BIG DATA ANALYSIS OF COMPLEX NETWORKS USING MACHINE LEARNING METHODS

A Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Electrical Engineering

by

Erte Pan

May 2016

BIG DATA ANALYSIS OF COMPLEX NETWORKS USING MACHINE LEARNING METHODS

Erte Pan

Approved:

Chair of the Committee
Zhu Han, Professor,
Electrical and Computer Engineering

Committee Members:

Haluk Ogmen, Professor,
Electrical and Computer Engineering

Saurabh Prasad, Assistant Professor,
Electrical and Computer Engineering

Miao Pan, Assistant Professor,
Electrical and Computer Engineering

Husheng Li, Associate Professor,
Electrical Engineering &
Computer Science
University of Tennessee

Lijun Qian, Professor,
Electrical and Computer Engineering
Prairie View A&M University

Suresh K. Khator, Associate Dean
Cullen College of Engineering

Badri Roysam, Professor and Chair
Electrical and Computer Engineering

Acknowledgements

I would like to thank my family who supported me whenever and whatever happened. I am so appreciative for everyone that came across my life during my Ph.D. study, bringing me lots of memorial moments to cherish in the future.

First of all, I would like to express my appreciation to my advisor, Professor Zhu Han, for his esteemed guidance, valuable advice, constant encouragement, and continuous support during my graduate studies. His profound academic background and keen insights has helped me achieve significantly improvement in my Ph.D. research and well prepared for future professional development, which will be an invaluable assets for my future career.

Furthermore, I would like to express my sincere gratitude to Professor Kirill Larin, Professor Badri Roysam, Professor Ben Jansen, Professor Saurabh Prasad, Professor David Jackson, Professor Haluk Ogmen, Professor Miao Pan, and Professor Robert Azencott. From whom I learn not only the profound knowledge and skills but also the professional ways to be involved in big projects. I would also like to thank the rest of my dissertation committee, Professor Husheng Li, and Professor Lijun Qian for their precious time and support on this dissertation.

My appreciation also goes to my dear colleagues Xunsheng Du, Huaqing Zhang, Yunan Gu, Yanru Zhang, Hung Ngyuen, Yong Xiao, Ali Arab, Nam Ngyuen, Mohammad Esmalifalak, Lanchao Liu, Najmeh Forouzandehmehr, Yi Huang, Jingyi Wang, and Mounika Sai, whom I always had a great time with in both on and off work times. I am also grateful to my academic introducers and friends: YanKai Tu, Ravi Kiran Manapuram, Narendran Sudheendran, Jiasong Li, Shang Wang, Maleeha Mashiatulla, Raghav K. Padmanabhan, Kedar Grama, Amin Merouane, Prashamesh Kulkarni, Nicolas Rey Villamizar, Murad Megjhani, Yanbin Lv, Yan Xu, Karthik Uppuluri, and Prithvi B. S.. I am grateful to all my friends who give me support during my

PhD life. They are Yin Shu, Zhaolong Han, Chen Wu, Peter Liu, Yu Zhang, Guoyan Cao, Jing Wang, Danni Li, Yingyu Li, Yun Hu, and Jia You.

Last but by no means least, I want to thank my old friends who grew up with me together and supported me always. They are Chufei Ouyang, Ming Lei, Weisheng Xie, Feipeng Jing, Kun Zhu, Lian Li, Yuan Zhan, Shiyue Huang, Ruoming Peng, Daiqian Zhang, Wei Qin, and Biao Tang.

BIG DATA ANALYSIS OF COMPLEX NETWORKS USING MACHINE LEARNING METHODS

An Abstract

of a

Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Electrical Engineering

by

Erte Pan

May 2016

Abstract

With the tremendous development of the modern complex networks such as smart grid and wireless communication domains, the data analysis tasks are significantly involved. In smart grid systems, there are emerging concerns on recognizing energy users' behavior patterns so that the energy trading companies are able to provide customized services. To understand users' usage patterns, efficient grouping methods are required such as clustering or nonparametric Bayesian models in machine learning field. In wireless communication field, a heat topic of locating personal devices and trajectory analysis is drawing more and more attention with the development of advanced personal devices such as the smart phones.

Given this background, this dissertation provides a theoretical research in smart grid systems and wireless communications networks with emphases on probabilistic clustering analysis, pricing scheme design, sublinear sampling, tensor voting theory and trajectory pattern recognition. The main contributions of this dissertation include: a comprehensive overview of basic concepts, models and state-of-the-art techniques used in smart grid and trajectory analysis is provided; a novel distance measurement for clustering analysis is proposed from the probabilistic point of view. Moreover, the stopping rules and clustering quality problems have been investigated with proposed novel metrics; the pricing schemes design has been formulated into an optimization problem. The novel sublinear sampling algorithm has been developed to address the computation efficiency in the context of big data; the tensor voting theory has been introduced to the trajectory inference problem and is implemented in the sparse sense to facilitate the computation. The fractal analysis has been employed as a novel method to extract trajectory features for trajectory pattern recognition tasks.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Complex Networks

World nowadays has witnessed the blooming of communication and information era. With the rapid development of advanced communication technology, social media and smart personal devices, various kinds of networks have evolved into more complex forms. Old fashioned electricity grid/networks have no functionalities of user-end to supplier-end communications. The information flow in the traditional energy grid system is only one directional, i.e., from users' usage data to the suppliers. Modern smart grid system has enabled the mutual communication for the both ends and brought new opportunities and challenges at the same time. With the information from the supplier's side, it is now possible for the energy users to understand their usage patterns and make their own choices towards several services to optimize their own benefits. On the other hand, the suppliers are now able to regulate the users' behavior and make the smart grid system more efficient. However, none of these aforementioned achievements can be realized without a deep analysis of the huge volume of smart meter data. The challenges come from multiple factors, such as the amount of data, various data sources, complexity of the networks and modelization. Similar challenges and opportunities lie in the field of communication networks with the revolution of personal smart devices such as smart phones. Locating devices is a key issue in many communication scenarios such as GPS tracking, residential security and so forth. Such kinds of trajectory analysis face the challenges including incomplete data, trajectory pattern recognition and so forth, which require advanced models and techniques.

In this thesis, we mainly focus our research on how to extract meaningful patterns of electricity users' usage and how to build an efficient pricing model for a smart grid system with fast computing speed and less computation burden based on the sublinear algorithm, which makes our model suitable in big data settings. We are going to present the methods on pragmatic feature extraction of smart meter data and propose our probabilistic clustering method with a novel monitoring criterion. We will address the pricing model based on the proposed sublinear algorithms that facilitate the computation and give guaranteed confidence. For the communication networking research, we will address the tracking problem, as an example to illustrate tensor methods, of inferring human mobility trace under the circumstance that the recorded location information exhibits missing and noisy data.

1.2 Smart Grid Networks

Smart grid introduces novel concepts on energy efficiency and conservation with the benefit of smart meters. Smart grid has enjoyed prevailing hype and gained worldwide recognition in the past few years. The interpretation of "smart" reflects the noticeable change in the way energy is generated, distributed, delivered, and consumed. Integrated of advanced electrical and communications infrastructures incorporated with modern process automation techniques, smart grid system is constructed such that two-way communication between electricity suppliers and residential electricity consumers is efficiently implemented. With these powerful characteristics brought by modern electrical technologies, smart grid significantly alters the way utility companies, governments, customers and business participants view about electricity transmission and its associated services.

The widely applied smart meters have shown their powerful capability in instant power usage recording and information communicating [1, 2]. The advanced metering

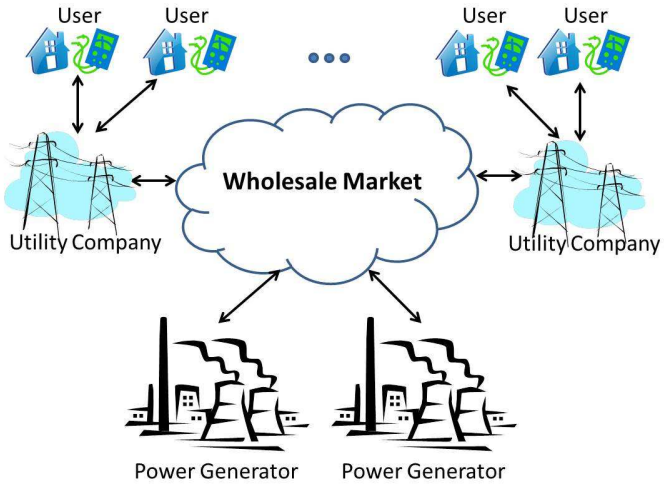

Figure 1.1: Advanced metering infrastructure (AMI) system in smart grid.

infrastructure (AMI) systems serve as the core of smart meter technique. The AMI systems are the composition of various components such as communications, consumer energy displays and controllers, and supplier business systems. Within the network, the collection and distribution of information have been effectively implemented, for instance, the time-of-use pricing information to customers, utility companies, and service providers. The employment of smart meters significantly alleviates the growing concern on energy conservation nowadays. With the benefit brought by smart meters, it is possible to record fine-grained power usages of electricity consumers and analyze their individual behaviors [3]. To this end, the pattern analysis based on the usage information is crucial to utility companies to offer user-oriented services which can achieve better power usage efficiency and economy benefits [4]. As a consequence, the dynamic pricing implemented based on the smart meter technique is able to regulate user behaviors, which can promisingly reduce the peak loads and accomplish the goal of energy conservation.

Facing the increasing concern on energy conservation all over the world, the power grids are currently undergoing substantial changes and upgrades. One important objective of the future grids is to provide a more customized electricity supply and

pricing approach that is suitable for different types of users. In many current markets, electricity is charged only by the amount of electricity used; without considering the time pattern a user consumes electricity or discriminating the peak or off-peak hours. We call this current pricing a *fixed-price* service. Such a fixed-price service has been adopted for decades. The development of smart meters makes it possible for the utility company to analyze users' behaviors, and therefore has the chance to offer load shape based pricing, referred as *differentiating user services*, i.e., pricing approaches that differ based upon when and how users consume electricity. With differentiating user services, the users can benefit from more choices to control their energy consumption and manage its cost. The utility company can also achieve better demand side management brought by these user behavior-oriented services and enjoy cost reduction when purchasing power in the peak hour from independent power providers in the wholesale power market. To this end, a methodology is presented on differentiating user services based on extracted characteristic consumer load shapes (usage profiles as a function of time) from a large smart meter data set. The distinct user subgroups are identified based upon their actual historic usage patterns, which are represented by the proposed electricity usage distributions. Since the big electricity user data cover millions of users and for each user the data are multi-dimensional and in fine-time granularity, we thus propose a sublinear algorithm to make the computation of the differentiating user service model efficient. The algorithm requests an input of only a small portion of users, and a sublinear amount of the electricity data from each of these selected users. We prove that the algorithm provides performance guarantees. Our simulated evaluation demonstrates the effectiveness of our algorithm and the differentiating user service model.

Being a heated topic ever since being proposed, smart grid research attracts many efforts year by year. Work in [1] gives a thorough summary on development, problems,

and applications of smart grid. Authors in [5] discuss the modern power delivery technologies. Work in [6] focuses on the privacy and security issues come along with smart grid. Work in [7] investigates smart grid technologies with an emphasis on communication components and the related standards.

