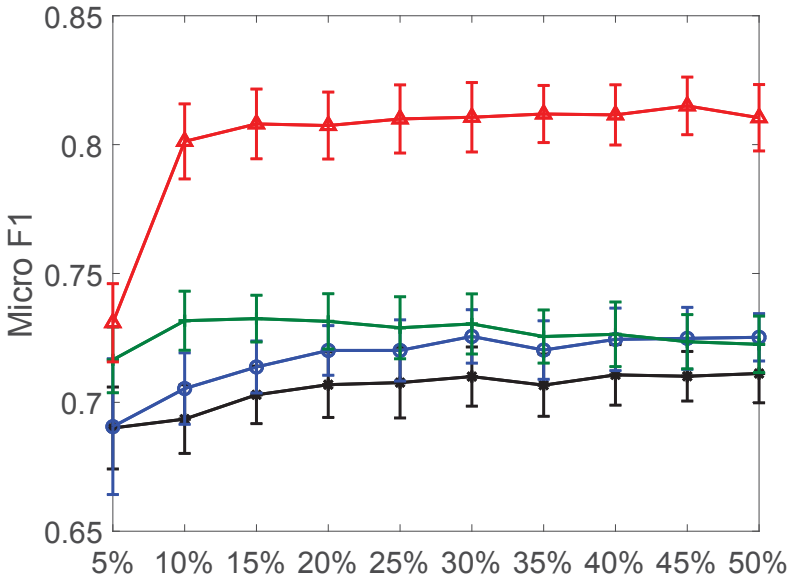

Figure 4.14: Micro F1 v.s. labeling rates for classification algorithms applied to AMPds. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

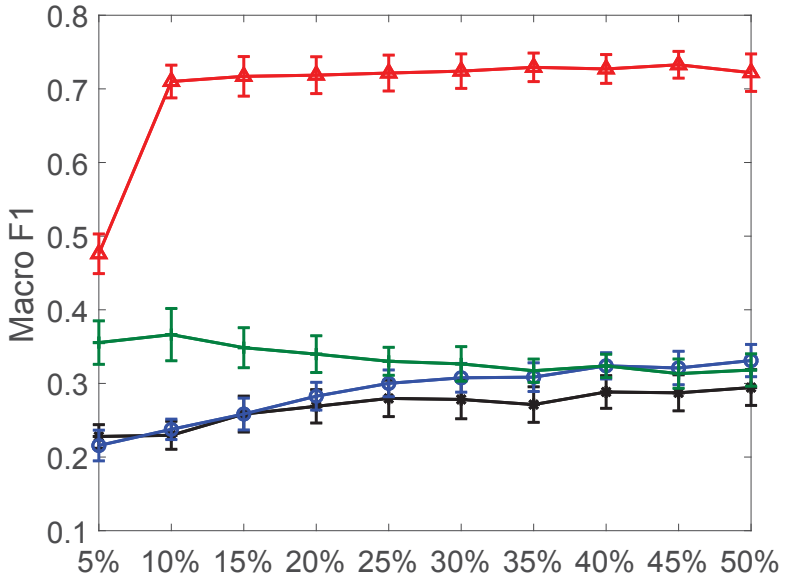

Figure 4.15: Macro F1 v.s. labeling rates for classification algorithms applied to AMPds. Four algorithms are used, including ML-GFHF (blue circles), ML-LGC (green crosses), ML-MR (red triangles), and ML-KNN (black Asterisk).

Table 4.11: The mean value and standard deviation (Std) of the running time for each method.

	ML-KNN	ML-GFHF	ML-LGC	ML-MR
Mean	26 s	489 s	291 s	741 s
Std	12 s	185 s	105 s	256 s

learning is an effective alternative to existing NIALM methods, with our best algorithm capable of outperforming state-of-the-art approaches.

4.6 Summary

We have proposed semi-supervised multi-label learning for non-intrusive appliance load monitoring. We have extended three graph-based semi-supervised learning algorithms to multi-label classification, and applied them to the NIALM problem. Our methods exploit a limited amount of ground-truth data, and a large pool of unlabeled observations, to accurately disaggregate household-level energy demand into the power draws of individual appliances. Experiments on five public energy disaggregation datasets show that our methods perform well on each one, with our best algorithm able to outperform the state-of-the-art. From the satisfying experimental results, the proposed methods could be an alternative to NIALM for practical use in demand side management programs.

Chapter 5

Manifold Regularization with Multiple Labels

As demonstrated in the previous chapter that the multi-label manifold regularization works significantly better than the other algorithms on NIALM, we therefore extend the method in a generalized case and incorporate a relliance weighting strategy to improve its performance.

5.1 Overview

This section extends a graph-based manifold regularization algorithm in the framework of semi-supervised learning for multi-label case. Specifically, it firstly constructs an adjacency graph over all the data points including the labeled data and unlabeled data with some similarity measure or distance measure. In addition to the empirical cost over the labeled data, it adds two regularization terms. The first regularization term controls the complexity of the function in the Reproducing Kernel Hilbert Space (RKHS). The second regularization term is the manifold regularization which uses the geometric structure of the data by assuming that points come from the same marginal distribution $p(\mathbf{x})$ should have the same conditional distribution (label distribution) $p(\mathbf{y}_i|\mathbf{x}), i = 1, 2, \cdots, \mathcal{L}$. Namely, functions for classification should vary smoothly along the geodesics of the geometry of $p(\mathbf{x})$. Moreover, a weighting strategy is proposed to trust the labeled data more than the predictions of the unlabeled training data for forecasting labels for future instances. And extensive experiments show that the proposed algorithm is effective in using unlabeled data for multi-label classification. The rest of this section

are organized as follows: Section 5.2 introduces the background of the graph based semi-superved learning. Section 5.3 presents the preliminaries, including introducing the basis and notations, regularization in reproducing Kernel Hilbert space and manifold regularization. Section 5.4 presents the proposed approach, including graph construction, manifold regularization with multiple labels and weighting strategy for trusting labeled data more for labeling future data points. Section 5.5 illustrates the experimental design including introducing the data sets, experimental setup, performance metrics and significance test. Section 5.6 presents experimental results and gives discussion. Section 5.7 concludes this work.

5.2 Background

In many real world applications, such as bioinformatics and video annotation, obtaining labeled data is sometimes very difficult, expensive and time-consuming since it requires domain experts to manually label the data. However, obtaining unlabeled data is quite easily and cheaply. For instance, there exists a lot of unlabeled videos or images on the web. Therefore there exists a large amount of unlabeled data and a limited number of labeled data. The large amount of unlabeled data could tell us useful information about the data, e.g., estimating the distribution of the data as well as the data structure[167]. Therefore, the semi-supervised learning becomes more and more popular in the machine learning community that people use unlabeled data to improve the learning performance [31].

Semi-Supervised Learning (SSL) has received extensive studies in the literature [33, 35, 17, 133, 18, 165, 123, 163, 99]. A comprehensive review on the semi-supervised learning was provided in [164], and an older survey on the learning problems with labeled and unlabeled data was presented in [120]. The common purpose of semi-supervised algorithms lies in exploiting the data structure composed of labeled data and unlabeled data to improve the learning performance. According to [31], self-training (also known as self-learning or self-labeling) is among the earliest approaches that use unlabeled data in classification. The idea of the self-training firstly appeared in [119]. In self-training, a classifier is firstly trained only with the labeled data, and then used to predict labels for unlabeled data. The process of exploiting the previous prediction to teach itself is repeated until all the unlabeled data is labeled.

As revealed in [120], the expectation-maximization (EM) [38] can be applied naturally to SSL as it can be used on joint models to train on labeled and unlabeled data. Co-Training [22] is a learning paradigm to address problems with strong structural prior knowledge available, and is regarded as a variant of EM on the probabilistic model [120, 31]. It assumes that features can be split into two complementary and independent feature subsets and each feature subset is enough to train a classifier for the data. Then each classifier uses its most confidently predicted points and their labels to teach the other classifier. The process of using the other classifier's most confidently predicted labels to teach itself is iterated until some criteria is achieved. Another framework of the SSL is the transductive learning based on the idea of performing predictions only for test samples [31]. Transductive Support Vector Machines (TSVM) is such an early work on the transductive learning [143]. Various extensions have been proposed [41, 32, 159, 27]; the common point is that the algorithms try to learn a hyperplane over the labeled data and the unlabeled data by optimizing a tradeoff between maximizing the margin over the labeled data and regularizing the decision boundary over low-density regions of all data samples.

Graph-based semi-supervised learning is an important sub-class of the SSL, and has drawn plentiful attentions in the recent past [31, 131, 130]. Various graph-based SSL algorithms have been developed [156, 68, 141, 151, 166, 21, 144, 75, 161, 19] and a number of successful applications can be found in recent publications [158, 6, 88, 89]. Some popular graph based algorithms include the Local and global consistency[161], Gaussian random fields and harmonic functions[166], mincuts[21], greedy max-cut[144], and spectral graph transducer[75]. All the graph based algorithms start with constructing a graph with nodes and edges. each node is represented by a sample including the labeled data and the unlabeled data. Edges connecting nodes and the weight of each edge is calculated by some similarity measure or distance measure representing the degree of similarity between two nodes. The labeled data are then used to perform graph clustering or propagate labels from labeled points to unlabeled points by minimizing the empirical cost over labeled data and regularizing the smoothness over the graph using all the data. Another representative semi-supervised learning is the manifold regularization [19], which assumes data on a low-dimensional manifold in the input space and recently

has attracted plentiful attentions in academia [105, 49, 132].

Most existing algorithms aim for problems with binary labels. However, in many application domains, like image classification, bioinformatics and news categorization, each instance contains more than one concept simultaneously, each concept is a label, which means that each instance associates with multiple labels simultaneously. Thus binary classification algorithms are unable to work for them. This work extends a semi-supervised graph-based binary algorithm for multi-label case and incorporates a reliance weighting strategy to improve the classification performance. A complete review of algorithms for multi-label learning can be found in[155][126].

5.3 Preliminaries on Manifold Regularization

This section presents the notations and basics that are used throughout this section, and revisits the basic principle of the manifold regularization.

5.3.1 Basics and Notations

In the framework of semi-supervised learning, the data set $\mathbb{D}$ in the training phase consists of two parts, namely, $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$, where $\mathbb{D}_l$ and $\mathbb{D}_u$ indicate the labeled and unlabeled training data sets, respectively. Both $\mathbb{D}_l$ and $\mathbb{D}_u$ are drawn from the same distribution $p(\mathbf{x})$, where $\mathbf{x}$ indicates a feature variable. In the case with single label, the feature space and label space of a data set $\mathbb{D}$ are denoted by $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$, respectively. Then, the labeled and unlabeled training data sets are represented by $\mathbb{D}_l = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, 2, \cdots, l\}$ and $\mathbb{D}_u = \{\mathbf{x}_i : \mathbf{x}_i \in \mathcal{X}, i = l+1, l+2, \cdots, l+u\}$, where l and u indicate the numbers of labeled and unlabeled instances. The total number of all training instances in $\mathbb{D}$ is $n = l + u$. The feature instance is $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^T$ for $i = 1, 2, \cdots, n$, where d indicates the feature dimension. The goal of semi-supervised learning with single label is to infer the labels $\tilde{Y} = \{\tilde{y}_i \in \mathcal{Y}, i = 1, 2, \cdots, e\}$ for future instances $\mathbb{D}_e = \{\tilde{\mathbf{x}}_i \in \mathcal{X}, i = 1, 2, \cdots, e\}$ given the training data set $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$ [167, 131].

In the case with multiple labels, the label space of $\mathbb{D}$ is denoted by $\mathcal{Y} = \{-1, 1\}^L$, where L indicates the number of labels. Analogously, the labeled training data set becomes $\mathbb{D}_l = \{(\mathbf{x}_i, \mathbf{y}_i) : \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}, i = 1, 2, \cdots, l\}$ and the label vector is $\mathbf{y}_i = [y_{i1}, y_{i2}, \cdots, y_{iL}]^T$, whereas the other notations remain

the same as the case with single label. The goal of semi-supervised learning with multiple labels is to infer the labels $\tilde{\mathbf{Y}} = \{\tilde{\mathbf{y}}_i \in \mathcal{Y}, i = 1, 2, \cdots, e\}$ for $\mathbb{D}_e = \{\tilde{\mathbf{x}}_i \in \mathcal{X}, i = 1, 2, \cdots, e\}$ given $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$.

Using the graph-based semi-supervised learning, a crucial step is to construct a graph $\mathcal{G} = (V, E)$ representing the connections between training instances $\mathbf{x}_i \in \mathcal{X}$ [167, 151, 131]. Specifically, $\mathcal{G} = (V, E)$ has n vertices V_i and each vertex V_i represents an instance $\mathbf{x}_i, i = 1, 2, \cdots, n$. The notation E is the edge set and E_{ij} is an edge connecting vertices V_i and V_j. There are three typical methods to construct such a graph, including the k nearest neighbor algorithm, ε distance measure and full connection with a certain similarity measure. For example, using the k nearest neighbor algorithm, each edge E_{ij} connects the vertices V_i and V_j if vertex V_i is among the k nearest neighbors of vertex V_j, or vertex V_j is among the k nearest neighbors of vertex V_i. A weight matrix $\mathbf{W}$ is formed over the graph $\mathcal{G} = (V, E)$. W_{ij} is the weight associates with edge E_{ij} and it represents the similarity between vertices V_i and V_j (namely, the training instances $\mathbf{x}_i$ and $\mathbf{x}_j$). Then, the unnormalized graph Laplacian is given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} W_{ij}$.