Viewing from data analysis and model construction perspectives, many studies are dedicated to the topics of user behaviors classification and pricing strategies. Authors in [8] employ the fuzzy c-means clustering to disaggregate and learn energy consumption patterns in smart meter data. In [9], researchers propose a day-ahead pricing scheme taking user reactions and dynamic adjustment of price into account. Work in [10] provides a sophisticated model to address the evolution of energy supply, user demand, and market prices under real-time pricing in a dynamic fashion. There are also works involving game theory in smart grid applications such as [11], where the case of only one supplier and multiple users is studied. It is worthwhile to point out that previous studies commonly devise dynamic pricing from the perspective of time intervals, i.e., different hours/seasons are treated differently in the pricing model. However, different behaviors of users are rarely discussed in the previous studies. In this chapter, we will take a close look at the problem of designing a pricing scheme that differentiates users.

Various research efforts have been dedicated to the data classification and clustering analysis in the smart meter field [12]. Authors in [8] employed the fuzzy c-means clustering to disaggregate and learn energy consumption patterns in smart meter data. Authors in [13] present an efficient and practical hybrid supervised self-organizing map/Bayesian identifier for plugged-in electric loads (PELs). The proposed identifier can classify PELs into clusters by inherent similarities and provide the probability of the unknown load belonging to a specific type of load. Authors in [14] investigate how temporal resolution of power demand profiles affects the quality of the clustering pro-

cess, the consistency of cluster membership, and the efficiency of the clustering process. The authors select three types of clustering algorithms, including k-means, agglomerative hierarchical algorithms and Bayesian non-parametric statistics. Authors in [15] address the important issue of clustering regarding the quality assessment of the clustering results. Authors in [16] propose a modern approach based on multiple linear regression and hourly information to create accurate and defensible forecasts. Their work consists of three key components: predictive modeling, scenario analysis, and weather normalization.

Sublinear algorithms enjoy many studies from the theoretical point of view [17]. Given a big data trace, sublinear algorithms have been developed to check the quantile of the data [18], whether the data stream is periodic [19], etc. One study related to our thesis is [20], where sublinear algorithms are developed to check whether two distributions are close with certain confidence parameters. However, the algorithm is not suitable for our user classification task due to its inherent nature that the confidence parameters remain undetermined under some conditions. Hence, we propose our novel sublinear algorithm to overcome this drawback and apply it to the pricing problem in smart grid systems, where sublinear serves as an efficient tool to differentiate energy users based on their behavior and reduce the computation load.

1.3 Communication Networks

Communication network is used extensively throughout the world to connect individuals and organizations. With massive information flow, security issue has drawn significant attention. For instance, in flight security situation, the control base station would like to know the position or trajectory of the aircraft via the GPS communication. As for residential security scenarios, the users wish to track their personal devices or travel histories for private purposes. All these newly emerging requirements

demand a profound understanding of trajectory data and trajectory pattern analysis.

Tensor theories nowadays have been widely applied to engineering problems and big data applications. From the numerical point of view, tensors are the extension concept of scalar, vector and matrix. In addition to the mathematical formation, tensors can be employed with specific physical meanings under different contexts. For instance, a tensor can be used to represent a geometric object whose geometric property is invariant to the coordinate systems. Also, a tensor can be utilized to describe linear relation between vectors, scalars and other multidimensional arrays. The relation can be mathematically expressed as a multi-linear mapping. Initially appearing as an abstract object frequently used in math and physics, tensors have been attracting increasing interest in a broad range of research fields such as engineering and data science. However, few studies have addressed their application in networking scenarios. In this thesis, we investigate the wide applications of tensor techniques with an emphasis on the tensor voting method, which serves as an artificial intelligence approach for automatic inference and perceptual grouping. To illustrate the efficiency of the tensor voting approach, we tackle the tracking problem of inferring human mobility traces, which can provide key location information of networking objects. The trace inferring problem is considered under the circumstance that the recorded location information exhibits missing data and noise. Based on the tensor voting theory, we propose a sparse tensor voting algorithm and an implementation scheme with computational efficiency. The model is constructed based on the geometric connections between the input signals and encodes the structure information in the tensor matrix. The missing location information and noise can be distinguished via tensor decomposition. Once the trace information has been completed, further analysis of the inferred trace can be performed based on feature extraction to differentiate different objects. Moreover, we propose several feature extraction methods to characterize

the inferred trace, including the scale invariant feature obtained from fractal analysis. The proposed methods for trace completion and pattern analysis are applied to real human mobility traces. The results show that our proposed approach effectively recovers human mobility trace from the incomplete and noisy data input, and discovers meaningful patterns of inferred traces from various objects.

As a powerful and popular tool, tensor theories have been employed in various research and pragmatic fields. In mechanics, the stresses are presented by tensors, which brings about concise modelization and efficient computation [21]. In data science, tensors are utilized to model the data cube in which the inherent property of data is encoded and can be revealed through tensor decomposition [22]. In image processing [23], tensors can be used to model the geometric objects such as the normal space, which contributes to the inference of grouping points. As one of tensor theories, tensor voting [24] is the artificial intelligence technique widely used for automatic perceptual grouping where the tensor voting algorithm estimates and infers geometric objects based on the principles derived from human psychology. In this paper, we focus on the tensor voting algorithm and study the tracking problem as an illustration for the sake of rapid development of networking, machine learning and signal processing which bring up many tracking issues of constant interests.

Significant research efforts nowadays have been devoted to the challenges and problems in mobile networks as a result of fast growth of wireless networks [25, 26]. As the innovative development of personal devices such as smart phones, a prominent amount of location services are emerging to serve individual user for various purposes. For example, LTE-Direct and device-to-device require location information so as to enable discovering nearby devices and their services. However, the location information might be missing for indoor applications due to loss of signals or incomplete even for the outdoor GPS data. It is therefore important and challenging to retrieve the

complete device tracking information to realize various location-based functions of wireless networks. Intensive research efforts have been focused on the tracking problem from diverse aspects, ranging from mobility trace study to the image processing field. Work in [27] investigates the patterns of human walk traces from the statistical point of view. The synthetic features of human walk trace are captured by the proposed mobility model. [28] employs the random walks model called Levy-walk, to emulate the characteristics reflected in the walking patterns. It proves that the statistical similarity does exist between the random walk model and human walks.

1.4 Big Data Challenges

Big data is a popular studied topic recently [29]. The challenge comes from volume of the data and the variety of the data. Among many approaches, the sublinear algorithm [30] is a new paradigm to solve various big data problems. The essence of sublinear algorithm is to use a small portion of the data to compute results with guarantees. More specifically, the output of the sublinear algorithm is an approximation to the optimal result. As compared to approximation algorithms, which implicitly indicates that the approximation succeeds for 100% times, sublinear algorithms output an approximation with a $(1-\delta)\%$ (e.g., 95%) confidence to succeed. Such sacrifice in confidence makes sublinear possible.

1.5 Thesis Organization

In Chapter 2, we address the electricity user load profiling problem by developing a nonparametric clustering approach to differentiate various types of energy users based on efficient pragmatic feature extraction. We propose a novel distance metric referred to as the modified Mahalanobis distance for the clustering approach to extract the

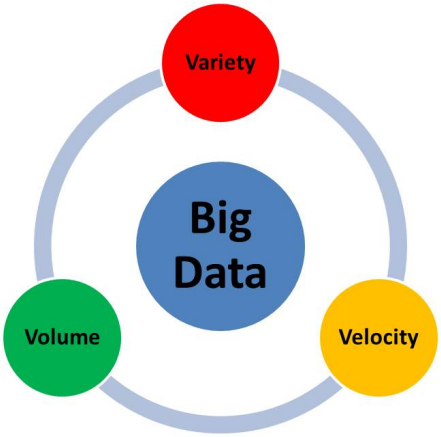

Figure 1.2: Key dimensions of big data.

prototype of user groups. In addition, we prosed several stopping rules and clustering quality measurements to assess the clustering performance.

In Chapter 3, The pricing model is constructed as an optimization problem which takes the consideration of different user behavior patterns and the cost for buying energy from the plants. The sublinear algorithms are proposed to deal with big data issues and enable the on-line differentiation of users.

The tensor voting method is illustrated in Chapter 4 with applications in the inference of human mobile traces. The proposed sparse tensor voting scheme automatically discovers the missing parts of the trajectory and completes the trace based on its geometric properties. Several trajectory features such as the fractal dimensions are proposed to be used in recognizing the trajectory patterns via supervised learning based on logistic regression.

Finally, conclusions and some possible future works are mentioned in Chapter 5.

Chapter 2

Hierarchical Clustering Analysis of Load Profiling for Big Smart Meter Data

Wide deployment of smart meters has incorporated efficiency into power systems, providing energy companies and grid operators with significant amounts of data. Specifically for Advanced Metering Infrastructure (AMI), we are faced with the challenges of big data analysis for user pattern recognition of smart meter data. One important objective is to identify coherent groups of users power consumption patterns. To achieve the objective, we address several major issues in this chapter. First, data preprocessing is enhanced by extraction of efficient and explicit pragmatic features by a multiscale analysis of load profiles and statistical regrouping of coarse grain outliers. Second, we introduce several key innovative techniques based on probabilistic algorithmics utilizing the modified Mahalanobis distance to efficiently implement automatic clustering of load profiles. The number of clusters is determined by applying gap statistic. Third, we propose a novel monitoring criterion referred to as the strong separation index to precisely evaluate the quality of automatic clustering results. Finally, through simulation of real data, our benchmark validated results should provide efficient methods to help utility and energy trading companies with insights on what types of consumers they have, a point which can be efficiently used for smart pricing based on consumer types. Our major contributions are:

1. We deal with the feature extraction problem by generating five explicit new features based on pragmatic coarse grain evaluations of power consumption patterns. This concrete feature extraction generates pragmatic coordinates for each user profile.

2. These pragmatic profile coordinates are analyzed by our proposed probabilistic clustering methods based on the modified Mahalanobis distance to discover efficient prototypes for user power usage profiles. The clustering problem is formulated as an optimization problem and the optimal solution is approximated by the Lloyd's algorithm. Moreover, the number of clusters is determined in a non-parametric fashion by utilizing the gap statistic. The corresponding clusters of user profiles should enable electricity business companies to efficiently implement dynamic pricing, customized offers, and so forth.

3. The performance of clustering results is quantified by a new probabilistic clusters separability criterion, namely the strong separation index, derived from precise statistical analysis of cluster data, and which replaces the cruder Euclidean distance metrics.

4. Our innovative approach is tested on a big real set of smart meter data. The numerical results show that our proposed methods achieve fast convergence speed with convincingly separation quality of different user groups. The experimental results also validate our proposed hierarchical clustering strategy which can produce more refined clustering analysis.

This chapter is organized as follows: Section 2.1 presents the motivation of our approach. Data preprocessing is addressed in Section 2.2. In Section 2.3, we present and discuss our stochastic clustering technique for smart meter big data. Experimental results are given in Section 2.4. In Section 2.5, we draw conclusions.

2.1 Motivation

In our benchmark smart meter data set, which involves hundreds of thousands of users and more than 12 months of data, the power consumption of each household

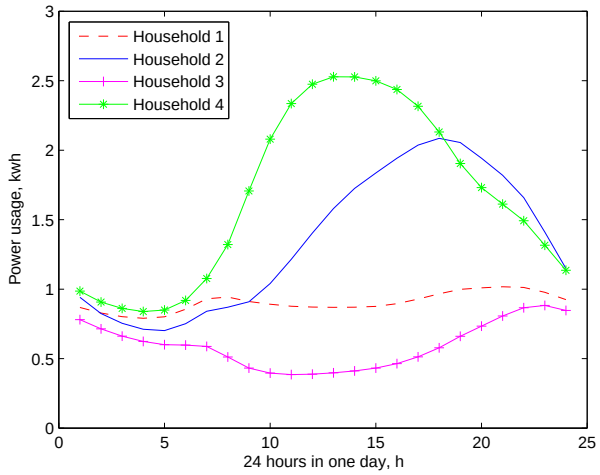

Figure 2.1: Examples of prototype user load profiles. These profiles are computed by averaging all users' profiles in distinguished user groups by our clustering analysis.

is recorded every 15 minutes, and we combine every successive four records into one hourly record. Each user user daily consumption profile is then a 24-dimensional vector, called the daily *load profile*. Several examples of profiles are presented in Fig. 2.1. Note that each household is representative of the group of users who share a similar behavior. In our benchmark data set, the proportion of smart meter recordings with partially missing information is less than 1% and we systematically clean away all corrupted data.

Users' behaviors differ significantly between weekdays and weekends. So we apply similar but separate analyses to weekday and weekend data separately. For brevity, we here focus our presentation only on weekday data. As is well known, total daily consumption is a key feature to differentiate users profiles. Thus for each user we compute the mean hourly consumption averaged over 5 weekdays. This is also a key quantifying factor for electricity business companies.

The shapes of daily users profiles are also of high practical interests for efficient regrouping of users. We point out that in our data set, the real dimensionality of users daily consumption profiles is 5 which is much smaller than the original 24 dimensions

as will be seen later on by PCA of the daily profiles.

In the next two Sections, we propose a multi-scale clustering strategy, which is summarized as:

1. Data preprocessing at coarse level: for each user compute TOT = average total daily consumption. Eliminate outliers having extreme TOT values. Perform 1 dimensional crude clustering at the TOT level.

2. Clustering at finer grain: the hierarchical stochastic clustering is implemented at successively finer scales. To efficiently cluster the data set in a stochastic way, the modified Mahalanobis distance is proposed from a probabilistic point of view. The proposed clustering technique is formulated as an optimization problem which is then solved based on the Lloyd's algorithm. To efficiently implement the Lloyd's algorithm, the gap statistic is employed to infer the pre-determination of the number of clusters. The stopping rules are proposed to terminate the clustering algorithm based on the stability of clustering structure. To monitor the quality of clustering procedure, a novel strong separation index is proposed which is derived from a probabilistic manner and is not dependent on the actual implementation of clustering algorithm.

2.2 Data Preprocessing

Denote by $\mathbf{x}^i$ the consumption profile of the i-th user, which is a vector in $\Re^{24}$. The average total daily consumption of user i is then $tot^i = \sum_{j=1}^{24} x_j^i$. Normalize this feature by $TOT^i = tot^i/mtot$, where $mtot$ is the mean of tot^i over all users. Let $dev(TOT)$ be the standard deviation of TOT. Let a and b be the 1% and 99% quantiles of the TOT values. Eliminate as outliers all users for which TOT^i is not within $[a, b]$.

We then implement a rough 1 dimensional clustering on the basis of the $\ln(TOT)^i$ values cluster via the Gaussian Mixture Model (GMM) [31], which models the distribution of $\ln(TOT)^i$ values as a mixture of M gaussian densities $f_1, \ldots, f_M$. The number M is determined by a quick analysis of the number of strong peaks observable on the histogram of all $\ln(TOT)^i$ values. Fig. 2.2 shows the histogram of $\ln(TOT)$ values. Classical variants of EM algorithms then provide the means $m_1 < m_2 < \ldots < m_M$ and standard deviations $\sigma_1, \sigma_2, \ldots, \sigma_M$ of $f_1, \ldots, f_M$. The best threshold T_j separating f_j from f_{j+1} is computed by solving in T the equation $f_j(T) = f_{j+1}(T)$. This defines M disjoint basic clusters B_j with $\ln(TOT)$ values lying between T_j and T_{j+1}. In our data set the histogram of $\ln(TOT)$ values gave us $M = 2$, and we thus generated two basic clusters B_1 and B_2. The algorithm for obtaining B_1 and B_2 is summarized as in Algorithm 1.

Algorithm 1: Algorithm for subgroups splitting via Gaussian fitting on $\ln(TOT)$.

for *Each possible value of* $T_j \in [min(\ln(TOT)), max(\ln(TOT))]$ **do**

1) Given the value of T_j for thresholding, all users are temporarily divided into group B_1 or group B_2 by testing his/her $\ln(TOT)^i$ against T_j.
2) Within each group, the Gaussian densities $f_1(\cdot)$ and $f_2(\cdot)$ are fitted and the parameters (μ_1, σ_1) and (μ_2, σ_2) are obtained by Maximum likelihood Estimation (MLE).
3) The optimal T^* is obtained by solving the equation $f_1(T^*) = f_2(T^*)$.
4) Keep iterating until $T_j = T^*$.

2.3 Clustering Analysis

Once the previous basic subgroup B_j has been obtained, further clustering can be applied. Clustering analysis is implementable by multiple methods, involving two core concepts, namely the representative prototype of each cluster and the quantification of similarity between pairs of load profiles. Classical prototypes are defined as the

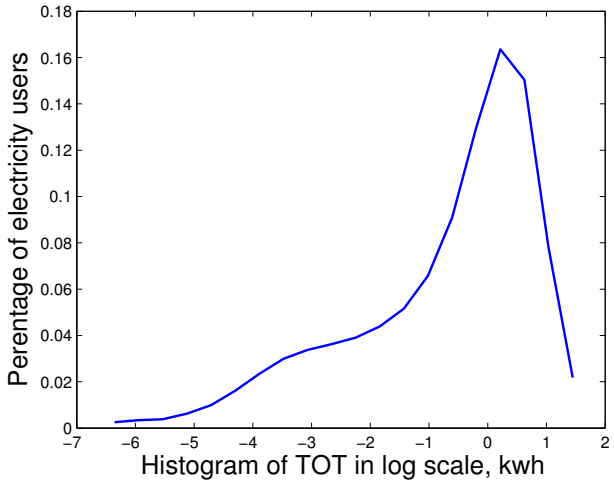

Figure 2.2: Empirical distribution of $\ln(TOT)$ which we model by a mixture of two Gaussian densities in order to split the set of users into two disjoint clusters B_1 and B_2.

mean vectors of each cluster. Similarity between a daily load profile and a prototype profile is often quantified by their Euclidean distance. Clustering methods such as K-means essentially minimize the adequate sums of squared Euclidean distances in the input data space. However, one drawback of several classical data-space clustering techniques is that separation boundaries are linear, which can often fail to fit the true boundaries of real data clusters. Sum-of-squares criteria that assign equal importance to each dimension of the input data are not well adapted to smart meter applications. To implement nonlinear separation of profile data, we propose a probabilistic extension of K-means clustering.

In order to efficiently employ our proposed probabilistic clustering technique based on modified Mahalanobis distance, the input data must first be de-correlated by PCA to linearly project the input data into a lower feature space. The linear mapping ϕ obtained by PCA transforms each given profile $\mathbf{x}$ into a mapped "feature space" point denoted $\phi(\mathbf{x}) \triangleq \mathbf{X}$, where $\mathbf{x} \in \Re^{24}, \mathbf{X} \in \Re^{D}$ and $D < 24$. To cluster the mapped feature space points, we apply a probabilistic clustering technique, which formulates the clustering problem as a log-likelihood maximization problem.

In the following subsections, we first present our probabilistic clustering technique based on the modified Mahalanobis distance. We then introduce the stopping rules based on the stability of clustering structure and the strong separation index to monitor the clustering quality. To make the proposed clustering method more efficient, we describe the data compression method by key pragmatic coordinates for the clustering algorithm input. In the end of this section, we summarize the overall algorithmic flow.

2.3.1 Modified Mahalanobis Distance and Clustering Algorithm

We start with formulating the clustering problem by introducing the concept of within-cluster scatter, which is a measure of compactness given the current partition of clusters. Given the number of clusters M, the within-cluster scatter is defined as

$$S = \frac{1}{N} \sum_{m=1}^{M} \sum_{n=1}^{N} z_{m,n} d(\mathbf{X}_n, \mathbf{o}_m). \tag{2.1}$$

The binary scalar $z_{m,n}$ are defined by : $z_{m,n} = 1$ if $\mathbf{X}_n$ is in the m-th cluster CLU_m, and $z_{m,n} = 0$ otherwise. The centroid C_m of CLU_m is defined by $\mathbf{o}_m = \frac{1}{N_m} \sum_{n=1}^{N} z_{m,n} \mathbf{X}_n$ where N_m is the number of points for the m-th cluster. The distances $d(\mathbf{X}_n, \mathbf{o}_m)$ will be defined here by the following *log-likelihood* expressions as

$$d(\mathbf{X}_n, \mathbf{o}_m) = (\mathbf{X}_n - \mathbf{o}_m)' \Sigma_m^{-1} (\mathbf{X}_n - \mathbf{o}_m) + \ln(|\Sigma_m|), \tag{2.2}$$

where Σ_m is the covariance matrix of cluster CLU_m. Transposition is denoted by $'$ and $|\Sigma_m|$ is the determinant of Σ_m. We can clearly consider these distances as the modified Mahalanobis distances.

The modified Mahalanobis distance is essentially a Gaussian log-likelihood. In the feature space $\Re^D$, assume that for the cluster CLU_m the data points distribution has a multivariate Gaussian density g_m with mean $\mathbf{o}_m$ and covariance matrix Σ_m. Hence,

Algorithm 2: Probabilistic clustering.

1) The initial centroid of each cluster is randomly selected.
2) Each mapped feature space point $\mathbf{X}$ is then assigned to i-th cluster CLU_i if the centroid of CLU_i is closest to $\mathbf{X}$, according to the modified Mahalanobis distance computed by (2.2) as above.
3) Centroids are updated after all mapped feature space points have been clustered.
4) Keep iterating until all clusters stabilize according to stopping rules given in the next subsection.

the log-likelihood that data point $\mathbf{X}_n \in \Re^D$ belongs to cluster CLU_m is proportional to the log-density, and hence given by

$$\begin{aligned} \ln[g_m(\mathbf{X}_n)] = &- \frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_m|) \\ &- \frac{1}{2}(\mathbf{X}_n - \mathbf{o}_m)'\Sigma_m^{-1}(\mathbf{X}_n - \mathbf{o}_m). \end{aligned} \tag{2.3}$$

Maximizing the log-likelihood of the actual assignations of all data point to current clusters is then equivalent to selecting binary z_{mn} minimizing the cost function given in (2.2). To our best knowledge, the use of the modified Mahalanobis distance is novel in smart grid research.

The clustering result is thus obtained by solving the optimization problem

$$\mathbf{Z} = \arg\min_{\mathbf{Z}} S, \tag{2.4}$$

where $\mathbf{Z}$ is the matrix $[z_{m,n}]$ and S is the within-cluster scatter as in (2.1). Since this optimization is NP-hard, we apply the Lloyd's algorithm to approximate the indicator matrix $\mathbf{Z}$ as summarized in Algorithm 2.