The label inference of the graph-based SSL is usually based on two graph assumptions [167, 151]: 1) the prediction should be close to the given labels on labeled vertices; 2) the prediction should be smooth on the whole graph (i.e., vertices that are close in the graph tend to have the same labels). The label inference algorithms for graph-based SSL can be categorized into two major classes, including the transductive learning (e.g., the graph Laplacian regularization [166, 161]) and the inductive learning (e.g., the manifold regularization [19]). The transductive learning infers the labels only on the unlabeled data and cannot make predictions on out-of-sample data. By contrast, the inductive learning infers the labels on the whole domain, i.e., a function $f : \mathcal{X} \to \mathcal{Y}$ is learned given $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$ and then the labels for $\mathbb{D}_e$ are predicted. The work in this paper is based on the manifold regularization [19], which is a typical inductive learning method, combining the inductive ability and the geometry of the data domain [162]. The next subsection revisits the regularization in the reproducing Kernel Hilbert space that is the core of manifold regularization.

5.3.2 Regularization in Reproducing Kernel Hilbert Space

For a Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists an associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_K$ of functions $\mathcal{X} \to \mathbb{R}$ with the norm $||\cdot||_K$[118]. The standard supervised learning estimates an unknown function $f \in \mathcal{H}_K$ from the labeled data set $\mathbb{D}_l$ as

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \gamma_A ||f||_K^2, \tag{5.1}$$

where $V(\mathbf{x}_i, y_i, f)$ is the loss function, such as the squared error loss $(y_i - f(\mathbf{x}_i))^2$ for regularized least square (RLS). $||f||_K^2$ is a regularization term in the RKHS imposing the smoothness condition on possible solutions. γ_A balances the tradeoff between the empirical cost and the regularization term. l is the number of labeled instances.

The difference of semi-supervised learning to supervised learning lies in the utilization of the marginal distribution of $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$ to improve the learning performance in addition to the empirical cost obtained over the labeled data set $\mathbb{D}_l$. According to the discussions in [19], there is an identifiable relation between marginal distribution $p(\mathbf{x})$ and conditional distribution $p(y|\mathbf{x})$, i.e., if two instances $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ are close in the intrinsic geometry of $p(\mathbf{x})$, then their conditional distributions $p(y|\mathbf{x}_i)$ and $p(y|\mathbf{x}_j)$ are similar.

Another regularization term can be added to ensure that the solution is smooth with respect to the marginal distribution $p(\mathbf{x})$. Incorporating the smoothness penalty term with respect to the graph Laplacian $\mathbf{L}$, the learning problem turns out to be[19]

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \gamma_A ||f||_K^2 + \frac{\gamma_I}{n^2} \mathbf{f}^T \mathbf{L} \mathbf{f}, \tag{5.2}$$

where $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$, and $\mathbf{f}^T \mathbf{L} \mathbf{f}$ is a penalty term that reflect the intrinsic structure of $p(\mathbf{x})$, and is a smoothness penalty corresponding to the probability distribution. $n = u + l$ is the number of total instances. The normalizing coefficient $\frac{1}{n^2}$ is the natural scale factor for the empirical estimate of the Laplace operator. Coefficients γ_A and γ_I controls the complexity of the function in the ambient space and the intrinsic geometry of the $p(\mathbf{x})$ respectively. In real case, $p(\mathbf{x})$ is unknown. But we can get the empirical estimate from sufficient large amount of unlabeled data $\mathbb{D}_u$ by assuming the data

set lies on a manifold in $\mathbb{R}^d$ and modeling the manifold with the adjacency graph $\mathcal{G} = (V, E)$ from the data set $\mathbb{D}$. According to the classical Representer Theorem[118], the solution to eqn. (5.2) in $\mathcal{H}_K$ is given by[19]

$$f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \theta_i K(\mathbf{x}_i, \mathbf{x}) \tag{5.3}$$

which is an expansion of the Representer Theorem in terms of labeled data and unlabeled data $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$. Accordingly, the problem is essentially an optimization problem over the space of coefficients θ_i.

The RKHS has been extended to vector valued functions [26] and exploits to formulate the vector-valued manifold regularization [105]. Let $\mathbf{F} = (f_1(\mathbf{x}_1), \cdots, f_n(\mathbf{x}_n)) \in \mathcal{Y}^n$ be components of a vector-valued function where each $f_i \in H_K$ [105]. Here $\mathcal{Y}$ can be $\mathbb{R}$ for the single label case or $\mathbb{R}^L$ for multi-label case. The optimization problem of the vector-valued manifold regularization is given by [105]

$$\begin{aligned} f^* &= \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\mathbf{x}_i, \mathbf{y}_i, f) + \gamma_A ||f||_K^2 \\ &\quad + \gamma_I < \mathbf{F}, M\mathbf{F} >_{\mathcal{Y}^n}, \end{aligned} \tag{5.4}$$

where the matrix M is a symmetric, positive operator, such that $< y, My >_{\mathcal{Y}^n}$ for all $y \in \mathcal{Y}^n$. $\mathcal{Y}^n$ is the usual n-direct product of $\mathcal{Y}$, with the inner product

$$< (y_1, \cdots, y_n), (w_1, \cdots, w_n) >_{\mathcal{Y}^n} = \sum_{i=1}^{n} < y_i, w_i >_{\mathcal{Y}} .$$

It has been proved in [105] that the minimization problem in (5.4) has a unique solution taken the form $f^*(\mathbf{x}) = \sum_{i=1}^{l+u} K(\mathbf{x}_i, \mathbf{x}) \mathbf{\Theta}_i$ for some vectors $\mathbf{\Theta}_i \in \mathcal{Y}, 1 \leq i \leq n$. The vector-valued manifold regularization is a generalized form of manifold regularization, and can be used for single label, multi-label, and multi-view learning [105, 104].

The Representer Theorem in the vector valued RKHS is given and proved in [105]. Let $\mathcal{H}_{K,\mathbf{x}} = \{\sum_{i=1}^{u+l} K(\mathbf{x}_i, \mathbf{x}) y_i, \mathbf{y} \in \mathcal{Y}^{u+l}\}$. For $f \in \mathcal{H}_{K,\mathbf{x}}^{\perp}$, the sampling operator $S_{\mathbf{x}}$ satisfies $< S_{\mathbf{x}} f, \mathbf{y} >_{\mathcal{Y}^{u+l}} = < f, \sum_{i=1}^{u+l} K(\mathbf{x}_i, \mathbf{x}) y_i >_{\mathcal{H}_K} = 0$. This holds true for all $\mathbf{y} \in \mathcal{Y}^{u+l}$ and yields to $S_{\mathbf{x}} f = (f(\mathbf{x}_1), \cdots, f(\mathbf{x}_{u+l})) = 0$. Denote the right handside of (5.4) by $I(f)$. For any arbitrary $f \in \mathcal{H}_K$, it can be decomposed orthogonally as $f = f_0 + f_1$, with $f_0 \in \mathcal{H}_{K,\mathbf{x}}$ and $f_1 \in \mathcal{H}_{K,\mathbf{x}}^{\perp}$. This

results in $I(f) = I(f_0 + f_1) \geq I(f_0)$ with equality if and only if $||f_1||_{\mathcal{H}_K} = 0$, since $||f_0 + f_1||_{\mathcal{H}_K} = ||f_0||_{\mathcal{H}_K} + ||f_1||_{\mathcal{H}_K}$. As a result, the minimizer of (5.4) must liet in $\mathcal{H}_{K,\mathbf{x}}$.

5.4 The Proposed Method

The work in [19] initially proposed the manifold regularization and proved the Representer Theorem for univariate cases; further, reference [105] proved the Representer Theorem for the general cases of the vector manifold regression. Following the two fundamental theoretical works, this work on the extended manifold regularization is essentially an important special case of the theorem in [105]. In existing literatures, there is no study on such a special case and no proof to the corresponding more compact Representer Theorem. Nevertheless, simpler proofs for special cases are usually interesting even if the general case has been proven. Lots of such examples can be found in the community of computing science. As shown in a comprehensive suvery in [103], the Dirichlets theorem was firstly proved by [42], followed by extensive studies of special cases of the Dirichlets theorem [121, 58, 107]. Analogously, studying the important special case of the manifold regularization and showing more compact proof of the Representer Theorem are also interesting and novel contributions. This is exactly what this chapter is going to solve. In this section, a systematic study of the semi-supervised multi-label classification based on an extended manifold regularization is presented. A reliance weighting strategy is proposed to improve the performance. The method includes three components, namely, the graph construction, the manifold regularization with multiple labels, and the adoption of a reliance weighting strategy.

5.4.1 Graph Construction

Given the whole data set $\mathbb{D} = \mathbb{D}_l \cup \mathbb{D}_u$, a full $n \times n$ distance matrix $\mathbf{U}$ is calculated between each pair of instances $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ based on a Gaussian kernel; and each element U_{ij} is calculated using a Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ as

$$U_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right), \tag{5.5}$$

where σ denotes the bandwidth of the Gaussian kernel. Equivalently, an alternative distance matrix $\mathbf{H}$ can be calculated with each element H_{ij} given

by [144, 71]

$$H_{ij} = \sqrt{U_{ii} + U_{jj} - 2U_{ij}}. \tag{5.6}$$

The constructed graph $\mathcal{G} = (V, E)$ is a fully connected graph given each edge E_{ij} weighted by H_{ij}. According to [144, 71], the graph sparsification can improve the efficiency of label inference. It removes edges by recovering a full $n \times n$ binary matrix $\mathbf{B}$ with 1's and 0's representing the presence and absence of connections, respectively. Three sparsification approaches can be used, including the ε-neighbor search, k-nearest neighbor search, and the b-matching [144, 71]:

1. The ε-neighbor search recovers a binary matrix $\mathbf{B}$ as

$$B_{ij} = \begin{cases} 1 & \text{if} \quad 1 - H_{ij} \leq \varepsilon \\ 0 & \text{if} \quad 1 - H_{ij} > \varepsilon \text{ or } i = j \end{cases} \tag{5.7}$$

2. The k-nearest neighbor search obtains the binary matrix $\mathbf{B}$ by minimizing the following optimization problem

$$\begin{aligned} &\min_{\mathbf{B} \in \{0,1\}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} B_{ij} H_{ij} \\ &s.t. \sum_{j=1}^{n} B_{ij} = k, B_{ii} = 0, \forall i, j = 1, \cdots, n. \end{aligned} \tag{5.8}$$

3. Using the b-matching algorithm, the optimization problem to recover $\mathbf{B}$ is

$$\begin{aligned} &\min_{\mathbf{B} \in \{0,1\}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} B_{ij} H_{ij} \\ &s.t. \sum_{j=1}^{N} B_{ij} = b, B_{ii} = 0, B_{ij} = B_{ji}, \forall i, j = 1, \cdots, n \end{aligned} \tag{5.9}$$

The binary matrix $\mathbf{B}$ obtained using the k-nearest neighbor search is not symmetric; thus the final $\mathbf{B}$ can be calculated as $B_{ij} = \max(B_{ij}, B_{ji})$. By contrast, the b-matching algorithm produces a graph with every node having the same number of neighbors, namely, $\mathbf{B} = \mathbf{B}^T$. Whichever of the above methods is applied, the graph $\mathcal{G} = (V, E)$ is sparsified by removing the edges, i.e., the weight W_{ij} for the edge E_{ij} is assigned with 0 if $B_{ij} = 0$. For an

edge E_{ij} with $B_{ij} = 1$, the weight W_{ij} can be calculated with respect to the distance matrix $\mathbf{H}$ and expressed as

$$W_{ij} = H_{ij} B_{ij} \tag{5.10}$$

The final graph $\mathcal{G} = (V, E)$ is then constructed and represented by a sparse weight matrix $\mathbf{W}$. Proceeding to label inference, the graph Laplacian is calculated as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where each element of $\mathbf{D}$ is $D_{ii} = \sum_{j=1}^{N} W_{ij}$ and $D_{ij} = 0$.

5.4.2 Manifold Regularization with Multiple Labels

In this subsection, we extend the manifold regularization in [19] to solve multi-label learning problem. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]^T$ denote the matrix of all feature instances and label instance. In $\mathbf{Y}$, $\mathbf{y}_i$ for $i \leq l$ takes 1 or -1 for its elements and $\mathbf{y}_i$ is an all-zero vector for $l < i \leq n$. In the framework of the Laplacian Regularized Least Squares (LapRLS) [19], the optimization problem of manifold regularization with multiple labels is

$$\begin{aligned} f^* \quad = \quad & \underset{f_j \in \mathcal{H}_K, j=1,\cdots,L}{\arg\min} \frac{1}{l} \operatorname{tr}\left((\mathbf{\Psi F} - \mathbf{Y})^T (\mathbf{\Psi F} - \mathbf{Y})\right) \\ & + \gamma_A ||f||_K^2 + \frac{\gamma_I}{n^2} \operatorname{tr}\left(\mathbf{F}^T \mathbf{L F}\right), \end{aligned} \tag{5.11}$$

where $\mathbf{F} = [f_j(\mathbf{x}_i)]_{n \times L}, i = 1, \cdots, n, j = 1, \cdots, L$ is a matrix representing the predicted outputs, $\operatorname{tr}(\cdot)$ denotes the trace of a matrix, and $\mathbf{\Psi}$ is a $n \times n$ diagonal matrix with the diagonal elements given by

$$\Psi_{ii} = \begin{cases} 1 & \text{for} \quad i \leq l, \\ 0 & \text{for} \quad l < i \leq n. \end{cases} \tag{5.12}$$

The second term $||f||_K^2 = \sum_{j=1}^{L} ||f_j||_K^2$ in eqn. (5.11) measures the complexity of $\mathbf{F}$ in the ambient space. The third term represents the intrinsic smoothness with respect to the geometric distribution. $\mathbf{L}$ is the graph Laplacian obtained in the graph construction phase. The optimization problem in (5.11) is essentially one natural extension of the LapRLS for multi-label cases as indicated in [105].

The minimization problem in eqn.(5.11) is guaranteed to have a unique global solution. The theorem for the solution in (5.11) are given and proved as follows.