The Lloyd's algorithm requires the pre-determination of the number M of clusters. To compute an initial quick estimate for M, one can apply the so called "gap statistic" [32]. The approach is inspired by the elbow phenomenon: the compactness measure S decreases monotonically as the number of clusters M increases, but the decreasing speed of the compactness measure drops significantly from a certain value of M onwards. In other words, the decreasing curve exhibits the elbow shape where

the corresponding M is supposed to be the optimal. The gap statistic formalizes this heuristic hint into a statistical metric by comparing $\log(S)$ with its expectation under an appropriate null reference distribution of the data. The gap metric is defined as

$$Gap(M) = E^*\{\log(S)\} - \log(S), \tag{2.5}$$

where E^* is the expectation over N mapped feature points sampled from the reference distribution. The implementation of gap statistic is summarized as: First, the proposed clustering technique based on the modified Mahalanobis distance is performed to the observed feature points in the feature space by varying the number of clusters from $m = 1, 2, ..., M$, producing the within-cluster cost S as a function of m. For comparison, N feature points are then generated from the uniform distribution over the range of the observed feature points. The sampling is repeated B times, producing the B reference data sets where the same K-means approach is applied and the gap metric is estimated as

$$Gap(m) = \frac{1}{B}\sum_{b=1}^{B}\log(S^*) - \log(S). \tag{2.6}$$

The standard deviation of the estimated expectation is given as

$$\sigma_m = \left[\frac{1}{B}\sum_{b=1}^{B}\{\log(S^*) - \bar{y}\}^2\right]^{1/2}, \tag{2.7}$$

where $\bar{y} = \frac{1}{B}\sum_{b=1}^{B}\log(S^*)$. Denote $\hat{\sigma}_m = [1 + 1/B]^{1/2}\sigma_m$. Considering the simulation error contained in the estimated expectation term, the optimal number of clusters is then determined by the smallest m which satisfies

$$Gap(m) \geq Gap(m+1) - \hat{\sigma}_{m+1}. \tag{2.8}$$

2.3.2 Stopping Rules and Clustering Quality

We now present the stopping rules monitoring our probabilistic clustering algorithm, to automatically detect stabilization of iterative clustering. Define $CLU_m(k)$

as the m-th cluster generated at the k-th iteration step. Let $\Delta_m(k)$ be the size of the symmetric set difference between the two sets $CLU_m(k)$ and $CLU_m(k+1)$. Define the *clustering shift* $\xi(k)$ at iteration k by

$$\xi(k) = \max_m \frac{\Delta_m(k)}{|CLU_m(k)|}, \tag{2.9}$$

where the cardinal of any set $CLU_m(k)$ is denoted $|CLU_m(k)|$. In our implementation of iterative probabilistic clustering, we stop the iterations as soon as $\xi(k) < 0.1\%$. Smaller thresholds can be used at higher computing cost. We also impose an a priori maximum on the number of iterations.

To monitor the quality of a given clustering, we focus on how well the clusters are separated from each other. To this end, we introduce a novel *strong separation index* $SEP(C, K)$ between clusters C and K. This index is defined and computed by the Algorithm 3. Let us interpret intuitively the index $SEP(C, K)$. When the two clusters C and K are well separated, most of the points in cluster C must be at "rather large" modified Mahalanobis distances of the centroid of cluster K. We quantify "rather large" in terms of the outliers observed within the cluster K. This notion is then symmetrized between C and K.

Once our iterative probabilistic clustering has stabilized according to the stopping rules given above, we evaluate the quality Q of the terminal clustering $CLU_1, \ldots, CLU_M$ by

$$Q = \max_{1 \le i < j \le M} SEP(CLU_i, CLU_j). \tag{2.12}$$

Typically, small values of Q such as Q less than 5% or 10% should be considered as good clustering results in multi-dimensional data. Our strong separation index is not dependent on the actual clustering algorithm, and hence can serve as a probabilistic performance evaluation for any clustering technique.

Algorithm 3: Strong Separation Index $SEP(C, K)$ between clusters C and K

1) Within any cluster C with centroid $\mathbf{o}_C$, and for all feature space data points $\mathbf{X}_n$ belonging to C, use formula 2.2 to compute all the modified Mahalanobis distances $d(n, C) = d(\mathbf{X}_n, \mathbf{o}_C)$.
2) For the cluster C, compute $F_C(\cdot)$ which will be the Cumulative Distribution Function (CDF) of all the distances $d(n, C)$ just evaluated in step 1.
3) Within cluster C, define and extract the core cluster cr_C as the set of all feature space points $\mathbf{X}_n$ belonging to C and verifying $F_C(d(n, C) < 90\%\}$.
4) For any cluster K compute similarly the CDF $F_K(\cdot)$ and extract its core denoted cr_K.
5) Compute the two indicators $s(C, K)$ and $s(K, C)$ defined by

$$s(C, K) = \max_{\mathbf{X}_n \in cr_K} [1 - F_C(d(n, C)], \tag{2.10}$$

$$s(K, C) = \max_{\mathbf{X}_n \in cr_C} [1 - F_K(d(n, K)]. \tag{2.11}$$

6) The symetrized strong separation index between clusters C and K is then $SEP(C, K) = \max[s(C, K), s(K, C)]$.

2.3.3 Data Compression by Key Pragmatic Features

Many clustering methods can directly deal with raw input data. However, clustering quality and computing cost generally improve when high dimensional data are adequately compressed.

For our application to smart meter data where each user is represented by a daily load profile which is a 24-dim vector, it is far better to extract a few key pragmatic coordinates rather than directly input the raw 24-dim data into the clustering algorithm.

For each 24-dim load profile $\mathbf{x}$, we define 4 new features by:

total daily consumption : $tot(\mathbf{x}) = x_1 + ... + x_{24}$,

night-time consumption : $nit(\mathbf{x}) = x_1 + ... + x_8$,

day-time consumption : $day(\mathbf{x}) = x_9 + ... + x_{17}$,

evening consumption : $eve(\mathbf{x}) = x_{18} + ... + x_{24}$.

We then relativize the 3 variables nit, day, eve with respect to total daily consumption by

$$rel.nit = nit/tot \ ; \ rel.day = day/tot \ ; \ rel.eve = eve/tot.$$

Finally we normalize each one of the 4 variables $tot, rel.nit, rel.day, rel.eve$ by dividing them by their respective means across all users, which generates the 4 normalized variables TOT, NT, DT, ET. These 4 features can be considered as the coarse grain profile features.

At finer time scales, we define several pragmatic coordinates refining NT, DT, ET. Define bNT and eNT as the two respective sums of the first 4 hours and of the last four hours of NT. Let then $bnt = bNT/mean(bNT)$ and $ent = eNT/mean(eNT)$ where means are computed cross users. Similarly, we derive from DT and ET the pragmatic finer normalized coordinates bdt, edt, bet, eet.

By examining the correlation matrix of our 10 pragmatic coordinates, i.e. TOT, NT, DT, ET, bnt, ent, bdt, edt, bet and eet, we select 5 pragmatic coordinates bdt, NT, bnt, ET and bet which are highly decorrelated and which provide a good approximation of the input vector space by a 5 dimensional feature space V. Our probabilistic clustering algorithms are then applied in the feature space V.

2.3.4 Summary of Overall Algorithmic Flow

The entire algorithmic flow is summarized as in Algorithm 4:

2.4 Numerical Results and Discussions

We have implemented the preceding algorithmic flow on a benchmark smart meter data set gathered from a large set of hundreds of thousands of Houston area electricity users. After elimination of all corrupted data we are confronted to daily load profiles,

Algorithm 4: Overall algorithmic flow for our probabilistic clustering approach.

1) Eliminate all corrupted smart meter recordings and generate all daily load profiles in dimension 24 as stated in Section 2.2.
2) Compute the 10 pragmatic features as described above in Section 2.3.3.
3) Study the histogram of $\ln(TOT)$ values to detect outliers with either extremely small or extremely large total daily consumption (see Section 2.2).
4) GMM analysis of the TOT values to implement subgroup splitting of data set (see Algorithm 1).
5) Exploratory PCA of the 10 pragmatic features to determine the an explicit feature space V in reduced dimension as illustrated at the beginning of Section 2.3.
6) Compute an exploratory gap statistic to roughly estimate the optimal number M of clusters (see Section 2.3.1).
7) Iterative probabilistic clustering with modified Mahalanobis distance in the feature space V (see Algorithm 2).
8) Compute quality of terminal clustering by strong separation index (see Algorithm 3).

which are vectors in $\Re^{24}$.

We then compute our 10 pragmatic features and first study the histogram of $\ln(TOT)$ values to truncate away all outliers with either extremely small or extremely large total daily consumption. Histogram analysis reveals that about 5% of users have extremely low total daily consumption. And we truncate them away into a special cluster of "inactive" users.

For the remaining users, we compute the standard quantiles q_k of TOT values for $k = 0\%, 5\%, 10\%, \ldots, 100\%$ and use these quantiles to split the users into 20 subsets of equal size according to the positioning of their TOT value within the quantile sequence.

Within each one of these 20 small subgroups, the standard deviations of TOT, NT, DT and ET are computed and plotted in Fig. 2.3. These plots indicate that the two extreme subgroups $TOT < q_{5\%}$ and $TOT > q_{90\%}$ have prominent larger variances than the other subgroups, and thus we also truncate away these users into two "TOT-extreme" clusters.

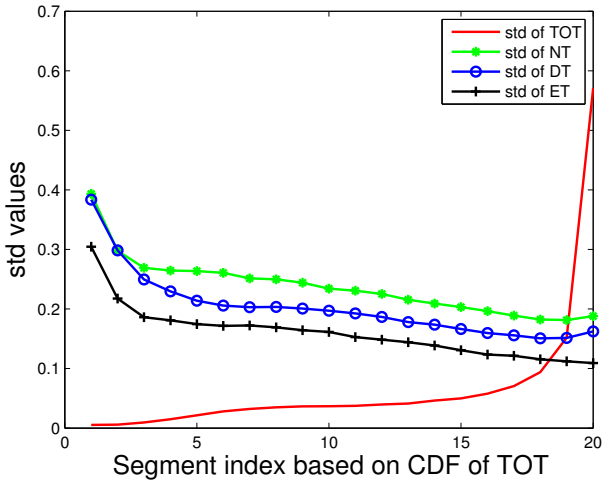

Figure 2.3: Standard deviation plots of DT, ET, NT and TOT vs. index of sub-set.

For the pruned data set now containing only active user users, we generate a new 5-dimension data vector from each user load profile, based on the 5 key pragmatic features defined in Section 2.3.3 as: bdt, NT, bnt, ET and bet.

To compute the "gap statistic", we repeatedly generate $B = 20$ reference data sets from the uniform distributions as described in Section 2.3.1. In order to better interpret the results, we define the residual value $Res(m)$ as $Res(m) = Gap(m) - Gap(m+1) + \hat{\sigma}_{m+1}$ and plot the curve of $Res(m)$ versus the number of clusters m as shown in Fig. 2.4. Computation of the "gap statistic" indicates that in this pragmatic feature space $V = \Re^5$, the initial cluster number should be around 3.

We then apply our iterative probabilistic clustering algorithm in this 5-dimension feature space to generate $M = 3$ clusters. The stopping rules are set at 300 for the maximum iteration number and 0.1% for the clustering shift $\xi(k)$ at terminal iteration k. The fast stabilization of our probabilistic clustering algorithm is indicated by Fig. 2.5, which plots $\xi(k)$ versus k. The fast convergence shows that our probabilistic clustering can be scalable for big data computations.

To visually check the quality of the terminal clustering CLU_1, CLU_2, CLU_3 just obtained, we randomly sample 3,000 data points from each one of these 3 clusters

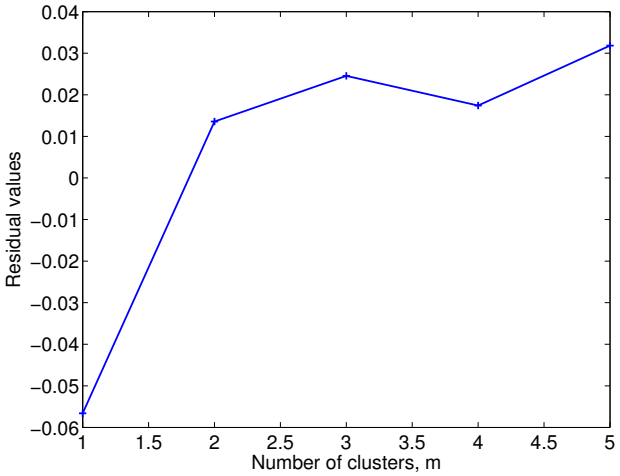

Figure 2.4: Gap statistic as shown in the curve of residual values $Res(m)$ versus the number of clusters m. The optimal m is indicated around 3.

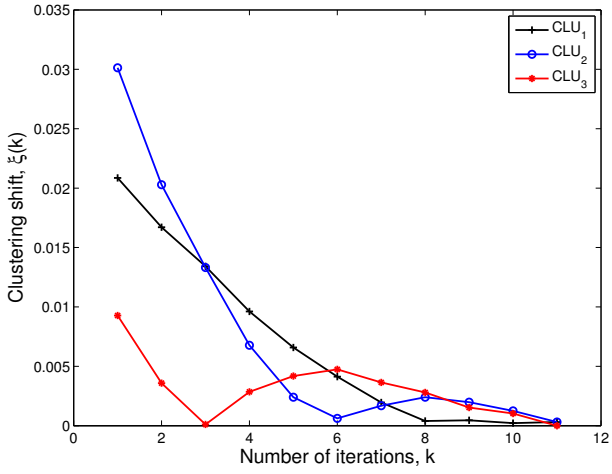

Figure 2.5: Clustering shift $\xi(k)$ vs iteration number k for the 3 terminal clusters generated in feature space V by probabilistic clustering.

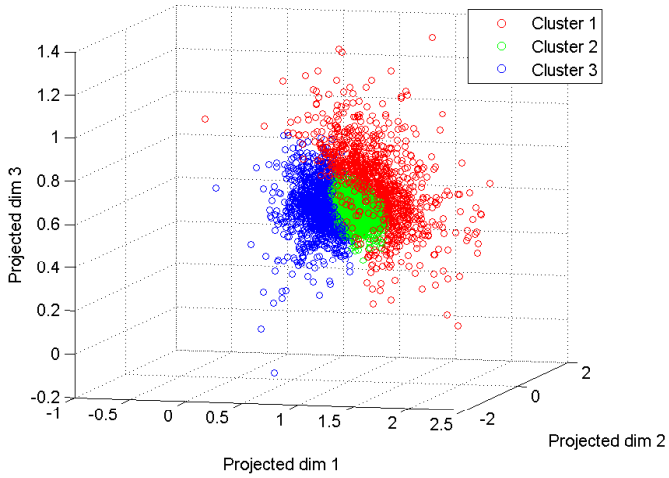

Figure 2.6: Scatter plot to visualize the 3 terminal clusters generated in feature space V by probabilistic clustering. The visualization is generated in $\Re^3$ by PCA projection from V onto the main 3 PCA coordinates

and project them onto the first 3 eigenvectors computed by PCA. In Fig. 2.6, we display this 3-dimension scatter plot, which shows that our terminal 3 clusters are well separated by non-linear boundaries.

The quality of this terminal clustering is evaluated by the three strong separation indices $SEP(CLU_i, CLU_j)$ where $1 \leq i < j \leq 3$ between our 3 pairs of clusters. These 3 separability values are all $\leq 5\%$, which indicates a good separation of clusters, and are given in Table 2.1.

In Fig 2.7, we plot the mean load profile (in dimension 24) for each one of our 3 terminal clusters, as well as the mean load profile of the whole users data set. These plots show that the main differences in the mean load profiles of our 3 clusters emerge either during morning or between hours 13:00 to 20:00.

Our clustering strategy is essentially hierarchical. Starting with our 3 "coarse grain" terminal clusters CLU_j, we proceed to try to further split each CLU_j into smaller groups as long as further splitting is validated by the strong separation index.

For instance for cluster CLU_2 which represents 33% of our whole data set, we implement our probabilistic clustering of CLU_2 into 2 terminal sub-clusters CLU_{21}

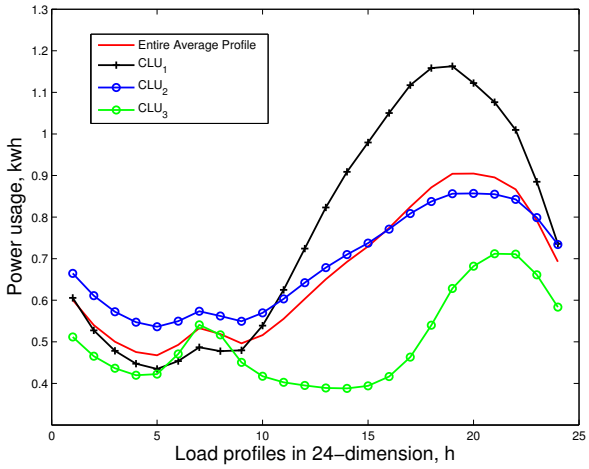

Figure 2.7: Load profiles of the terminal 3 clusters generated by iterative probabilistic clustering algorithm in feature space.

Table 2.1: Strong separation indices between clusters : $SEP_{ij} = SEP(CLU_i, CLU_j)$.

Cluster Index (i,j)	(CLU_1, CLU_2)	(CLU_1, CLU_3)
SEP_{ij}	0.03	0.05

Cluster Index (i,j)	(CLU_2, CLU_3)
SEP_{ij}	0.05

and CLU_{22}, of respective sizes 51% and 49% within CLU_2. The stabilization of sub-clustering CLU_2 is shown in Fig. 2.8. As can be seen, the sub-clustering converges fast and the clustering shifts $\xi(k)$ for both sub-clusters are close to each other because the sizes of two sub-clusters CLU_{21} and CLU_{22} are close. The pairwise separation indices between CLU_{21}, CLU_{22}, $CLU1$, CLU_3 are given in Table 2.2. Clearly CLU_{21} and CLU_{22} are very well separated with separation index close to 0, and remain also well separated from CLU_1 and CLU_3. These observations from Table 2.2 indicate strong evidence to further split CLU_2. This validates the hierarchical clustering of CLU_2. The mean load profiles of CLU_{21} and CLU_{22} in dimension 24 are plotted in Fig. 2.9 which shows significant differences between these two sub-clusters.

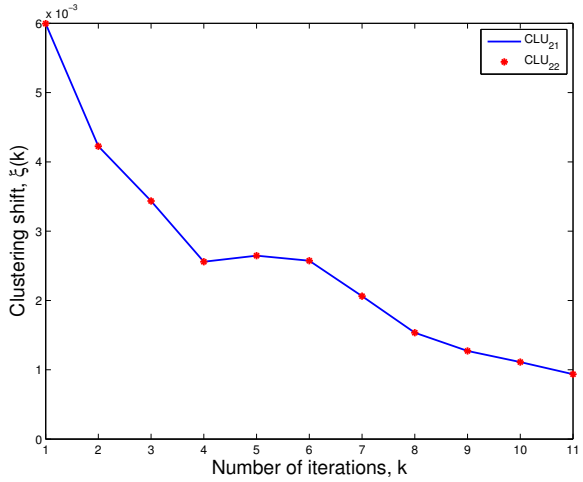

Figure 2.8: Clustering shift $\xi(k)$ vs iteration number k for sub-clustering CLU_2.

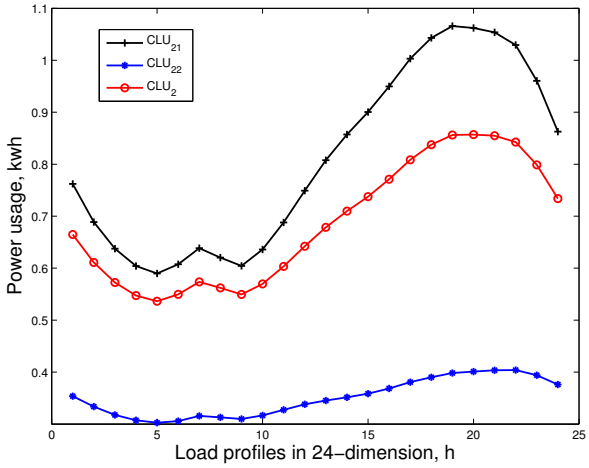

Figure 2.9: Load profiles of the three clusters CLU_2, CLU_{21} and CLU_{22}, generated by splitting of cluster CLU_2 using iterative probabilistic clustering.

Table 2.2: Strong separation indices between sub-clusters.

Cluster Index (i, j)	(CLU_1, CLU_{21})	(CLU_1, CLU_{22})
SEP_{ij}	0.03	0.03
Cluster Index (i, j)	(CLU_1, CLU_3)	(CLU_{21}, CLU_{22})
SEP_{ij}	0.05	0
Cluster Index (i, j)	(CLU_{21}, CLU_3)	(CLU_{22}, CLU_3)
SEP_{ij}	0.05	0.05

2.5 Summary

In this chapter, we address the electricity user load profiling problem involved in the analysis of real big data recorded from smart meter reading. We develop a non-parametric clustering approach to differentiate various types of energy users. Our approach involves several key technical ideas as follows: First, an efficient pragmatic feature extraction by multi-scale analysis of 24 dimensional daily load profiles, beginning at coarse grain levels. Second, a novel iterative probabilistic clustering algorithm based on the modified Mahalanobis distances between load profiles in feature space and log-likelihood optimization. This algorithm converges rather fast to well separated clusters and can be scalable for big data sets. Third, a new strong separation index to characterize the separability of any pair of clusters by precise statistical analysis. Fourth, a hierarchical approach to automatic clustering by starting with the generation of a small number of clusters at coarse grain scales and iteratively attempting sub-cluster splitting at finer scales. Finally, the simulation results on our large benchmark data set of smart meter data indicate that our approach is easily implementable at the computational level and performs quite convincingly on our benchmark data set.