Theorem 1. *The minimizer of optimization problem in eqn.(5.11) admits an expansion*

$$f_j^*(\mathbf{x}) = \sum_{i=1}^{n} \Theta_{ij} K(\mathbf{x}_i, \mathbf{x}), j = 1, 2, \cdots, L \tag{5.13}$$

in terms of the labeled and unlabeled instances; $K(\cdot, \cdot)$ represents the kernel function, which must be positive semi-definite.

Proof. In the multi-label classification problem (5.11), the norm of the function f can be represented by the sum of each function f_j in the Reproducing Kernel Hilbert Space $\mathcal{H}_K$, i.e., $||f||_K^2 = \sum_{j=1}^{L} ||f_j||_K^2$.

Given any function in the Reproducing Kernel Hilbert Space $\mathcal{H}_K$, it can be decomposed into two orthogonal components; specifically given each f_j, it can be decomposed to a function f_j^0 in the linear subspace spanned by $\{K(x_i, \cdot)\}_{i=1}^n$ and f_j^1 orthogonal to f_j^0 [19]. Accordingly, f_j can be represented by

$$f_j = f_j^0 + f_j^1 = \sum_{i=1}^{n} \Theta_{ij} K(x_i, \cdot) + f_j^1,$$

Since $||f_j||_K^2 = ||f_j^0||_K^2 + ||f_j^1||_K^2 \geq ||f_j^0||_K^2$, there is

$$||f||_K^2 = \sum_{j=1}^{L} ||f_j||_K^2 = \sum_{j=1}^{L} ||f_j^0||_K^2 + \sum_{j=1}^{L} ||f_j^1||_K^2 \geq \sum_{j=1}^{L} ||f_j^0||_K^2$$

The equality is achieved if and only if $||f_j^1||_K^2 = 0$, $j = 1, 2, \cdots, L$. Therefore the minimizer must be $f_j^*(\mathbf{x}) = \sum_{i=1}^{n} \Theta_{ij} K(\mathbf{x}_i, \mathbf{x})$, $j = 1, 2, \cdots, L$. □

Denote the $\mathbf{K}$ as a $n \times n$ matrix of the kernel estimation with respect to all the data samples $\mathbf{X}$, and $\mathbf{\Theta}$ as a $n \times L$ matrix of the coefficients. The solution can be represented by

$$\mathbf{F}^* = \mathbf{K}\mathbf{\Theta}. \tag{5.14}$$

Therefore, the problem in eqn.(5.11) is reduced to optimizing over the finite dimensional space of coefficients $\mathbf{\Theta}$. According to [19], the kernel function $K(\cdot, \cdot)$ must be positive semi-definite which gives rise to an RKHS. A choice of the kernel function is the heat kernel, which can be approximated using a sharp Gaussian kernel. Thus, $\mathbf{U}$ in eqn. (5.5) can be taken as the kernel matrix $\mathbf{K}$.

5.4.3 Reliance Weighted Kernel for Performance Improvement

In the framework of manifold regularization, the classifier is trained using both labeled training set $\mathbb{D}_l$ and the unlabeled training set $\mathbb{D}_u$. Although both $\mathbb{D}_l$ and $\mathbb{D}_u$ contribute to the classification, the prediction of the label vector $\tilde{\mathbf{y}}$ of an unforeseen future sample $\tilde{\mathbf{x}}$ is based on the label information provided by the labeled training set $\mathbb{D}_l$. Naturally, this motivates us to have more trust to the labeled training set than the unlabeled training set for label referring. Thus, a reliance weighting strategy is proposed to assign different weights to the training instances, so as to make $\mathbb{D}_l$ have more impacts to label inference than $\mathbb{D}_u$. Given a heat kernel function $K(\mathbf{x}_i, \mathbf{x})$, the weighted kernel function for $\mathbf{x}$ is

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}) = K(\mathbf{x}_i, \mathbf{x}) \cdot \Xi_i \tag{5.15}$$

where Ξ_i represents the reliance weights of the ith instance. Denote the $\tilde{\mathbf{K}}$ as the matrix of the weighted kernel estimation with respect to all the data samples $\mathbf{X}$, and the reliance weight matrix $\boldsymbol{\Xi}$ as

$$\boldsymbol{\Xi} = \begin{bmatrix} \Xi_1 & 0 & \cdots & 0 \\ 0 & \Xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Xi_n \end{bmatrix} \tag{5.16}$$

Then, the weighted kernel matrix is $\tilde{\mathbf{K}} = \mathbf{K}\boldsymbol{\Xi}$. In order to yield to the minimizer in (5.13), the kernel function $\tilde{K}(\cdot, \cdot)$ must be positive semi-definite.

Proposition 1. *Given a heat kernel function $K(\cdot, \cdot)$, the weighted kernel $\tilde{K}(\cdot, \cdot) = K(\cdot, \cdot) \cdot \Xi_i$ is positive semi-definite if and only if $\Xi_i \geq 0$.*

Proof. Given an arbitrary vector $\mathbf{v} \in \mathbb{R}^d$, we have

$$\mathbf{v}^T \tilde{\mathbf{K}} \mathbf{v} = \sum_{i=1}^{d} \sum_{j=1}^{d} K(\mathbf{x}_i, \mathbf{x}_j) \cdot \Xi_i \cdot v_i v_j. \tag{5.17}$$

where v_i and v_j are the ith and jth elements of $\mathbf{v}$. The kernel estimation based on a heat kernel function is always nonnegative, namely, $K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$. Therefore, $K(\mathbf{x}_i, \mathbf{x}_j) \cdot \Xi_i \geq 0$ if and only if $\Xi_i \geq 0$. Accordingly, $\mathbf{v}^T \tilde{\mathbf{K}} \mathbf{v} \geq 0$ if and only if $\Xi_i \geq 0$. As a conclusion, the weighted kernel $\tilde{K}(\cdot, \cdot) = K(\cdot, \cdot) \cdot \Xi_i$ is positive semi-definite if and only if $\Xi_i \geq 0$. □

Using the reliance weighted kernel function instead of the heat kernel function, the solution in (5.14) can be transferred to be

$$\mathbf{F}^* = \tilde{\mathbf{K}}\boldsymbol{\Theta} = \mathbf{K}\boldsymbol{\Xi}\boldsymbol{\Theta}. \tag{5.18}$$

The coefficient matrix $\boldsymbol{\Theta}^*$ can be estimated by differentiating the righthand side of (5.11) as

$$\begin{aligned}\frac{2}{l}\boldsymbol{\Psi}\mathbf{K}\boldsymbol{\Xi}(\boldsymbol{\Psi}\mathbf{K}\boldsymbol{\Xi}\boldsymbol{\Theta}^* - \mathbf{Y}) + 2\gamma_A\mathbf{K}\boldsymbol{\Xi}\boldsymbol{\Theta}^* \\ + \frac{2\gamma_I}{n^2}(\mathbf{K}\boldsymbol{\Xi})^T\mathbf{L}\mathbf{K}\boldsymbol{\Xi}\boldsymbol{\Theta}^* = 0\end{aligned}$$

The coefficients matrix is eventually obtained as

$$\boldsymbol{\Theta}^* = \left(\boldsymbol{\Psi}\mathbf{K}\boldsymbol{\Xi} + l\gamma_A\mathbf{I} + \frac{l\gamma_I}{n^2}\mathbf{L}\mathbf{K}\boldsymbol{\Xi}\right)^{-1}\mathbf{Y}. \tag{5.19}$$

where $\mathbf{I}$ is a $n \times n$ identity matrix.

For unforeseen future samples $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_e]^T$ in $\mathbb{D}_e$, the label matrix $\tilde{\mathbf{F}}$ is obtained as follows: firstly, a $e \times n$ kernel matrix $\mathbf{K}_e$ is calculated using eqn. (4.1), i.e., $\tilde{K}_{ij} = K(\tilde{\mathbf{x}}_i, \mathbf{x}_j)$ for $i = 1, 2, \cdots, e$ and $j = 1, 2, \cdots, n$. Next, the output $\tilde{\mathbf{F}}$ for $\tilde{\mathbf{X}}$ can be calculated as

$$\tilde{\mathbf{F}} = \mathbf{K}_e\boldsymbol{\Xi}\boldsymbol{\Theta}^*. \tag{5.20}$$

Eventually, the label matrix $\tilde{\mathbf{Y}}$ of $\tilde{\mathbf{X}}$ is obtained by comparing each element of $\tilde{\mathbf{F}}$ with 0. In this paper, the learning method with a general positive semi-definite kernel function is called the Multi-Label Manifold Regularization (ML-MR), and that with a reliance weighted positive semi-definite kernel function is called the ML-MR with the Reliance Weighting strategy (ML-MRRW). In fact, the ML-MRRW is reduced to ML-MR when the reliance matrix $\boldsymbol{\Xi}$ is an identity matrix.

The setting of reliance weights can be based on prior knowledge. A choice is to assign uniform weights, namely, $\Xi_i = \nu_1 \in [0, 1], 1 \leq i \leq l$ and $\Xi_i = \nu_2 \in [0, 1], l < i \leq l+u$ for all labeled and unlabeled training instances, respectively. These two parameters decide the balance of impacts to label inference from labeled and unlabeled training data. The extended manifold regularization is supervised if $\nu_1 = 1$ and $\nu_2 = 0$ are used, and is unsupervised for the choice of $\nu_1 = 0$ and $\nu_2 = 1$. The relation $\nu_1 = \nu_2$ indicates that the impacts of $\mathbb{D}_l$ and $\mathbb{D}_u$ to label inference are equal, whereas $\nu_1 > \nu_2$ indicates that more weight

is put on labeled instances $\mathbb{D}_l$ than that on unlabeled instances $\mathbb{D}_u$. In this work, we are trying to improve the performance of manifold regularization by trusting more on labeled instances than unlabeled instances when predicting for future instances, thus the choices of ν_1 and ν_2 must follow two criterions, namely, $\nu_1 = 1$ and $\nu_1 > \nu_2 > 0$.

5.5 Experimental Design

This section designs experiments to validate the effectiveness of the proposed method in semi-supervised learning with multiple labels. Some commonly used benchmark data sets are selected. Other semi-supervised multi-label classification methods are exploited for comparisons. Performance metrics for multi-label classification are introduced.

5.5.1 Datasets

Four public data sets from different domains are chosen for the experimental study. Table 5.1 presents the basic information about these data sets. The first data set "Emotions" [138] consists of sampled wave forms of sound clips generated from different genres of musical songs; and each instance is labeled by 6 emotions, including the amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-aggressive emotions. The second data set "Scene" [25] is a commonly used image data set with each image represented by a 294-dimension feature vector and labeled by 6 classes, including the beach, sunset, field, fall-foliage, mountain, and urban. The third data set "Yeast" [47] is formed by micro-array expression data and phylogenetic profiles with 2107 genes. Each gene is associated with a set of functional classes, which are grouped into 14 functional categories. The last data set "mediamill" [125] consists of digital video achieves for the TREC Video Retrieval Evaluation (TRECVID) challenge; and this data contains 120-dimensional visual features and 101 annotation concepts. These data sets are properly formatted and represented as features and labels. So there is no further pre-processing with these data sets.

Table 5.1: Basic information of the selected public data sets.

Data set	Domain	# Features	# labels	# instances
Emotions [138]	Music	72	6	593
Scene [25]	Image	294	6	2409
Yeast [47]	Life	103	14	2417
Mediamill [125]	Video	120	101	43907

5.5.2 Experiment Setup

The experiments are carried out to test the performance of the extended graph-based manifold regularization algorithm for multi-label semi-supervised case, and compare it with some well-known semi-supervised multi-label classification algorithms. In each experiment, the data set is firstly partitioned into two parts: the training data and out-of-sample testing data occupy two third and one third of the whole data set, respectively. Then, the labels of a portion of the instances in the training data are omitted to construct labeled training data and unlabeled training data. The labeling rate η varies from {5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%}. For each labeling rate, experiments are conducted 100 times by randomly resampling the labeled training data, unlabeled training data, and out-of-sample testing data. The first three datasets "Emotions", "Scene", and "Yeast" are fully used in the experiments, whereas only a portion (10% randomly selected) of the "Mediamill" data is used in view of the large amount of repetitive experiments.

In each experiment, six algorithms are applied: 1) the Multi-Label Manifold Regularization (ML-MR), 2) the ML-MR with the Reliance Weighting strategy (ML-MRRW) in Section 5.4.3, 3) the Multi-Label Local and Global Consistency (ML-LGC)[151], 4) the Multi-Label Gaussian Fields and Harmonic Functions (ML-GFHF)[151], 5) the Fixed-Size Multi-Label Regularized Kernel Spectral Clustering (ML-FSKSC)[102], and 6) the Multi-Label k Nearest Neighbors (MLkNN) [154]. Among all of the algorithms, MLkNN is supervised and all the other algorithms are semi-supervised. Accordingly, the MLkNN algorithm only uses the labeled training data in the training phase, whereas all the other algorithms exploits both the labeled training data and unlabeled training data. The parameters in each algorithm are determined by parameter exploration using a small portion of the data. For the ML-MRRW algorithm, a good choice of two parameters for the reliance weighting strategy

is $[\nu_1, \nu_2] = [1, 0.1]$.

5.5.3 Performance Metrics

Many performance metrics or criteria for multi-label classification have been proposed and a complete metrics is given in [126] and [155]. In this work, three popular metrics are used to evaluate the performance of algorithms in learning multi-label problems.

The average precision calculates the average fraction of labels ranked above a particular label that are truly predicted. The larger the value of it, the better the learning performance.

$$\begin{aligned} \text{avgprec}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\mathbf{y_i}|}\sum_{y_{ij}\in\mathbf{y_i}} \\ \frac{|\{y'_{ij}|\text{rank}_f(\mathbf{x}_i, y'_{ij}) \leq \text{rank}_f(\mathbf{x}_i, y_{ij}), y'_{ij} \in \mathbf{y}_i\}|}{\text{rank}_f(\mathbf{x}_i, y_{ij})} \end{aligned} \tag{5.21}$$

where y'_i is the chosen particular label. y_{ij} is the jth label of instance i.

$F1$ is a popular measure for single label. It is the harmonic mean of precision and recall.

$$F1 = \frac{2 \times tp}{2 \times tp + fp + fn} \tag{5.22}$$

where tp is the number of true positives, tn is the number of true negatives, fp is the number of false positives, and fn is the number of false negatives. Macro $F1$ and micro $F1$ are multi-label classifier metrics derived by computing the $F1$ measure across the label set; either after summing true and false positives and false negatives across all labels, or by averaging the $F1$ measure for each label.

$$F1_{micro} = F1\left(\sum_{\lambda=1}^{L} tp_\lambda, \sum_{\lambda=1}^{L} fp_\lambda, \sum_{\lambda=1}^{L} fn_\lambda\right) \tag{5.23}$$

$$F1_{macro} = \frac{1}{L}\sum_{\lambda=1}^{L} F1\left(tp_\lambda, fp_\lambda, fn_\lambda\right) \tag{5.24}$$

where tp_λ is the number of true positives, fp_λ is the number of false positives, and fn_λ is the number of false negatives of label λ after being evaluated by binary evaluation of $F1$. Larger values of $F1_{micro}$ and $F1_{macro}$ denote better performance.

5.5.4 Significance Test

Statistical test is usually used for comparisons of algorithms in machine learning community to verify that there is a significant difference among the algorithms statistically[39, 56, 122]. In this paper, Friedman test and a post hoc test are utilized to verify our proposed algorithm outperforming the other algorithms. Friedman's Test is a simple and robust nonparametric method for testing the differences between multiple algorithms over multiple data sets. It ranks the algorithms from the smallest rank to the largest rank based on their performance value for each data set separately, and average ranks are assigned to ties. The Friedman's statistic F_R[54] is given by

$$F_R = \frac{12}{NK(K+1)} \sum_{i=1}^{K} R_i^2 - 3N(K+1) \tag{5.25}$$

where K is the number of algorithms applied on N different data sets, R_i is the rank sum of the ith algorithm, $i = 1, 2, 3, \cdots, K$. The null hypothesis H_0 is that there are no significant differences between the algorithms, the alternative hypothesis H_1 is that there are significant differences between the algorithms. F_R tests the null hypothesis H_0 against the alternative hypothesis H_1. For K is larger than 5, the statistical test F_R can be approximated by a chi-square distribution with $K-1$ degree of freedom. Thus, for any pre-chosen α level of significance, a critical chi-square value χ^2_α is also determined by looking up the Chi-Square table in terms of the degree of freedom $K-1$ and level of significance α. The null hypothesis H_0 is rejected if $F_R > \chi^2_\alpha$; otherwise, do not reject H_0. In this paper, there are 6 algorithms, leading to the degree of freedom $K-1=5$. Thus, the critical chi-square value is $\chi^2_\alpha = 12.5916$ under the level of significance $\alpha = 0.05$.

When the null hypothesis is rejected, the analysis proceeds with a post hoc test[39, 56, 122]. Denote the difference $D_{ij} = R_i - R_j$ between the rank sums of algorithms i and j. The performance of two algorithms is significantly different if the difference $|D_{ij}|$ between their corresponding rank sums is no less than the *critical difference*

$$CD = z\sqrt{\frac{NK(K+1)}{6}} \tag{5.26}$$

where z is the z-score from the standard normal curve corresponding to $\frac{\alpha}{K(K-1)}$, and α is the level of significance. It can be concluded that the performance of

the algorithm i is significantly better than that of the algorithm j, if $|D_{ij}| \geq CD$ and $D_{ij} < 0$; otherwise, worse, if $|D_{ij}| \geq CD$ and $D_{ij} > 0$.

5.6 Experimental Results and Discussion

To evaluate the performance of the proposed algorithm, experiments are conducted using proposed ML-MR and ML-MRRW against three famous semi-supervised multi-label algorithms and a famous supervised multi-label algorithm on the public datasets. When calculating the friedman's statistic test and post hoc statistic test for each data set, the sampled data sets under each labeling rate (from 5% to 50%) are considered as different data sets, thus, there are 10 different data sets to calculate the statistical significance differences between the algorithms for each data set.

5.6.1 Case I: Emotions

The experimental results for the "Emotions" data are shown in Figs. 5.1, 5.2, and 5.3. The sub-figures from left to right present the average precision, Micro F1, and Macro F1, respectively for all the algorithms under different labeling rates respectively. The error bars indicate one standard deviations of the metrics. Table 5.2 presents the Friedman's statistics F_R for the three different performance metrics. It can be found that all of them are greater than the critical chi-square value $\chi^2_\alpha = 12.5916$. Thus, the null hypothesis is rejected, and it can be concluded that there is significant difference between performances of the 6 algorithms. Further, the difference between the rank sums of the ML-MRRW and the other algorithm is calculated and presented in Table 5.3. Denote MLkNN, ML-GFHF, ML-LGC, ML-FSKSC, ML-MR by algorithm 1, 2, 3, 4, and 5, respectively. Then, $D_{6i}, i = 1, 2, \cdots, 5$ represents the difference between rank sums of the ML-NRRW and the ith algorithm. The critical difference for $K = 6$ and $\alpha = 0.05$ is $CD = 7.77$. For each performance metric, any difference value $|D_i| \geq CD$ indicates the significant difference between ML-MRRW and the algorithm i with respect to this metric. Further, $|D_i| \geq CD$ and $D_i < 0$ indicate ML-MRRW outperforms the algorithm i with respect to the metric in the corresponding column of the Table 5.3.

The following conclusions can be drawn from the plots and tables:

Table 5.2: The Friedman's statistics F_R for different performance metrics in Case I.

	A-precision	Micro-F1	Macro-F1
F_R	39.6571	37.1429	38.5714

Table 5.3: The difference between the rank sums of the ML-MRRW and each other algorithm in Case I.

	A-precision	Micro-F1	Macro-F1
D_{61}	-38	-42	-45
D_{62}	-9	-25	-30
D_{63}	-13	-1	-5
D_{64}	-42	-26	-20
D_{65}	-24	-20	-20

1. The five semi-supervised multi-label learning algorithms show much better overall performance compared to the MLkNN method, except that ML-FSKSC has lower average precision for large labeling rates.

2. The ML-MRRW algorithm has the highest average precision, Micro F1, and Macro F1 among the semi-supervised multi-label learning algorithms, for most labeling rates.

3. Overall, ML-MRRW, namely, the extended ML-MR incorporating the reliable weighting strategy outperforms all the other algorithms and shows significant superior performance.

Moreover, ML-MRRW is also compared with an extensive supervised multi-label algorithms from the state-of-the-art literature [95] on the "Emotions" data in Table 5.4. The last column presents the mean values of the average precision, Micro F1, and Macro F1 for ML-MRRW under the labeling rate 50%. From the Table 5.4, ML-MRRW outperforms most algorithms in terms of A-precision (A-precision stands for average precision), Macro-F1 and Micro-F1.

5.6.2 Case II: Scene

The experimental results for the "Scene" data are shown in Figs. 5.4, 5.5, and 5.6. Table 5.5 presents the Friedman's statistics F_R for the three different performance metrics. It can be found that all of them are greater

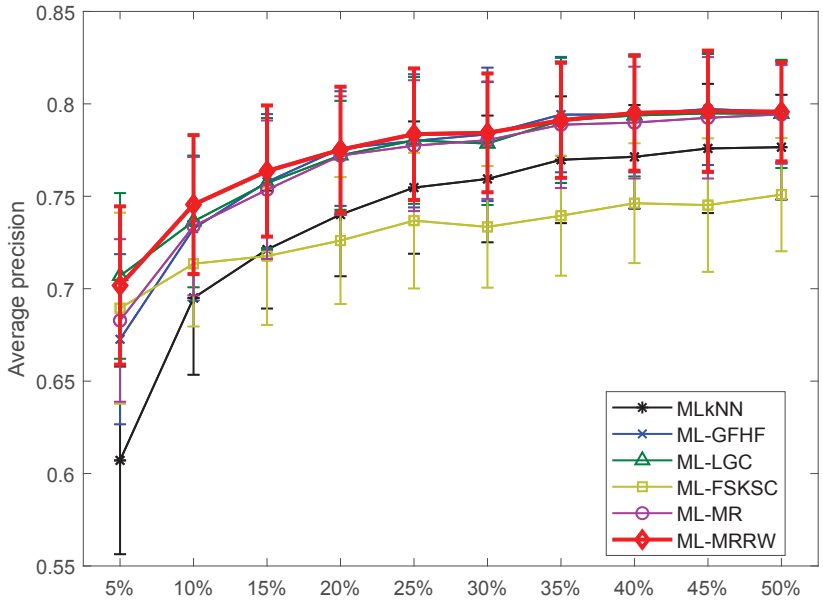

Figure 5.1: Average precision v.s. labeling rates for six classification algorithms applied to the "Emotions" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

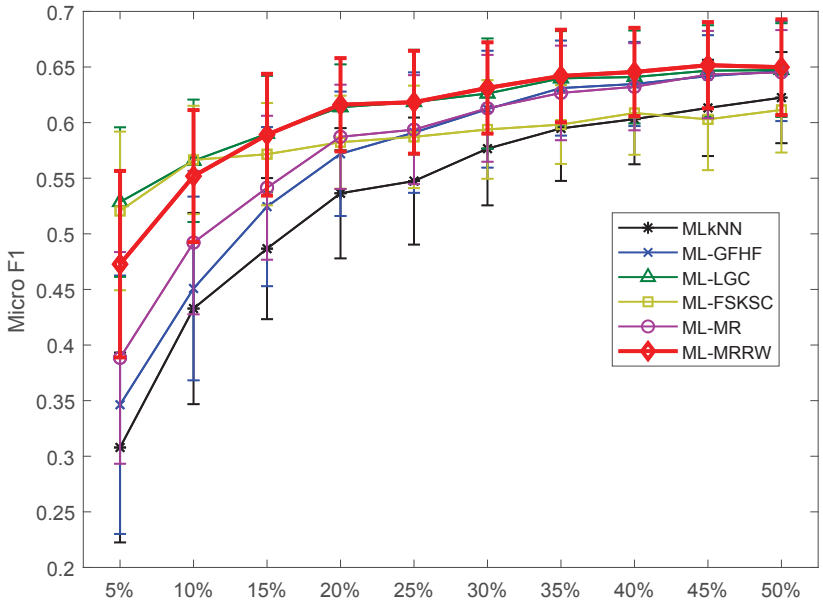

Figure 5.2: Micro F1 v.s. labeling rates for six classification algorithms applied to the "Emotions" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

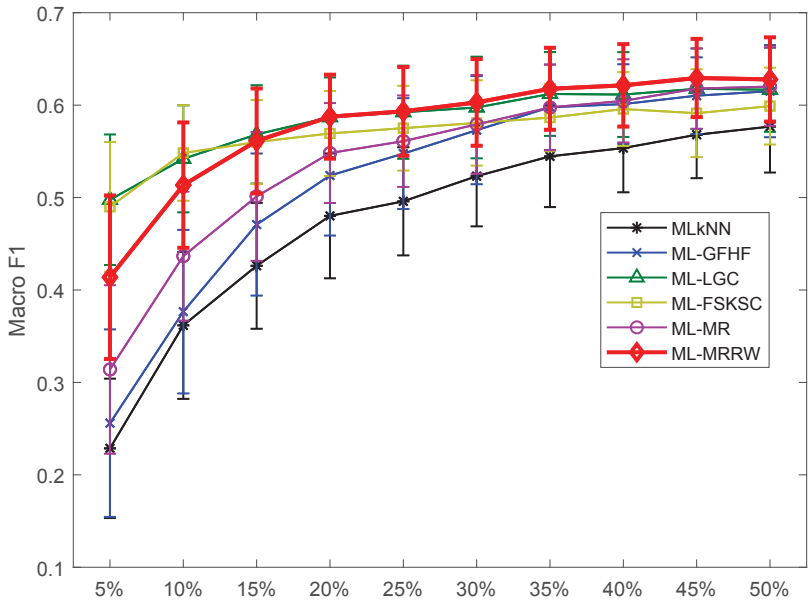

Figure 5.3: Macro F1 v.s. labeling rates for six classification algorithms applied to the "Emotions" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

Table 5.4: Comparison with the state-of-the-art literature [95] on the "Emotions" data.

	BR	CC	CLR	QWML	HOMER
A-precision	0.721	0.724	0.718	0.679	0.698
Micro-F1	0.509	0.503	0.512	0.528	0.588
Macro-F1	0.440	0.420	0.443	0.458	0.570
	ML-C4.5	PCT	ML-KNN	RAKEL	ECC
A-precision	0.759	0.713	0.649	0.713	0.687
Micro-F1	0.655	0.571	0.457	0.533	0.554
Macro-F1	0.630	0.568	0.385	0.488	0.500
	RFML-C4.5	RF-PCT	**ML-MRRW**		
A-precision	0.812	0.812	0.796		
Micro-F1	0.647	0.672	0.650		
Macro-F1	0.620	0.650	0.628		

Table 5.5: The Friedman's statistics F_R for different performance metrics in Case II.

	A-precision	Micro-F1	Macro-F1
F_R	42	40.7429	41.6000

Table 5.6: The difference between the rank sums of the ML-MRRW and each other algorithm in Case II.

	A-precision	Micro-F1	Macro-F1
D_{61}	-45	-21	-18
D_{62}	-39	-43	-40
D_{63}	-15	-20	-17
D_{64}	-27	2	8
D_{65}	-12	-26	-23

than the critical chi-square value $\chi^2_\alpha = 12.5916$. Thus, the null hypothesis is rejected, and it can be concluded that there is significant difference between performances of the 6 algorithms. Further, the difference between the rank sums of the ML-MRRW and each other algorithm is calculated and presented in Table 5.6.