Chapter 3

Analyzing Big Smart Metering Data Towards Differentiating User Services: A Sublinear Approach

With the advances of the information and communications technology, and smart meters in particular, fine grained user electricity usage of households is available for analyzing electricity usage behaviors. The information makes it possible for utility companies to provide differentiating user services from the time-of-use perspective, i.e., different pricing for users based upon when and how users consume power. In this chapter, we present a methodology on differentiating user services based on extracted characteristic consumer load shapes (usage profiles as a function of time) from a large smart meter data set. We identify distinct user subgroups based upon their actual historic usage patterns, which are represented by the proposed electricity usage distributions. Since the big electricity user data cover millions of users and for each user the data are multi-dimensional and in fine-time granularity, we thus propose a sublinear algorithm to make the computation of the differentiating user service model efficient. The algorithm requests an input of only a small portion of users, and a sublinear amount of the electricity data from each of these selected users. We prove that the algorithm provides performance guarantees. Our simulated evaluation demonstrates the effectiveness of our algorithm and the differentiating user service model.

The remaining part of this chapter proceeds as follows. In Section 3.1, we present a data trace study and clarify how we characterize the users by their electricity distributions. Section 3.2 is devoted to the differentiating user service model. We also show

that the complexity to compute the model is non-trivial. We then develop a novel sublinear algorithm in Section 3.3 and prove its performance bounds. In Section 3.4, we evaluate our algorithms through simulation, and finally, we conclude our paper in Section 3.5.

3.1 The User Electricity Usage Behavior and a Data Trace Study

In this section, we first propose a model to characterize user electricity usage patterns. We then use real user data to validate our model.

In this paper, we classify users according to their electricity usage distribution. A distribution in this paper is defined as a probability density function of a continuous/discrete random variable, which describes the likelihood for this random variable to take on a given value. We admit that there are many ways to characterize a user; for example, the total or average electricity he/she consumes in a month, the peak hour electricity usage in week days, etc. We believe the electricity usage distribution can more accurately characterize a user because a distribution provides a full spectrum of the user electricity usage. For instance, it is straightforward to see that if the usage distributions of two user are the same, the corresponding averages/expectations are also the same. And the peak hour usage of two users are close in the probabilistic sense; yet if the averages/expectations or peak hour usage are the same, their distributions can be different.

Formally, let set $\mathcal{S}$ be an element set from the set $\mathcal{T}$ indicating time instants. In other words, $\mathcal{T}$ is the set of daily sampling frequency or sampling time mark at different scales. In this paper, $\mathcal{T} = \{t_1, t_2, t_3\}$ with $t_1 = \{0\text{min}, 15\text{min}, 30\text{min}, ..., 1425\text{min}\}$, $t_2 = \{0\text{hour}, 1\text{hour}, ..., 23\text{hour}\}$ and $t_3 = \{1\text{day}\}$. Let $\mathbf{x}$ be a multi-dimensional random

variable representing the daily electricity usage of a user, such that $\mathbf{x} \in \Re^{D(\mathcal{S})}$, $D(\mathcal{S}) \in \mathbb{Z}$ and $\mathcal{S} \in \mathcal{T}$, where $D(\mathcal{S})$ is the dimension of $\mathbf{x}$ at the scale $\mathcal{S}$ with $D(\mathcal{S}) = Card(\mathcal{S})$. $\mathcal{S}$ is called the scale indicator. Hence, we obtain three different descriptions for the same one data vector: at the scale $\mathcal{S} = t_1$, $\mathbf{x} = (x_1, .., x_{96})$; at the scale $\mathcal{S} = t_2$, $\mathbf{x} = (\hat{x}_1, .., \hat{x}_{24})$; at the scale $\mathcal{S} = t_3$, $\mathbf{x} = \tilde{x}$. And there exists the relationship: $\hat{x}_i = \sum_{j=4\times(i-1)+1}^{j=4\times i} x_j$ and $\tilde{x} = \sum_{j=1}^{24} \hat{x}_j$. Based on the daily usage pattern $\mathbf{x}$ of the user, we define the feature referred as "usage distribution" to characterize a user's behavior from the statistical point of view as the following:

Definition 1. *The electricity usage distribution $P\{\mathbf{x}\}$ at scale $\mathcal{S}$ is a distribution on the daily electricity usage $\mathbf{x}$, where $\mathbf{x} \in \Re^{D(\mathcal{S})}$.*

Given the historian recordings of one-dimensional random variable $\mathbf{x}$, the electricity usage distribution $P\{\mathbf{x}\}$ can be approximated using the empirical distribution/histogram. For multi-variate case where $D(\mathcal{S}) > 1$, $P\{\mathbf{x}\}$ can be estimated by assuming a certain structure (for instance Multi-variate Gaussian distribution) and fitting the parameters based on the dataset. From our trace data, we find that though the exact electricity usage of each household differs, many households share the same distribution. This becomes the basis for our classification. To represent the electricity usage distribution of each category, we choose to use a benchmark distribution. Formally, a benchmark distribution is defined as:

Definition 2. *A benchmark distribution is an electricity usage distribution $P\{\mathbf{x}\}$ which corresponds to the prototype of a group of users sharing similar electricity usage patterns. It has the expectation $\mathbf{p} = E\{\mathbf{x}\}$ with fixed values derived from real data statistics, satisfying $\mathbf{p} \in \Re^{D(\mathcal{S})}$, $D(\mathcal{S}) \in \mathbb{Z}$ and $\mathcal{S} \in \mathcal{T}$, such that each of $\mathbf{p}^i$, $i = 1, ..., D(\mathcal{S})$ is a fixed value.*

We now validate the electricity usage distribution, and the benchmark distribution. In Fig. 3.1, we show an illustration of different average daily usage patterns of two

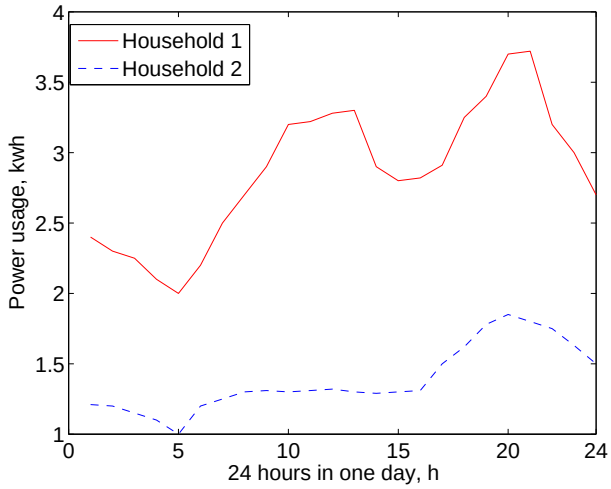

Figure 3.1: Illustrative example of different average daily usage patterns of two benchmark distributions.

benchmark distributions at scale $S = t_2$. As can be seen, there are differences in peak hours and peak usage between the two benchmark distributions. In our real data analysis, it is also discovered that even though some users' average daily behaviors are similar (i.e. similar peak hours and peak usage), their usage distributions are quite different: for instance, some users have no power consumption at all during the weekends, but their average daily behaviors are still similar to those who consume power constantly throughout each day; some users mainly consume the power in the summer and end up with the similar average daily pattern with those who frequently go out during summer season and use power mostly in winter. At scale $\mathcal{S} = t_2 = \{0\text{hour}, 1\text{hour}, ..., 23\text{hour}\}$, through the histogram analysis on real data variate-wisely, it is revealed that each of the 24 variates conforms approximately to a Gaussian distribution. The users shared with similar average daily behaviors may end up with close Gaussian means but different variances in each dimension. All these demonstrate that using distribution to classify users is reasonable. In order to classify users by their electricity usage distributions, we take advantage of the benchmark distributions that are predefined by the utility company. We will detail the method in the next section.

3.2 Differentiating User Services: The Model and Big Data Challenge

3.2.1 An Overview

Many works nowadays study pricing strategies for utility companies [33, 34]. The overall model can be abstracted as follows. For the following discussion, let us assume that one objective of a power marketer is to maximize its potential profit from its customer base. The profit equals its revenue minus its cost. The revenue of a utility company depends on its *pricing strategy* (sometimes also called as pricing coefficient), i.e., the unit price for electricity. For a fixed price strategy, the unit price for a unit electricity at any time, for any household is fixed. Newly proposed pricing strategies have dynamic unit price according to different situations. For example, the pricing strategy in [33] is implemented as a quadratic cost function of energy usage at each hour. This kind of increasing, convex cost function is reported by the literature to be a suitable model for thermal generators; and the pricing strategy in [34] is expressed as piecewise linear functions whose pricing coefficients stay as different constant values in different time intervals within 24 hours.

In this paper, we introduce a differentiating user service model that seeks to classify customers based upon their characteristic usage as defined by historic usage distribution. To operationalize this model, the utility must 1) classify different users (by setting benchmark distributions); and 2) set different pricing coefficients for different users. Clearly, both factors influence the revenue of the utility company. Here, we try to limit the scope of our study where we assume that these two factors are given. These two factors can be determined by certain optimization criteria [35], gaming with other utility companies [36], or other external concerns of the utility company [7].

With these two factors, we present a differentiating user service model for the

utility company to compute its revenue and profit in Section 3.2.2. We observe that such computation is itself non-trivial because it involves big data. We analyze the data complexity of this model in Section 3.2.3. In Section 3.3, we will develop sublinear algorithms that can efficiently conduct such computation.

3.2.2 The Differentiating User Service Model

We now formally present our differentiating user services model. Our differentiating user service model has three elements: 1) A set of benchmark distributions $\{P_1\{\mathbf{x}\}, P_2\{\mathbf{x}\}, ..., P_i\{\mathbf{x}\}\}, i = 1, ..., L$ with corresponding expectations $\{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^i\}$ where L is the total types of users. These expectations are used for the utility company to specify a pricing plan; 2) pricing coefficients: the unit pricing rate ($dollar/kw \cdot h$) for different types of users; and 3) power cost coefficients: the utility company needs to pay for buying the electricity from the power plants. We denote the user type indicator as the binary scalar $z_{m,n}$: $z_{m,n} = 1$ if the n-th user is in the m-th category with benchmark distribution $P_m\{\mathbf{x}\}$ and expectation $\mathbf{p}^m$; $z_{m,n} = 0$ otherwise. The user type indicator $z_{m,n}$ can be obtained via classification based on the electricity usage distribution $P\{\mathbf{x}_m\}$ of the m-th user and the given benchmark distributions, which will be elaborated in the next subsection. Assume now the user type indicators are known, we have the total bill gain G of the utility company as

$$G = \sum_{n=1}^{N} \sum_{m=1}^{L} z_{m,n} f_m(\mathbf{p}^m), \tag{3.1}$$

where variable N stands for the number of users and function $f_m(\mathbf{p}^m)$ stands for the bill charge for the typical m-th category/type user. For simplicity and consistency, we will refer to the m-th user type as type $\mathbf{p}^m$ for the rest of this paper.

To ease the presentation, we simplify our differentiating user service model and set $L = 2$, i.e., we only consider two types of users in this paper. For the $L > 2$ case, the differentiating user service model can be easily extended in a similar way. The total

bill gain of the utility company is

$$G = \sum_{n=1}^{N} (z_{1,n} f_1(\mathbf{p}^1) + z_{2,n} f_2(\mathbf{p}^2)), \tag{3.2}$$

To transform (3.2) into a concise form, we have

$$G = \alpha \cdot N \cdot f_1(\mathbf{p}^1) + \beta \cdot N \cdot f_2(\mathbf{p}^2), \tag{3.3}$$

where the coefficients α and β indicate the percentage of type $\mathbf{p}^1$ and type $\mathbf{p}^2$ users among the total population, respectively. As can be seen, coefficients α and β are functions of user type indicators $z_{m,n}$. Hence, α and β are dependent on the classification results based on the electricity usage distributions $P\{\mathbf{x}_m\}$ of each user and the given benchmark distributions.