The following conclusions can be drawn from the plots and tables:

1. The average precisions of ML-LGC, ML-GFHF, ML-FSKSC, and M-LkNN, are quite close, whereas the ML-MR and ML-MRRW have significant large values on this metric under different labeling rates.

2. The performances of the six algorithms vary significantly on the metrics Micro F1 and Macro F1. For the smallest labeling rate, ML-FSKSC, ML-LGC, and ML-MRRW outperforms MLkNN; ML-MR and ML-GFHF perform worse than MLkNN. For the largest labeling rate, ML-MRRW, ML-FSKSC, and ML-MR outperforms MLkNN; ML-LGC and ML-GFHF perform worse than MLkNN.

3. Overall, ML-FSKSC and ML-MRRW achieve the best performances with Micro F1 and Macro F1. ML-MRRW performs better than ML-FSKSC with Micro F1 and Macro F1 for high labeling rates.

Moreover, ML-MRRW is also compared with an extensive supervised multi-label algorithms from the state-of-the-art literature [95] on the "Scene" data in Table 5.7. The last column presents the mean values of the average precision,

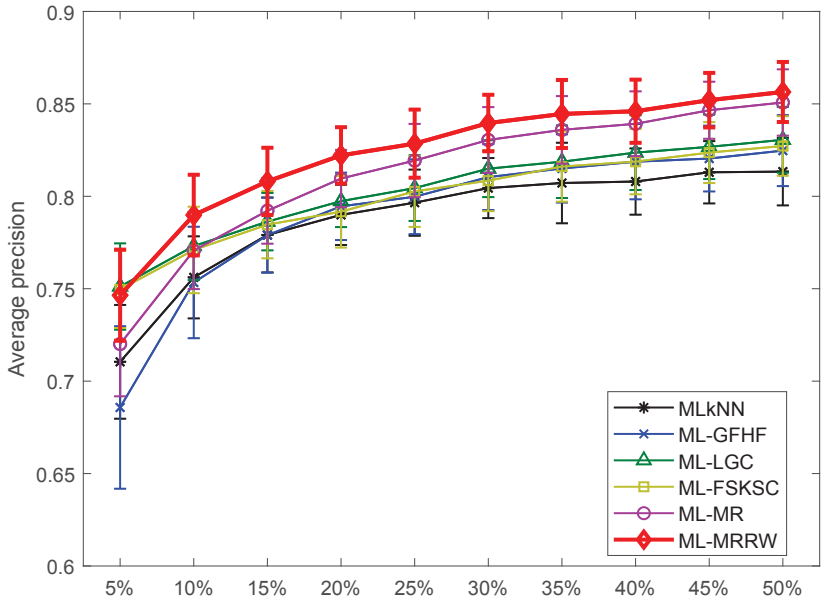

Figure 5.4: Average precision v.s. labeling rates for six classification algorithms applied to the "Scene" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

Micro F1, and Macro F1 for ML-MRRW under the labeling rate 50%. From the Table 5.7, ML-MRRW outperforms HOMER, ML-C4.5, PCT, and ML-KNN in terms of A-precision (A-precision stands for average precision). It performs better than ML-C4.5, PCT, ML-KNN, and RF-PCT in terms of Macro-F1 and it performs better than ML-C4.5, PCT, RFML-C4.5 and RF-PCT in terms of Micro-F1.

5.6.3 Case III: Yeast

The experimental results for the "Yeast" data are shown in Figs. 5.7, 5.8, and 5.9. Table 5.8 presents the Friedman's statistics F_R for the three different performance metrics. It can be found that all of them are greater than the critical chi-square value $\chi^2_\alpha = 12.5916$. Thus, the null hypothesis is rejected, and it can be concluded that there is significant difference between performances of the 6 algorithms. Further, the difference between the rank sums of the ML-MRRW and each other algorithm is calculated and presented in Table 5.9.

The following conclusions can be drawn from the plots and tables:

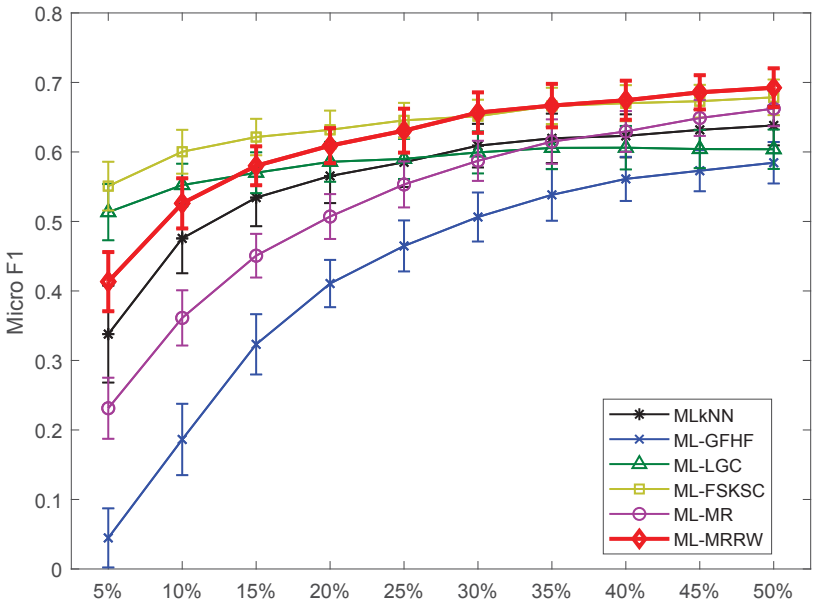

Figure 5.5: Micro F1 v.s. labeling rates for six classification algorithms applied to the "Scene" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

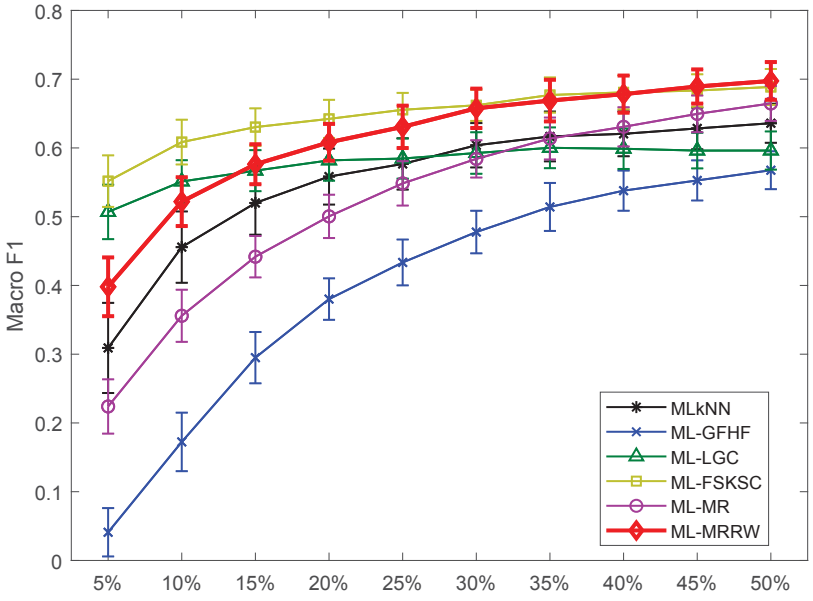

Figure 5.6: Macro F1 v.s. labeling rates for six classification algorithms applied to the "Scene" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

Table 5.7: Comparison with the state-of-the-art literature [95] on the "Scene" data.

	BR	CC	CLR	QWML	HOMER
A-precision	0.893	0.881	0.886	0.864	0.848
Micro-F1	0.761	0.757	0.758	0.756	0.764
Macro-F1	0.765	0.762	0.762	0.759	0.768
	ML-C4.5	PCT	ML-KNN	RAKEL	ECC
A-precision	0.751	0.745	0.851	0.862	0.856
Micro-F1	0.593	0.516	0.661	0.772	0.762
Macro-F1	0.596	0.593	0.692	0.777	0.770
	RFML-C4.5	RF-PCT	**ML-MRRW**		
A-precision	0.862	0.874	0.856		
Micro-F1	0.717	0.669	0.697		
Macro-F1	0.514	0.658	0.692		

Table 5.8: The Friedman's statistics F_R for different performance metrics in Case III.

	A-precision	Micro-F1	Macro-F1
F_R	46.7429	36.4000	43.6571

Table 5.9: The difference between the rank sums of the ML-MRRW and each other algorithm in Case III.

	A-precision	Micro-F1	Macro-F1
D_{61}	-35	-33	-28
D_{62}	-16	-33	-32
D_{63}	-33	-25	-5
D_{64}	-49	-45	14
D_{65}	-11	-14	-15

1. The ML-MRRW and ML-MR algorithms have the best performances in terms of the average precision for almost all the labeling rates.

2. ML-MRRW has the superior performance among the six algorithms in terms of Micro F1.

3. Although ML-FSKSC has the best performance in terms of Macro F1, it has poor performances with the other two metrics, namely, the average precision and Micro F1.

Moreover, ML-MRRW is also compared with an extensive supervised multi-

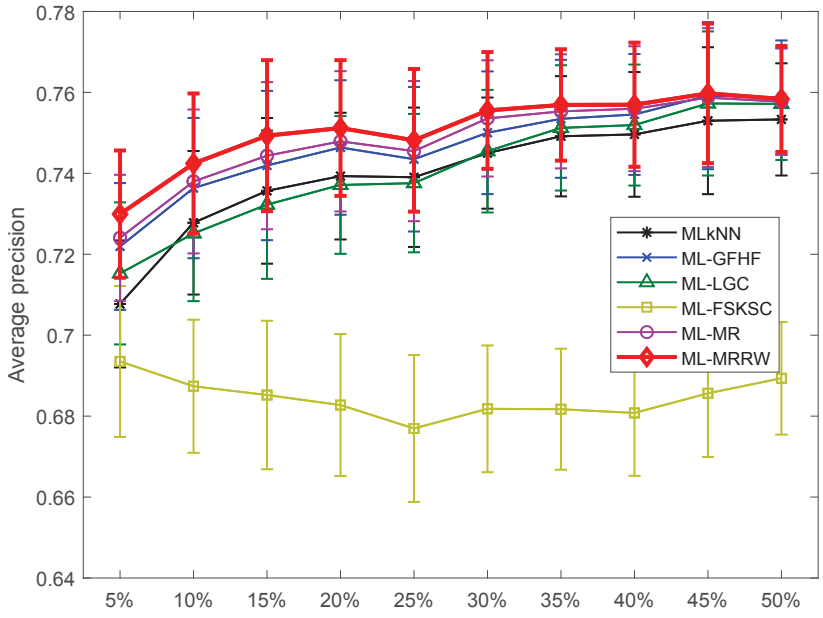

Figure 5.7: Average precision v.s. labeling rates for six classification algorithms applied to the "Yeast" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

label algorithms from the state-of-the-art literature on scene in Table 5.10. The last column presents the mean values of the average precision, Micro F1, and Macro F1 for ML-MRRW under the labeling rate 50%. From the Table 5.10, ML-MRRW outperforms all the algorithms in terms of A-precision (A-precision stands for average precision). It performs better than ML-C4.5, PCT, ML-KNN, RFML-C4.5 and RF-PCT in terms of Micro-F1 and it performs better than all the algorithms except for HOMER in terms of Micro-F1.

Case IV: Mediamill

The experimental results for "Mediamill" data are shown in Figs. 5.10, 5.11, and 5.12. Table 5.11 presents the Friedman's statistics F_R for the three different performance metrics. It can be found that all of them are greater than the critical chi-square value $\chi^2_\alpha = 12.5916$. Thus, the null hypothesis is rejected, and it can be concluded that there is significant difference between performances of the 6 algorithms. Further, the difference between the rank sums of the ML-MRRW and each other algorithm is calculated and presented in Table 5.12.

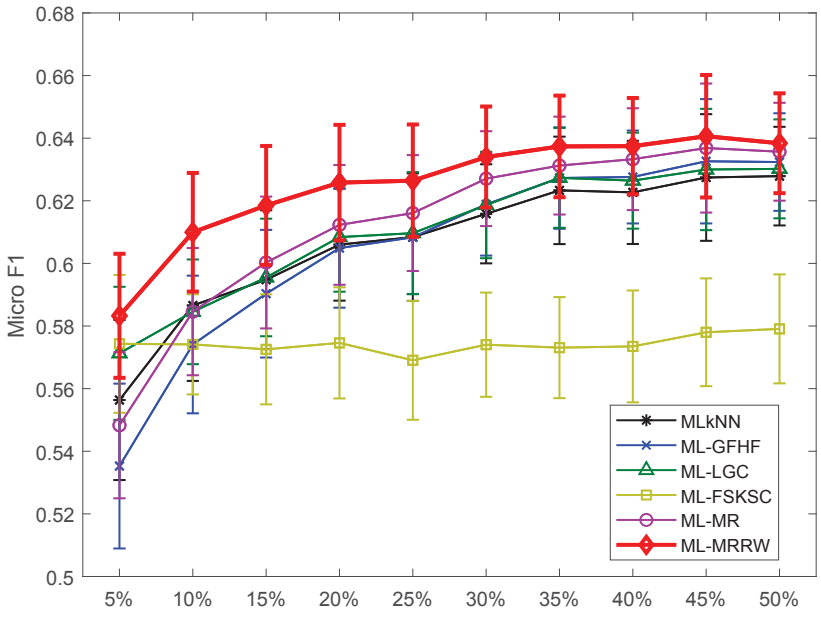

Figure 5.8: Micro F1 v.s. labeling rates for six classification algorithms applied to the "Yeast" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

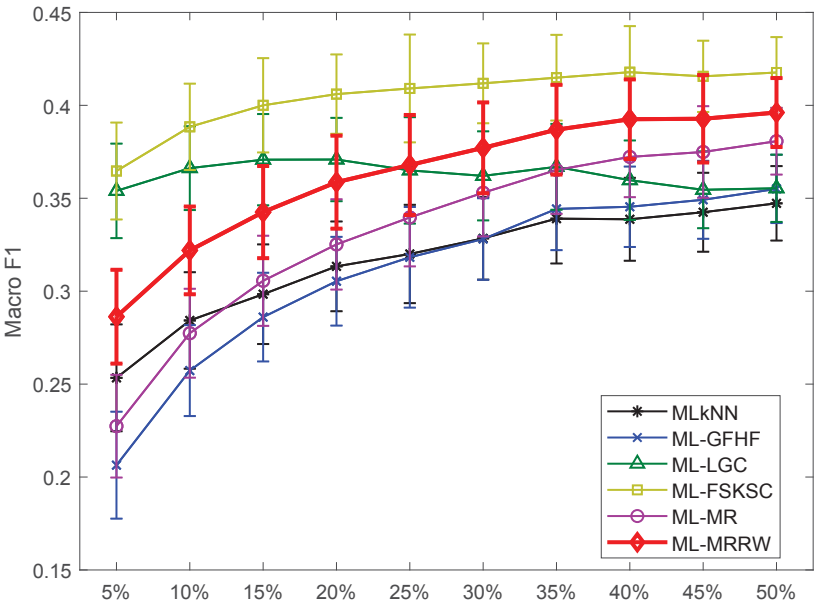

Figure 5.9: Macro F1 v.s. labeling rates for six classification algorithms applied to the "Yeast" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

Table 5.10: Comparison with the state-of-the-art literature [95] on the "Yeast" data.

	BR	CC	CLR	QWML	HOMER
A-precision	0.722	0.727	0.719	0.718	0.663
Micro-F1	0.652	0.650	0.655	0.654	0.673
Macro-F1	0.392	0.390	0.392	0.394	0.447
	ML-C4.5	PCT	ML-KNN	RAKEL	ECC
A-precision	0.620	0.705	0.732	0.715	0.667
Micro-F1	0.610	0.577	0.625	0.656	0.658
Macro-F1	0.370	0.293	0.336	0.359	0.350
	RFML-C4.5	RF-PCT	**ML-MRRW**		
A-precision	0.738	0.744	0.758		
Micro-F1	0.593	0.617	0.638		
Macro-F1	0.283	0.322	0.396		

Table 5.11: The Friedman's statistics F_R for different performance metrics in Case IV.

	A-precision	Micro-F1	Macro-F1
F_R	34.3429	46.0571	47.8857

Table 5.12: The difference between the rank sums of the ML-MRRW and each other algorithm in Case IV.

	A-precision	Micro-F1	Macro-F1
D_{61}	-24	-44	-39
D_{62}	7	-26	-29
D_{63}	-18	-25	-15
D_{64}	9	-45	-49
D_{65}	14	-10	-12

The following conclusions can be drawn from the plots and tables:

1. From the sub-figure of average precisions, it can be identified that the six algorithms achieve similar performances. The ML-MRRW and ML-MR outperform other algorithms for high labeling rates.

2. From the sub-figures of Micro F1 and Macro F1, it can be seen that the ML-MR and ML-MRRW methods outperform all the other methods quite a lot. Especially, the ML-MRRW method achieves the best performances on these two metrics.

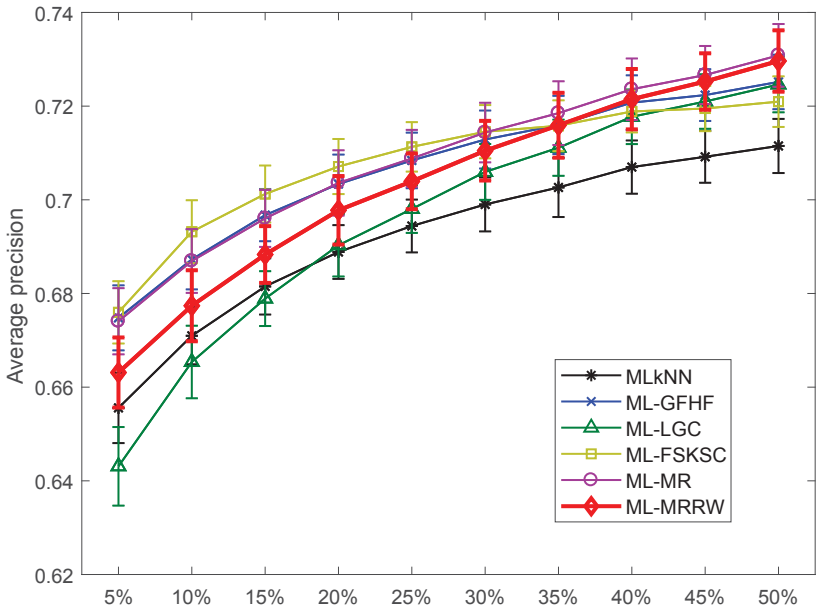

Figure 5.10: Average precision v.s. labeling rates for six classification algorithms applied to the "Mediamill" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

3. Overall, ML-MRRW defeats ML-MR and show superior performances over all the other algorithms on Micro F1 and Macro F1.

Moreover, ML-MRRW is also compared with an extensive supervised multi-label algorithms from the state-of-the-art literature on mediamill in Table 5.13. The last column presents the mean values of the average precision, Micro F1, and Macro F1 for ML-MRRW under the labeling rate 50%. Since this experiment only exploits a portion of the data set. In order to compare the metrics with those obtained from the whole "Mediamill" data, it is necessary to calculate the confidence interval. The 95% confidence intervals for A-precision, Macro-F1, and Micro-F1 using the ML-MRRW are obtained as [0.7268, 0.7325], [0.3355, 0.3539], and [0.6343, 0.6404], respectively. Comparing the metrics, it can be found that ML-MRRW performs better than all algorithms except for RF-PCT in terms of A-precision (A-precision stands for average precision). It outperforms all algorithms in terms of Macro-F1 and Micro-F1.

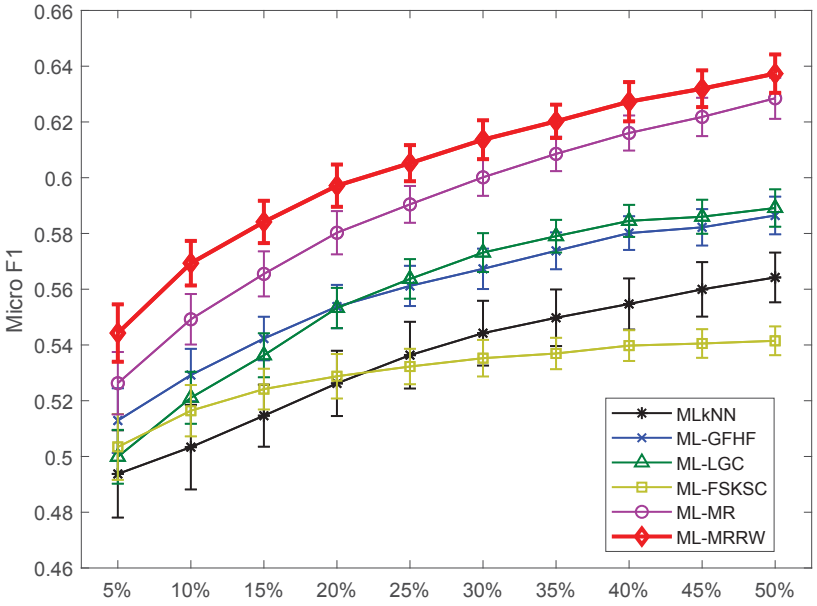

Figure 5.11: Micro F1 v.s. labeling rates for six classification algorithms applied to the "Mediamill" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

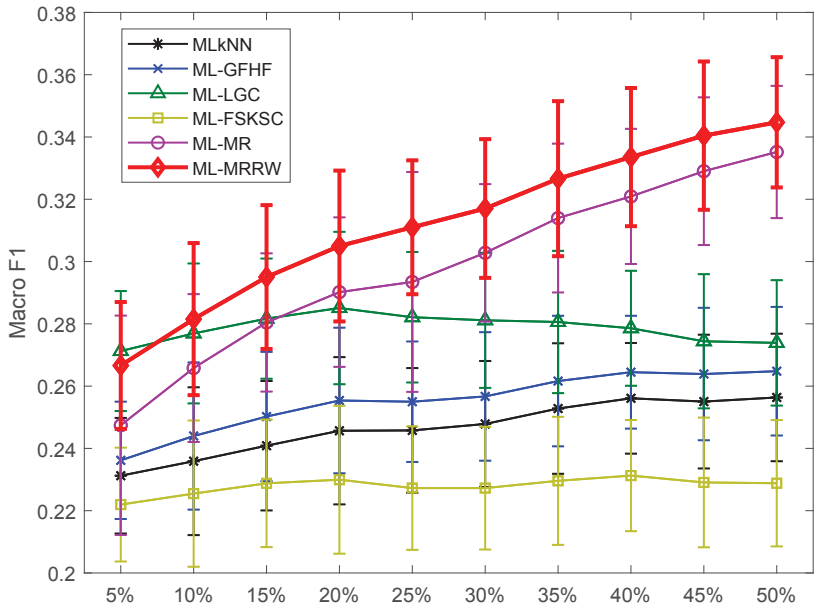

Figure 5.12: Macro F1 v.s. labeling rates for six classification algorithms applied to the "Mediamill" data. ML-KNN (black Asterisk), ML-GFHF (blue cross), ML-LGC (green triangle), ML-FSKSC (yellow square), ML-MR (magenta circle), and ML-MRRW (red diamond)

Table 5.13: Comparison with the state-of-the-art literature [95] on the "Mediamill" data.

	BR	CC	CLR	QWML	HOMER
A-precision	0.686	0.672	0.450	0.492	0.583
Macro-F1	0.056	0.052	0.037	0.037	0.073
Micro-F1	0.533	0.509	0.118	0.119	0.553
	ML-C4.5	PCT	ML-KNN	RAKEL	ECC
A-precision	0.669	0.654	0.703	0.492	0.453
Macro-F1	0.003	0.031	0.113	0.019	0.022
Micro-F1	0.007	0.477	0.545	0.440	0.453
	RFML-C4.5	RF-PCT	**ML-MRRW**		
A-precision	0.728	0.737	0.730		
Macro-F1	0.088	0.112	0.345		
Micro-F1	0.546	0.563	0.637		

5.7 Summary

This section studies the semi-supervised multi-label classification problem, and extends the graph-based manifold regularization to the multi-label case. The proposed method includes three essential components, including the graph construction, the manifold regularization with multiple labels, and the exploitation of a reliance weighting strategy. Especially, the last component is supposed to improve the learning ability by assigning higher weights to labeled training set and lower weights to unlabeled training sets. Extensive experiments are conducted on public data sets to test the performance of the proposed Multi-Label Manifold-Regularization (ML-MR) with/without the Reliance Weighting (RW) strategy. Other well-known semi-supervised and supervised multi-label algorithms are used for comparisons. Generally, the experimental results show that the proposed ML-MR and ML-MRRW algorithms have overall better performance that all the other algorithms under different labeling rates with statistical significance test. And ML-MRRW show better performance than ML-MR, indicating the proposed reliance weighting strategy is effective in improving the learning performance of the ML-MR method. Further, unlike the other algorithms, ML-MRRW works consistently well on all the datasets. Also ML-MRRW is compared with 12 supervised multi-label algorithms from the literature on the public data sets. Generally, ML-MRRW can compete against most of the supervised multi-label algorithms. All in all,

ML-MRRW is a promising semi-supervised multi-label algorithm.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis firstly gives a broad overview of NIALM in terms of data acquisition, feature extraction, reference learning and energy disaggregation. But mostly, the research work focuses on developing learning algorithms for NIALM. The achievements of the research work in this thesis are summarized as follows:

It explores the use of four bespoke multi-label supervised classification algorithms in NIALM. Four existing bespoke multi-label classification algorithms (ML-KNN, ML-RBF, ML-BPNN, ML-SVM) have been applied to disaggregate several public data sets. Comparison have been made to the existing literature on these datasets. And find that two of the algorithms are consistently superior to the other two, with the consistently "best" generally outperforming all other approaches that have been attempted on these data sets.

A semi-supervised multi-label disaggregation framework for NIALM is proposed, which we believe is a better match for the characteristics of the NIALM problem than the single-label approaches (supervised or unsupervised) used currently. Following this framework, we extend single-label graph based semi-supervised learning algorithms to the multilabel case. Where needed, we also extend the algorithms to allow out-of-sample prediction. And experiments on public NIALM datasets demonstrate that the best of our proposed method outperforms current results in the literature.

A graph-based semi-supervised manifold regularization algorithm is extended for multi-label problems. A weighting strategy is proposed to put more

trust in the labeled instances when forecasting labels for unforeseen points. The proposed approach is compared against four famous semi-supervised multi-label algorithms and a well-known supervised multi-label algorithm with extensive experimental study.

6.2 Future Work

Any study needs continuous efforts and research work. In the field of NIALM, the following promising directions deserve efforts for future work:

A worthwhile direction is to incorporate more electric and non-electric features like reactive power, temporal information and seasonality effects into disaggregation to improve its performance. For instance, reactive power can disaggregate well between purely resistive, inductive and capacitive appliances. Time usage duration could provide useful information, some appliances are running all the time like fridge and some devices are used for a few minutes like microwave. And seasonality changes appliance usage patterns. Incorporating these as additional signatures into semi-supervised learning might be able to improve the disaggregation accuracy and it would be worthwhile future work.

Another promising direction would be focusing on energy disaggregation in addition to operational states detection. Particulary for multi-state and continuously varying appliances. Currently, it's possible to identify the operational states of multi-state and continuously varying appliances, but accurately estimating their energy consumption is still a big challenge since they have different energy consumption level in different states.

In the field of further developing SSML algorithms, two immediate directions that suggest themselves are generalizing the concept of similarity in the adjacency matrix, and exploring alternatives to LapRLS.