Given two types of users, we accordingly model two types of pricing plans for the individual bill, referred as the flat plan and the dynamic plan, respectively, where

$$\text{flat plan: } \tilde{f}_F(\mathbf{p}^1) = c_f \sum_{i=1}^{D(S)} p_i^1 \tag{3.4}$$

and

$$\text{dynamic plan: } \tilde{f}_D(\mathbf{p}^2) = c_p \sum_{i \in \mathbf{P}} p_i^2 + c_o \sum_{j \in \mathbf{P}^c} p_j^2. \tag{3.5}$$

Vectors $\mathbf{p}^1$ and $\mathbf{p}^2$ are the associated expectations of benchmark distributions $P_1\{\mathbf{x}\}$ and $P_2\{\mathbf{x}\}$, respectively. The fixed coefficient c_f represents the pricing rate $dollar/kw \cdot h$ for the flat plan, and c_p and c_o regulate the pricing rate for the dynamic cluster where the peak usage and off-peak usage are charged, respectively. The set $\mathcal{P}$ is a set of peak-hour time mark at the scale $\mathcal{S}$ ($\mathcal{S} \neq t_3$) and $\mathcal{P}^c$ is the complement of $\mathcal{P}$ which satisfies $\mathcal{P}^c = \mathcal{S} - \mathcal{P}$.

Considering the fact that some $\mathbf{p}^2$ type users may actually prefer the flat plan or some $\mathbf{p}^1$ type users incline to accept the dynamic plan, we denote the fixed probability factor a_f as the probability of $\mathbf{p}^1$-type choosing the flat plan, and a_d as the probability

of $\mathbf{p}^1$-type choosing the dynamic plan. Also, we denote b_f as the probability of $\mathbf{p}^2$-type choosing the flat plan and b_d as the probability of $\mathbf{p}^2$-type choosing the dynamic plan. There are many other approaches to model this kind of users' reaction or preference towards the provided various pricing schemes: for example, authors in [37] model the users' reaction as binary values conditioned on some constraints; in [9], the users' preference is expressed in the functional form with tunable parameters. Since modeling users' preference is not the focus of this paper, we use the fixed-real-valued parameters instead. Hence, the complete individual bill for a $\mathbf{p}^1$-type user is

$$f_1(\mathbf{p}^1) = a_f \tilde{f}_F(\mathbf{p}^1) + a_d \tilde{f}_D(\mathbf{p}^1), \quad a_f + a_d = 1. \tag{3.6}$$

The complete individual bill for a $\mathbf{p}^2$-type user is

$$f_2(\mathbf{p}^2) = b_f \tilde{f}_F(\mathbf{p}^2) + b_d \tilde{f}_D(\mathbf{p}^2), \quad b_f + b_d = 1. \tag{3.7}$$

We also investigate the expense that the utility company costs on buying the power from the power plants. Since the electricity market charges differently at peak hours and off-peak hours, we model the expense as

$$E = \sum_{i=1}^{N} \left(a_p \sum_{j \in \mathcal{P}} x_{ij} + a_o \sum_{j \notin \mathcal{P}} x_{ij} \right). \tag{3.8}$$

The fixed coefficients a_p and a_o represent the pricing rates in the unit of $dollar/kw \cdot h$ corresponding to the peak usage and off-peak usage, respectively. The value x_{ij} indicates the electricity consumption of the i-th input user data point at time dimension j. So far, we have fully developed the expression of the net profit $\hat{G}$ as

$$\begin{aligned} \hat{G} = G - E =& \alpha \cdot N \cdot f_1(\mathbf{p}^1) + \beta \cdot N \cdot f_2(\mathbf{p}^2) \\ & - \sum_{i=1}^{N} \left(a_p \sum_{j \in \mathcal{P}} x_{ij} + a_o \sum_{j \notin \mathcal{P}} x_{ij} \right). \end{aligned} \tag{3.9}$$

As stated before in this subsection, the percentage coefficients α and β are the output results of user classification, which will be further elaborated in the next subsection.

The rest parameters $\{\mathbf{p}^1, \mathbf{p}^2\}$ and $\theta = \{N, c_f, c_p, c_o, a_f, a_d, b_f, b_d, a_p, a_o\}$ are determined by the utility company.

3.2.3 Model Analysis and The Big Data Challenge

We now look into the computation of the differentiating user service model. The expected profit comes from two ends: 1) the expected number of users belonging to each groups, and 2) within each group, the expected number of users adopting differentiating user services and the expected number of users remaining in the fixed-price services. In our model, the percentage of different groups of users, α and β, is computed at the first step as an output of classifying users. The expected total profit of a utility company is then calculated with given pricing coefficients and power cost coefficients, the two key parameters in our model. We are particularly interested in the percentage values because they can be utilized for quick estimation of the bill income without bothering to calculate each user's power consumption in the big data setting. Moreover, the percentage values serve as the feedback indicator from users. By comparing the percentage values of different years, the utility company can gather the feedback information on how the users adjust their usage behaviors so that the company may update the current pricing plans in the dynamic fashion.

To compute α and β, we need to classify users by comparing the electricity usage distribution of each user to the benchmark distribution. To simplify the notation for given two user types in total, we use binary variable z_i as the user type indicator for the i-th user instead of previously used $z_{m,n}$. Hence, the computation is calculated as $\alpha = \frac{1}{N}\sum_{i=1}^{N} z_i$ and $\beta = \frac{1}{N}\sum_{i=1}^{N}(1-z_i) = 1-\alpha$. z_i is determined via user classification as

$$z_i = \begin{cases} 1, & Dis(P\{\mathbf{x}_i\}, P_1\{\mathbf{x}_i\}) \leq Dis(P\{\mathbf{x}_i\}, P_2\{\mathbf{x}_i\}), \\ 0, & Dis(P\{\mathbf{x}_i\}, P_1\{\mathbf{x}_i\}) > Dis(P\{\mathbf{x}_i\}, P_2\{\mathbf{x}_i\}). \end{cases} \tag{3.10}$$

$P_1\{\mathbf{x}_i\}$ and $P_2\{\mathbf{x}_i\}$ represent the benchmark distributions of $\mathbf{p}^1$-type and $\mathbf{p}^2$-type, respectively. The function $Dis(\cdot)$ is the closeness measure between two distributions, and we choose the function $Dis(\cdot)$ to be the L-2 distance[1] [38] between the two given distributions. We are particularly interested in discovering the percentage for the reason that once the new pricing plans are offered to all the users, the utility company may learn the feedback of user reaction by analyzing the current percentage values and compare them to the historical ones. In this way, the percentage values provide the utility company with the guidance on adjusting pricing.

Note that a straightforward computation of (3.9) needs to evaluate each user; and for each user to compute the L_2 distance of his/her distribution and the benchmark distribution, the computation needs to access each data of the user. In Table 3.1, we show a few illustrative examples of the amount of data need to be processed, given the number of users m_u, and the number of data points m_d of numerical discrete electricity usage distribution each user has at scale $S = t_2 = \{0hour, 1hour, ..., 23hour\}$.

Table 3.1: Examples of the amount of data need to be processed in GB unit.

(m_u, m_d)	(2×10^5, 8760)	(2×10^6, 8760)	(2×10^6, 17520)
Data (GB)	14.016	140.16	280.32

We would like to remind that, as discussed in Section IV.A, such computation needs to be executed many times if it is part of an optimization where different benchmark distributions, and different price coefficients are evaluated. We thus develop a much more efficient computation approach through a novel sublinear algorithm in next section.

[1]Other distance measures can be chosen for the closeness measure $Dis(\cdot)$ as well. However, we choose L-2 distance for fast computation reason.

3.3 The Problem and Sublinear Algorithms

3.3.1 The Problem and Algorithm Sketch

Our objective is to know the percentages of $\mathbf{p}^1$ and $\mathbf{p}^2$ so that we can compute the total income of the utility company. We have shown in the previous section that the complexity is high. We also note that the complexity comes from the big data collecting complexity, not the computational complexity. To this end, we propose a novel sublinear algorithm where we substantially reduce the amount of sampled data needed in computation and obtain quality outputs. We first formally define the algorithm quality we use in this paper.

Definition 3. ***Algorithm Quality****: We measure accuracy in terms of the absolute deviation of the computed answer $\hat{a}$ from the exact answer a. We assume that such deviation is less than ϵ. In addition, this deviation does not exceed in most cases; for example, only $\delta\%$ such deviation exceeds ϵ and δ is small. More precisely, we would like to have that $Pr[|\hat{a} - a| \geq \epsilon] \leq \delta$. Here we refer to ϵ as the accuracy parameter and δ as the confidence parameter.*

We now present the sketch of our algorithm development. Our objective is that given the quality parameters ϵ, δ, we use a subset of data to compute α, β and we guarantee the results are within ϵ, δ.

In our algorithm development, we split the output quality parameters ϵ and δ into ϵ_1, δ_1, and ϵ_2, δ_2. We first develop two sub algorithms AlgoPercent() and AlgoDist(), each of which is itself a sublinear algorithm. AlgoPercent() samples a subset of users, and for each user use his/her full electricity data. It guarantees ϵ_1, δ_1. AlgoDist() applies to a single user and sample a subset of his/her electricity data. It guarantees ϵ_2, δ_2. Finally we develop an overall algorithm for distributed service model computing AlgoDSMC() that call AlgoPercent() and AlgoDist() as sub functions. AlgoDSMC()

guarantees ϵ and δ. In what follows, Section 3.3.2 develops AlgoPercent(), Section 3.3.3 develops AlgoDist(), and Section 3.3.4 develops AlgoDSMC().

3.3.2 Sublinear on Percentage

The objective is to use a small portion of users to determine the percentage of $\mathbf{p}^1$-type and $\mathbf{p}^2$-type users. Since our proposed algorithm does not require the information of all input users, we refer to this property as "sublinear on percentage". Our proposed algorithm is referred as AlgoPercent(), taking ϵ_1 and δ_1 as the parameter, and all the user data $\mathbf{X}$ as the input. ϵ_1 specifies an error bound for the output estimated $\hat{a}$, while δ_1 means indicates the confidence/probability of success that the error bound can be maintained. Instead of computing over the total N users, AlgoPercent() first sub-samples $m_1 > -\frac{\log \delta_1}{2\epsilon_1^2}$ users. The user type classification is then performed on each one of the m_1 users to obtain the percentage of $\mathbf{p}^1$ and $\mathbf{p}^2$ users, respectively. AlgoPercent() is summarized in the Algorithm 5.

Algorithm 5: *AlgoPercent*$(\mathbf{X}, \epsilon_1, \delta_1)$

Sub-sample m_1 out of N users.
Perform user classification and compute $\hat{a}$ as illustrated in (3.10).