Bibliography

[1] U.S. Energy Information Administration. Electricity explained: Use of electricity. http:// www.eia.gov/ energyexplained/ index.cfm?page= electricity_use [online], April 26 2012. Retrieved.

[2] Kofi Afrifa Agyeman, Sekyung Han, and Soohee Han. Real-time recognition non-intrusive electrical appliance monitoring algorithm for a residential building energy management system. *Energies*, 8(9):9029–9048, 2015.

[3] Hana Altrabalsi, Vladimir Stankovic, Jing Liao, and Lina Stankovic. Low-complexity energy disaggregation using appliance load modelling. *AIMS Energy*, 4(1):884–905, 2016.

[4] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges. Blued: A fully labeled public dataset for event-based non-intrusive load monitoring research. In *Proceedings of the 2nd KDD workshop on data mining applications in sustainability (SustKDD)*, pages 1–5, 2012.

[5] Kyle D Anderson, Mario E Bergés, Adrian Ocneanu, Diego Benitez, and José MF Moura. Event detection for non intrusive load monitoring. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3312–3317. IEEE, 2012.

[6] Rana Aamir Raza Ashfaq, Xi-Zhao Wang, Joshua Zhexue Huang, Haider Abbas, and Yu-Lin He. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, 378:484–497, 2017.

[7] Sean Barker, Aditya Mishra, David Irwin, Emmanuel Cecchet, Prashant Shenoy, and Jeannie Albrecht. Smart*: An open data set and tools for

enabling research in sustainable homes. In *SustKDD, August*, page 112, 2012.

[8] Karim Said Barsim, Roman Streubel, and Bin Yang. Unsupervised adaptive event detection for building-level energy disaggregation. In *Proceedings of power and energy student summt (PESS)*, Stuttgart, Germany, 2014.

[9] Kaustav Basu, Vincent Debusschere, and Seddik Bacha. Appliance usage prediction using a time series based classification approach. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 1217–1222, Montreal, QC, Canada, 2012. IEEE.

[10] Kaustav Basu, Vincent Debusschere, and Seddik Bacha. Load identification from power recordings at meter panel in residential households. In *Electrical Machines (ICEM), 2012 XXth International Conference on*, pages 2098–2104. IEEE, 2012.

[11] Kaustav Basu, Vincent Debusschere, and Seddik Bacha. Residential appliance identification and future usage prediction from smart meter. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, pages 4994–4999, Vienna, Austria, 2013. IEEE.

[12] Kaustav Basu, Vincent Debusschere, Seddik Bacha, Ujjwal Maulik, and Sanghamitra Bondyopadhyay. Nonintrusive load monitoring: A temporal multilabel classification approach. *IEEE Transactions on industrial informatics*, 11(1):262–270, 2015.

[13] Kaustav Basu, Vincent Debusschere, Ahlame Douzal-Chouakria, and Seddik Bacha. Time series distance-based methods for non-intrusive load monitoring in residential buildings. *Energy and Buildings*, 96:109–117, 2015.

[14] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. Nilmtk: an open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*, pages 265–276. ACM, 2014.

[15] Nipun Batra, Amarjeet Singh, and Kamin Whitehouse. Neighbourhood nilm: A big-data approach to household energy disaggregation. *arXiv preprint arXiv:1511.02900*, 2015.

[16] Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 80–89. ACM, 2014.

[17] Mikhail Belkin and Partha Niyogi. Using manifold stucture for partially labeled classification. In *Advances in neural information processing systems*, pages 953–960, 2003.

[18] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

[19] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[20] Mario E Berges, Ethan Goldman, H Scott Matthews, and Lucio Soibelman. Enhancing electricity audits in residential buildings with nonintrusive load monitoring. *Journal of Industrial Ecology*, 14(5):844–858, 2010.

[21] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, page 1926, 2001.

[22] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[23] Roberto Bonfigli, Stefano Squartini, Marco Fagiani, and Francesco Piazza. Unsupervised algorithms for non-intrusive load monitoring: An up-to-date overview. In *Environment and Electrical Engineering (EEEIC), 2015 IEEE 15th International Conference on*, pages 1175–1180, Chicago, USA, 2015. IEEE.

[24] Vinicius P Borin, Carlos H Barriquello, Mário L da Silva Martins, and Alexandre Campos. A novel approach for home appliances identification using vector projection length and stockwell transform. In *Power Electronics Conference and 1st Southern Power Electronics Conference (COBEP/SPEC), 2015 IEEE 13th Brazilian*, pages 1–6, Fortaleza, Brazil, 2015. IEEE.

[25] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[26] Claudio Carmeli, Ernesto De Vito, and Alessandro Toigo. Vector valued reproducing kernel hilbert spaces of integrable functions and mercer theorem. *Analysis and Applications*, 4(04):377–408, 2006.

[27] Hakan Cevikalp and Vojtech Franc. Large-scale robust transductive support vector machines. *Neurocomputing*, 235:199–209, 2017.

[28] Hsueh-Hsien Chang, Lung-Shu Lin, Nanming Chen, and Wei-Jen Lee. Particle-swarm-optimization-based nonintrusive demand monitoring and load identification in smart meters. *IEEE Transactions on Industry Applications*, 49(5):2229–2236, 2013.

[29] Hsueh-Hsien Chang, Putu Wegadiputra Wiratha, and Nanming Chen. A non-intrusive load monitoring system using an embedded system for applications to unbalanced residential distribution systems. *Energy Procedia*, 61:146–150, 2014.

[30] Hsueh-Hsien Chang, Hong-Tzer Yang, and Ching-Lung Lin. Load identification in neural networks for a non-intrusive monitoring of industrial electrical loads. In *International Conference on Computer Supported Cooperative Work in Design*, pages 664–674. Springer, 2007.

[31] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-supervised learning*. The MIT Press: Cambridge, Massachusetts / London, England, 2006.

[32] Olivier Chapelle, Vikas Sindhwani, and Sathiya S Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9(Feb):203–233, 2008.

[33] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *AISTATS*, pages 57–64. Citeseer, 2005.

[34] Agnim I Cole and Alexander Albicki. Data extraction for effective non-intrusive identification of residential power loads. In *Instrumentation and Measurement Technology Conference, 1998. IMTC/98. Conference Proceedings. IEEE*, volume 2, pages 812–815, St. Paul, MN, USA, 1998. IEEE.

[35] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7(Aug):1687–1712, 2006.

[36] Sarah Darby. The effectiveness of feedback on energy consumption: a review for defra of the literature on metering, billing and direct displays. Technical report, Environmental Change Institiute, University of Oxford, 2006.

[37] Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 96–103, 2005.

[38] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[39] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[40] HGCP Dinesh, PH Perera, GMRI Godaliyadda, MPB Ekanayake, and JB Ekanayake. Individual power profile estimation of residential appliances using low frequency smart meter data. In *Industrial and Information Systems (ICIIS), 2015 IEEE 10th International Conference on*, pages 140–145, Peradeniya, Sri Lanka, 2015. IEEE.

[41] Shifei Ding, Zhibin Zhu, and Xiekai Zhang. An overview on semi-supervised support vector machine. *Neural Computing and Applications*, 28(5):969–978, 2017.

[42] Peter Gustav Lejeune Dirichlet and Beweis des Satzes. dass jede unbegrentze arithmetische progression, deren erstes glied und differenz ganze zahlen ohne gemeinschaftlichen factor sind, unendlich viele primzahlen enth alt. *Abh. der Königlichen Preuss. Akad. der Wiss*, pages 45–81, 1837.

[43] M. Dong, H. E. Mazin, and Wilsun Xu. Data segmentation algorithms for a time-domain harmonic source modeling method. In *Electrical Power & Energy Conference (EPEC), 2009 IEEE*, pages 1–6. IEEE, 2009.

[44] Ming Dong, Paulo CM Meira, Wilsun Xu, and CY Chung. Non-intrusive signature extraction for major residential loads. *IEEE Transactions on Smart Grid*, 4(3):1421–1430, 2013.

[45] Ming Dong, Paulo CM Meira, Wilsun Xu, and Walmir Freitas. An event window based load monitoring technique for smart meters. *IEEE transactions on smart grid*, 3(2):787–796, 2012.

[46] Dominik Egarter and Wilfried Elmenreich. Autonomous load disaggregation approach based on active power measurements. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 293–298, St. Louis, MO, USA, 2015. IEEE.

[47] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2002.

[48] Zekeriya Erkin, Juan Ramón Troncoso-Pastoriza, Reginald L Lagendijk, and Fernando Pérez-González. Privacy-preserving data aggregation in smart metering systems: An overview. *IEEE Signal Processing Magazine*, 30(2):75–86, 2013.

[49] Mingyu Fan, Nannan Gu, Hong Qiao, and Bo Zhang. Dimensionality reduction: An interpretation from manifold regularization perspective. *Information Sciences*, 277:694–714, 2014.

[50] Anthony Faustine, Nerey Henry Mvungi, Shubi Kaijage, and Kisangiri Michael. A survey on non-intrusive load monitoring methodies

and techniques for energy disaggregation problem. *arXiv preprint arXiv:1703.00785*, 2017.

[51] Francisco-Javier Ferrández-Pastor, Juan-Manuel García-Chamizo, Mario Nieto-Hidalgo, Vicente Romacho-Agud, and Francisco Flórez-Revuelta. Using wavelet transform to disaggregate electrical power consumption into the major end-uses. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 272–279. Springer, 2014.

[52] Marisa Figueiredo, Ana De Almeida, and Bernardete Ribeiro. Home electrical signal disaggregation for non-intrusive load monitoring (nilm) systems. *Neurocomputing*, 96:66–73, 2012.

[53] Marisa B Figueiredo, Ana De Almeida, and Bernardete Ribeiro. An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 31–40. Springer, 2011.

[54] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[55] Pengfei Gao, Shunfu Lin, and Wilsun Xu. A novel current sensor for home energy use monitoring. *IEEE Transactions on Smart Grid*, 5(4):2021–2028, 2014.

[56] Salvador Garcia and Francisco Herrera. An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677–2694, 2008.

[57] Clark W Gellings. The concept of demand-side management for electric utilities. *Proceedings of the IEEE*, 73(10):1468–1470, 1985.

[58] Andrew Granville. On elementary proofs of the prime number theorem for arithmetic progressions, without characters. In *Proceedings of the Amalfi Conference on Analytic Number Theory*, pages 157–195, 1989.

[59] Zhenyu Guo, Z Jane Wang, and Ali Kashani. Home appliance load modeling from aggregated smart meter data. *IEEE Transactions on power systems*, 30(1):254–262, 2015.

[60] George W Hart. *Nonintrusive Appliance Load Data Acquisition Method: Progress Report*. MIT Energy Laboratory, 1984.

[61] George W Hart. Prototype nonintrusive appliance load monitor, 1985.

[62] George William Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[63] Taha Hassan. Bi-level characterization of manual setup residential non-intrusive demand disaggregation using enhanced differential evolution. In *Proc. 1st Int. Workshop Non-Intrusive Load Monitoring*, 2012.

[64] Taha Hassan, Fahad Javed, and Naveed Arshad. An empirical investigation of vi trajectory based load signatures for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 5(2):870–878, 2014.

[65] Simon Haykin. *Unsupervised Adaptive Filtering Volume I: Blind Source Separation*. John Wiley & Sons: New York, 2000.

[66] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, Inc.: Upper Saddle River, NJ, 2009.

[67] K. He, L. Stankovic, J. Liao, and V. Stankovic. Non-intrusive load disaggregation using graph signal processing. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2017.

[68] Weiming Hu, Jin Gao, Junliang Xing, Chao Zhang, and Stephen Maybank. Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):172–188, 2017.

[69] Bert C Huang and Tony Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *International Conference on Artificial Intelligence and Statistics*, pages 195–202, 2007.

[70] Farrokh Jazizadeh, Burcin Becerik-Gerber, Mario Berges, and Lucio Soibelman. An unsupervised hierarchical clustering based heuristic algorithm for facilitated training of electricity consumption disaggregation systems. *Advanced Engineering Informatics*, 28(4):311–326, 2014.

[71] Tony Jebara, Jun Wang, and Shih-Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448. ACM, 2009.

[72] Lei Jiang, Su Huai Luo, and Jia Ming Li. Intelligent electrical appliance event recognition using multi-load decomposition. In *Advanced Materials Research*, volume 805, pages 1039–1045. Trans Tech Publ, 2013.

[73] Lei Jiang, Suhuai Luo, and Jiaming Li. Automatic power load event detection and appliance classification based on power harmonic features in nonintrusive appliance load monitoring. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pages 1083–1088, Melbourne, VIC, Australia, 2013. IEEE.

[74] Yulieth Jimenez, Cesar Duarte, Johann Petit, and Gilberto Carrillo. Feature extraction for nonintrusive load monitoring based on s-transform. In *Power Systems Conference (PSC), 2014 Clemson University*, pages 1–5, Clemson, SC, USA, 2014. IEEE.

[75] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 290–297, 2003.

[76] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, Boston, MA, USA, 2003.

[77] Jack Kelly and William Knottenbelt. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 55–64, Seoul, South Korea, 2015. ACM.

[78] Hyungsul Kim, Manish Marwah, Martin Arlitt, Geoff Lyon, and Jiawei Han. Unsupervised disaggregation of low frequency power mea-

surements. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 747–758. SIAM, 2011.

[79] J Zico Kolter and Tommi Jaakkola. Approximate inference in additive factorial hmms with application to energy disaggregation. In *International conference on artificial intelligence and statistics*, pages 1472–1482, 2012.

[80] J Zico Kolter and Matthew J Johnson. Redd: A public data set for energy disaggregation research. In *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62, 2011.

[81] Frederik Laasch, Alain Dieterlen, and Dirk Benyoucef. Event detection using adaptive thresholds for non-intrusive load monitoring. *the Baden-Württemberg Symp. Inf. Comm. Syst.*, 50:63, 2015.

[82] Ying-Xun Lai, Chin-Feng Lai, Yueh-Min Huang, and Han-Chieh Chao. Multi-appliance recognition system with hybrid svm/gmm classifier in ubiquitous smart home. *Information Sciences*, 230:39–55, 2013.

[83] Hong Yin Lam, GSK Fung, and WK Lee. A novel method to construct taxonomy electrical appliances based on load signaturesof. *IEEE Transactions on Consumer Electronics*, 53(2):653–660, 2007.

[84] Xuan-Chien Le, Baptiste Vrigneau, and Olivier Sentieys. l1-norm minimization based algorithm for non-intrusive load monitoring. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 299–304, St. Louis, MO, USA, 2015. IEEE.

[85] WK Lee, GSK Fung, HY Lam, FHY Chan, and Mark Lucente. Exploration on load signatures. In *International conference on electrical Engineering (ICEE)*, volume 152, pages 1–5, Sapporo, Japan, 2004.

[86] Ding Li and Scott Dick. Whole-house non-intrusive appliance load monitoring via multi-label classification. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 2749–2755, Vancouver, Canada, 2016. IEEE.

[87] Ding Li and Scott Dick. Disaggregating household appliance loads using multi-label classification methods. *International Transactions on Electrical Energy Systems*, 2017. submitted.

[88] Ding Li and Scott Dick. A graph-based semi-supervised learning approach towards household energy disaggregation. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*, pages 1–7, Naples, Italy, 2017. IEEE.

[89] Ding Li and Scott Dick. Residential household non-intrusive load monitoring via graph-based multi-label semi-supervised learning. *IEEE Transactions on Smart Grid*, 2018. accepted.

[90] Ding Li, Kyle Sawyer, and Scott Dick. Disaggregating household loads via semi-supervised multi-label classification. In *Fuzzy Information Processing Society (NAFIPS)*, pages 1–5, Redmond, WA, 2015. IEEE.

[91] Jian Liang, Simon KK Ng, Gail Kendall, and John WM Cheng. Load signature study part i: Basic concept, structure, and methodology. *IEEE transactions on power Delivery*, 25(2):551–560, 2010.

[92] Yu-Hsiu Lin and Men-Shen Tsai. Development of an improved time–frequency analysis-based nonintrusive load monitor for load demand identification. *IEEE Transactions on Instrumentation and Measurement*, 63(6):1470–1483, 2014.

[93] Thillainathan Logenthiran, Dipti Srinivasan, and Tan Zong Shun. Demand side management in smart grid using heuristic optimization. *IEEE transactions on smart grid*, 3(3):1244–1252, 2012.

[94] Mehdi Maasoumy, B Sanandaji, Kameshwar Poolla, and Alberto Sangiovanni Vincentelli. Berds-berkeley energy disaggregation data set. In *Proceedings of the Workshop on Big Learning at the Conference on Neural Information Processing Systems (NIPS)*, 2013.

[95] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern recognition*, 45(9):3084–3104, 2012.

[96] Julien Maitre, Guillaume Glon, Sebastien Gaboury, Bruno Bouchard, and Abdenour Bouzouane. Efficient appliances recognition in smart homes based on active and reactive power, fast fourier transform and decision trees. In *AAAI Workshop: Artificial Intelligence Applied to Assistive Technologies and Smart Environments*, Austin, TX, USA, 2015.

[97] Stephen Makonin, Fred Popowich, Ivan V Bajić, Bob Gill, and Lyn Bartram. Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring. *IEEE Transactions on Smart Grid*, 7(6):2575–2585, 2016.

[98] Stephen Makonin, Fred Popowich, Lyn Bartram, Bob Gill, and Ivan V Bajic. Ampds: a public dataset for load disaggregation and eco-feedback research. In *Electrical Power & Energy Conference (EPEC), 2013 IEEE*, pages 1–6. IEEE, 2013.

[99] Pavan Kumar Mallapragada, Rong Jin, Anil K Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):2000–2014, 2009.

[100] Lukas Mauch and Bin Yang. A new approach for supervised power disaggregation by using a deep recurrent lstm network. In *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*, pages 63–67, Orlando, FL, USA, 2015. IEEE.

[101] Paula Meehan, Conor McArdle, and Stephen Daniels. An efficient, scalable time-frequency method for tracking energy usage of domestic appliances using a two-step classification algorithm. *Energies*, 7(11):7041–7066, 2014.

[102] Siamak Mehrkanoon and Johan AK Suykens. Multi-label semi-supervised learning using regularized kernel spectral clustering. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 4009–4016. IEEE, 2016.

[103] ROMEO MEŠTROVIC. Euclids theorem on the infinitude of primes: A historical survey of its proofs (300 bc–2012) and another new proof. *arXiv preprint arXiv:1202.3670*, 2012.

[104] Ha Quang Minh, Loris Bazzani, and Vittorio Murino. A unifying framework in vector-valued reproducing kernel hilbert spaces for manifold regularization and co-regularized multi-view learning. *Journal of Machine Learning Research*, 17(25):1–72, 2016.

[105] Ha Quang Minh and Vikas Sindhwani. Vector-valued manifold regularization. In *International Conference on Machine Learning*, pages 57–64, 2011.

[106] Hala Najmeddine, Khalil El Khamlichi Drissi, Christophe Pasquier, Claire Faure, Kamal Kerroum, Alioune Diop, Thierry Jouannet, and Michel Michou. State of art on load monitoring methods. In *Power and Energy Conference, 2008. PECon 2008. IEEE 2nd International*, pages 1256–1258. IEEE, 2008.

[107] Wladyslaw Narkiewicz. *The development of prime number theory: from Euclid to Hardy and Littlewood*. Springer Science & Business Media, 2013.

[108] Gulisong Nasierding and Abbas Z Kouzani. Comparative evaluation of multi-label classification methods. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 679–683. IEEE, 2012.

[109] Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. An unsupervised training method for non-intrusive appliance load monitoring. *Artificial Intelligence*, 217:1–19, 2014.

[110] Oliver Parson, Siddhartha Ghosh, Mark J Weal, and Alex Rogers. Non-intrusive load monitoring using prior models of general appliance types. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 22–26, 2012.

[111] Om P Patri, Anand V Panangadan, Charalampos Chelmis, and Viktor K Prasanna. Extracting discriminative features for event-based electricity disaggregation. In *Technologies for Sustainability (SusTech), 2014 IEEE Conference on*, pages 232–238. IEEE, 2014.

[112] Thomas Picon, Mohamed Nait Meziane, Philippe Ravier, Guy Lamarque, Clarisse Novello, Jean-Charles Le Bunetel, and Yves Raingeaud.

Cooll: Controlled on/off loads library, a public dataset of high-sampled electrical signals for appliance identification. *arXiv preprint arXiv:1611.05803*, 2016.

[113] Manfred Pöchacker, Dominik Egarter, and Wilfried Elmenreich. Proficiency of power values for load disaggregation. *IEEE Transactions on Instrumentation and Measurement*, 65(1):46–55, 2016.

[114] Dwi Rahayu, Balakrishnan Narayanaswamy, Shonali Krishnaswamy, Cyril Labbé, and Deva P Seetharam. Learning to be energy-wise: Discriminative methods for load disaggregation. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, pages 1–4. ACM, 2012.

[115] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. On the accuracy of appliance identification based on distributed load metering data. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*, pages 1–9. IEEE, 2012.

[116] Antonio G Ruzzelli, C Nicolas, Anthony Schoofs, and Gregory MP O'Hare. Real-time recognition and profiling of appliances through a single electricity sensor. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2010.

[117] Pedram Samadi, Hamed Mohsenian-Rad, Robert Schober, and Vincent WS Wong. Advanced demand side management for the future smart grid using mechanism design. *IEEE Transactions on Smart Grid*, 3(3):1170–1180, 2012.

[118] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[119] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[120] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh, 2000.

[121] Atle Selberg. An elementary proof of dirichlet's theorem about primes in an arithmetic progression. *Annals of Mathematics*, pages 297–304, 1949.

[122] Michael R Sheldon, Michael J Fillyaw, and W Douglas Thompson. The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International*, 1(4):221–228, 1996.

[123] Vikas Sindhwani, S Sathiya Keerthi, and Olivier Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the 23rd international conference on Machine learning*, pages 841–848. ACM, 2006.

[124] Munendra Singh, Sanjeev Kumar, Sunil Semwal, and RS Prasad. Residential load signature analysis for their segregation using waveletsvm. In *Power Electronics and Renewable Energy Systems*, pages 863–871. Springer, 2015.

[125] Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 421–430. ACM, 2006.

[126] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. Technical report, Oregon State University, Corvallis, 2010.

[127] Kathleen Spees and Lester Lave. Impacts of responsive load in pjm: load shifting and real time pricing. *The Energy Journal*, pages 101–121, 2008.

[128] D Srinivasan, WS Ng, and AC Liew. Neural-network-based signature recognition for harmonic source identification. *IEEE Transactions on Power Delivery*, 21(1):398–405, 2006.

[129] T. D. Stickels and L. C. Markel. Residential load control and metering equipment: costs and capabilities. Technical report, Electrotek Concepts, Inc., Mountain View, CA (USA); Electric Power Research Inst., Palo Alto, CA (USA), 1987.

[130] Amarnag Subramanya and Jeff Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12(Nov):3311–3370, 2011.

[131] Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

[132] Shiliang Sun and Xijiong Xie. Semisupervised support vector machines with tangent space intrinsic manifold regularization. *IEEE transactions on neural networks and learning systems*, 27(9):1827–1839, 2016.

[133] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in neural information processing systems*, pages 945–952, 2002.

[134] Seyed Mostafa Tabatabaei, Scott Dick, and Wilsun Xu. Toward non-intrusive load monitoring via multi-label classification. *IEEE Transactions on Smart Grid*, 8(1):26–40, 2017.

[135] Floris Takens. Detecting strange attractors in turbulence. *Lecture notes in mathematics*, 1981.

[136] Guoming Tang, Kui Wu, and Jingsheng Lei. A distributed and scalable approach to semi-intrusive load monitoring. *IEEE Transactions on Parallel and Distributed Systems*, 27(6):1553–1565, 2016.

[137] Guoming Tang, Kui Wu, Jingsheng Lei, and Jiuyang Tang. A simple model-driven approach to energy disaggregation. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 566–571, Venice, Italy, 2014. IEEE.

[138] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

[139] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

[140] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[141] Enmei Tu, Yaqian Zhang, Lin Zhu, Jie Yang, and Nikola Kasabov. A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification. *Information Sciences*, 367:673–688, 2016.

[142] Akshay SN Uttama Nambi, Antonio Reyes Lua, and Venkatesha R Prasad. Loced: Location-aware energy disaggregation framework. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 45–54. ACM, 2015.

[143] Vladimir Vapnik. *Statistical learning theory. 1998*, volume 3. Wiley, New York, 1998.

[144] Jun Wang, Tony Jebara, and Shih-Fu Chang. Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, 14(Mar):771–800, 2013.

[145] Zhenyu Wang and Guilin Zheng. Residential appliances identification and monitoring by a nonintrusive method. *IEEE transactions on Smart Grid*, 3(1):80–92, 2012.

[146] Warit Wichakool, Al-Thaddeus Avestruz, Robert W Cox, and Steven B Leeb. Modeling and estimating current harmonics of variable electronic loads. *IEEE Transactions on Power Electronics*, 24(12):2803–2811, 2009.

[147] Voon Siong Wong, Yung Fei Wong, Tom Drummond, and Y Ahmet Şekercioğlu. A fast multiple appliance detection algorithm for non-intrusive load monitoring. In *Computational Intelligence Applications In Smart Grid (CIASG), 2013 IEEE Symposium on*, pages 80–86, Singapore, 2013. IEEE.

[148] Matt Wytock and J Zico Kolter. Contextually supervised source separation with application to energy disaggregation. In *the AAAI C. Art. Int.*, pages 486–492, Quebec City, QC, Canada, 2014.

[149] Michael Zeifman. Disaggregation of home energy display data using probabilistic approach. *IEEE Transactions on Consumer Electronics*, 58(1):23–31, 2012.

[150] Michael Zeifman and Kurt Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE transactions on Consumer Electronics*, 57(1):76–84, 2011.

[151] Zheng-Jun Zha, Tao Mei, Jingdong Wang, Zengfu Wang, and Xian-Sheng Hua. Graph-based semi-supervised learning with multiple labels. *Journal of Visual Communication and Image Representation*, 20(2):97–103, 2009.

[152] Min-Ling Zhang. Ml-rbf: Rbf neural networks for multi-label learning. *Neural Processing Letters*, 29(2):61–74, 2009.

[153] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[154] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: a lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[155] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

[156] Zhao Zhang, Mingbo Zhao, and Tommy WS Chow. Graph based constrained semi-supervised learning framework via label propagation over adaptive neighborhood. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2362–2376, 2015.

[157] Bochao Zhao, Lina Stankovic, and Vladimir Stankovic. On a training-less solution for non-intrusive appliance load monitoring using graph signal processing. *IEEE Access*, 4:1784–1799, 2016.

[158] Ye Zhao, Roy Ball, Jerry Mosesian, Jean-François de Palma, and Brad Lehman. Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays. *IEEE Transactions on Power Electronics*, 30(5):2848–2858, 2015.

[159] Yufeng Zhao, Yao Zhao, and Zhenfeng Zhu. Tsvm-hmm: Transductive svm based hidden markov model for automatic image annotation. *Expert Systems with Applications*, 36(6):9813–9818, 2009.

[160] Mingjun Zhong, Nigel Goddard, and Charles Sutton. Latent bayesian melding for integrating individual and population models. In *Advances in neural information processing systems*, pages 3618–3626, 2015.

[161] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[162] X. Zhou and M. Belkin. Semi-supervised learning. In *Chapter 22, Academic Press Library in Signal Processing*, 2013.

[163] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.

[164] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[165] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. In *Technical Report CMU-CALD-02-107*. Citeseer, 2002.

[166] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[167] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[168] Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.