Recall that z_i is the user type indicator defined in Section 3.2.3. We assume that z_i are independent and define $Y = \sum_{i=1}^{N} z_i$, where N is the total number of users. Assume the percentage of $\mathbf{p}^1$-user among the population is α. We have $\alpha = \frac{1}{N}E[Y]$. Since z_i are all independent, $E[z_i] = \alpha$. Let $\Lambda = \sum_{i=1}^{m_1} z_i$, where m_1 is the total number of users we sampled. Let $\overline{\Lambda} = \frac{1}{m_1}\Lambda$. The next lemma says that the expectation of the sampled set $\overline{\Lambda}$ is the same as the expectation of all set. Using these notations, we then have the following lemma associated with AlgoPercent(). It is straightforward to see that $\overline{\Lambda}$ is the unbiased estimator of α. Based on this, we show that $m_1 > -\frac{\log \delta_1}{2\epsilon_1^2}$ is the constraint that is required to maintain the ϵ_1 error bound.

Lemma 1. *Given ϵ_1, δ_1, to guarantee that we have a probability of $1-\delta_1$ success that the percentage of $\mathbf{p}^1$-type users will not deviate from the true α for more than ϵ_1, the number of users we need to sample must be at least $-\frac{\log \delta_1}{2\epsilon_1^2}$.*

Proof. By the Hoeffding Inequality, which provides an upper bound on the probability that the sum of random variables deviates from its expected value, we have

$$Pr[\overline{\Lambda} - E[\overline{\Lambda}] \leq -\epsilon_1] \leq e^{-2\epsilon_1^2 m_1}, \tag{3.11}$$

$$Pr[\overline{\Lambda} - E[\overline{\Lambda}] \geq \epsilon_1] \leq e^{-2\epsilon_1^2 m_1}, \tag{3.12}$$

and

$$Pr[|\overline{\Lambda} - E[\overline{\Lambda}]| > \epsilon_1] \leq e^{-2\epsilon_1^2 m_1}. \tag{3.13}$$

Therefore, recall that $\overline{\Lambda}$ is the unbiased estimator of α, we have

$$Pr[|\overline{\Lambda} - \alpha| > \epsilon_1] = Pr[|\overline{\Lambda} - E[\overline{\Lambda}]| > \epsilon_1] \leq e^{-2\epsilon_1^2 m_1}. \tag{3.14}$$

To make sure that $e^{-2\epsilon_1^2 m_1} < \delta_1$, we need to have $m_1 > -\frac{\log \delta_1}{2\epsilon_1^2}$. □

3.3.3 Sublinear on Distribution

In this subsection, we will introduce one existent sublinear algorithm, based on which our proposed modified sublinear method will be elaborated afterwards. Sublinear algorithms can be regarded as one branch of approximation algorithms with confidence guarantees. The term "sublinear" is traditionally interpreted as an algorithm runs in sublinear time or complexity. However, in this paper, we point out that "sublinear" can also be used to indicate the algorithm uses $o(N)$ samples in space, where N is the total number of input elements.

The objective is that for each user, we use a sublinear number of samples from its electricity data distribution to determine whether the user belongs to the category

of $\mathbf{p}^1$ or $\mathbf{p}^2$. We call this algorithm AlgoDist(). The distribution of the user behavior is denoted as $P\{x\}$ which is a 24-dimensional distribution at the scale $S = t_2$. We provide the heuristic sublinear sampling method to deal with the 24-dimensional distributions. In details, the user distribution is compared with the benchmark distribution by AlgoDist() on one dimension at a time for totally 24 times (corresponding to 24 hours per day). If the distribution passes the closeness test by the AlgoDist() for at least 12 times, it is then regarded as close to the compared benchmark distribution. As a result, the user is classified as the compared type. Since there are two defined types of users, the proposed AlgoDist() only compares the testing user distribution with the $\mathbf{p}^1$-type benchmark. If the closeness test fails, the user is automatically classified as the other type.

The distribution of one dimensional random variable can be expressed as the histogram in the discretized fashion, with the bin size denoted as δ. We denote the set $\mathcal{SP} = \{1\delta, ..., n\delta\}$ as the sample space of the distribution. We assume that the separated distribution under the investigation is discrete distribution over the n elements in the set $\mathcal{SP}$. Moreover, the distribution is assumed to be represented by a vector $\mathbf{p} = (p_1, ..., p_n)$ where p_i is the probability of sampling the i-th element in the set $\mathcal{SP}$. Given a benchmark distribution in the probability vector form as $\mathbf{q} = (q_1, ..., q_n)$, we want to test whether the distribution $\mathbf{p}$ is close enough to $\mathbf{q}$ in the L_2-distance. The traditional way is to compute the whole distribution in the L_2-distance. However, it suffers from the heavy computation which is unacceptable in the big data analysis. Inspired by [20], we propose a novel modified sublinear algorithm based on the original sublinear algorithm. The key idea of the original sublinear algorithm is that by using the sublinear sampling, we can repeatedly draw a much smaller portion of all the users. Within the smaller portion of users, each distribution of user behavior is again repeatedly sampled in a smaller portion of the entire distribution. This kind

of sublinear sampling is also applied to the benchmark distribution at the same time. Two metrics are proposed to measure the closeness between the distributions: 1) the collision probability is defined as the probability that a sample from each of $\mathbf{p}$ and $\mathbf{q}$ yields the same element and is equal to $\mathbf{p}\cdot\mathbf{q}$; 2) the self-collision of $\mathbf{p}$ and that of $\mathbf{q}$ are defined similarly as $\mathbf{p}\cdot\mathbf{p}$ and $\mathbf{q}\cdot\mathbf{q}$, respectively. The complete $DistTest(\mathbf{p}, \mathbf{q}, m_2, \epsilon_2, \delta_2)$ is a realization from the original sublinear method [20] and summarized as in Algorithm 6 (the related proof can be found in [20]). Note that in our application, the error parameter ϵ_2 serves as the classification criteria: if the L_2-distance two testing distributions are within ϵ_2, the testing user is classified as the $\mathbf{p}^1$ type; otherwise, the testing user is classified into the $\mathbf{p}^2$ type.

From [20], it is proved that the error and confidence factors of DistTest() are guaranteed by the following theorem:

Theorem 1. *Given ϵ_2, δ_2 and distributions $\mathbf{p}$ and $\mathbf{q}$, the DistTest() of testing closeness passes with the probability at least $1-\delta_2$ if $||\mathbf{p}-\mathbf{q}|| \leq \epsilon_2/2$ while it passes with the probability less than δ_2 if $||\mathbf{p}-\mathbf{q}|| > \epsilon_2$. The running time of the DistTest() is $O(\epsilon_2^{-4} log(1/\delta_2))$.*

Algorithm 6: $DistTest(\mathbf{p}, \mathbf{q}, m_2, \epsilon_2, \delta_2)$ based on L_2-distance test

for $i = 1, 2, \ldots, O(log(1/\delta_2))$ **do**
- Let F_p = a set of m_2 samples from $\mathbf{p}$.
- Let F_q = a set of m_2 samples from $\mathbf{q}$.
- Let r_p be the number of pairwise self-collisions in F_p.
- Let r_q be the number of pairwise self-collisions in F_q.
- Let Q_p = a set of m_2 samples from $\mathbf{p}$.
- Let Q_q = a set of m_2 samples from $\mathbf{q}$.
- Let s_{pq} be the number of collisions between Q_p and Q_q.
- Denote $r = \frac{2m_2}{m_2-1}(r_p + r_q)$.
- Denote $t = 2s_{pq}$.
- **if** $r - t > m_2^2\epsilon_2^2/2$ **then**
  - then reject, i.e. consider the two distributions are different.

Reject if the majority of the iterations reject, accept otherwise.

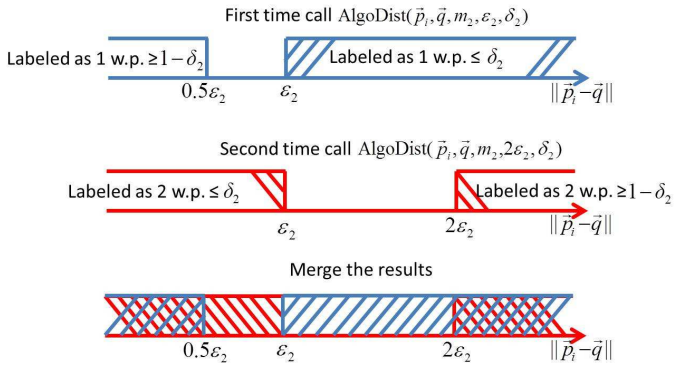

Figure 3.2: Illustration of the underlying philosophy of designing Algorithm 3.

The main drawback of directly employing the original sublinear algorithm in classification is that the confidence of the classification output remains undetermined when the L_2-distance of the testing distribution $\mathbf{p}$ and the benchmark distribution $\mathbf{q}$ is truly in the interval $[\epsilon_2/2, \epsilon_2]$ according to Theorem 1. Based on the original sublinear algorithm $DistTest(\mathbf{p}, \mathbf{q}, m_2, \epsilon_2, \delta_2)$ as indicated in the Algorithm 6, we propose a novel modified sublinear algorithm to overcome this drawback and give the complete confidence estimates. Follow the previous assumptions, let $\mathbf{p}^i$ be the power usage distribution corresponding to the i-th user, $\forall i = 1, 2, \ldots, N$. Suppose the classification by comparing a set of user electricity usage distributions $\{\mathbf{p}^i\}$ with $\mathbf{q}$ outputs two labels with the label 1 indicating user type (class) 1 whose $\mathbf{p}^i$ is close to $\mathbf{q}$; the label 2 indicating user type (class) 2 whose $\mathbf{p}^i$ is away to $\mathbf{q}$. We call the proposed modified sublinear algorithm $AlgoDist()$ and summarize it as in Algorithm 7. The underlying philosophy of designing Algorithm 7 is to call DistTest() twice but with different parameters ϵ_2 and $2\epsilon_2$, respectively, which help to get rid of the interval of undetermined confidence. Each time we call DistTest(), the classified labels of the output are retained partially. Both results are then merged with some treatment to the overlapped labels, in order to obtain a complete and consistent labeled result. The idea is illustrated in Fig. 3.2.

The algorithm accuracy of $AlgoDist()$, i.e., the classification accuracy, is given by

Algorithm 7: Modified sublinear algorithm $AlgoDist(\mathbf{p}^i, \mathbf{q}, m_2, \epsilon_2, \delta_2)$ based on $DistTest()$

for $i = 1, 2, \ldots, N$ **do**

$Step1$: Employ $AlgoDist(\mathbf{p}^i, \mathbf{q}, m_2, \epsilon_2, \delta_2)$ and obtain the classification results as $\{LabelSet1\}$.
$Step2$: Employ $AlgoDist(\mathbf{p}^i, \mathbf{q}, m_2, 2\epsilon_2, \delta_2)$ and obtain the classification results as $\{LabelSet2\}$.
$Step3$: Keep the labeled 1 in $\{LabelSet1\}$ and reject all the labeled 2.
$Step4$: Keep the labeled 2 in $\{LabelSet2\}$ and reject all the labeled 1.
$Step5$: Combine the retained labels into $\{LabelSet3\}$; If the same user is both labeled as 1 in $\{LabelSet1\}$ and labeled as 2 in $\{LabelSet2\}$, his/her label is randomly determined as either 1 or 2 in $\{LabelSet3\}$.
$Step6$: Output $\{LabelSet3\}$ as the final classification results.

the following Lemma 2:

Lemma 2. *Given ϵ_2, δ_2 and distributions $\{\mathbf{p}^i\}$ and $\mathbf{q}$, the AlgoDist() of classifying users is based on the L_2-distance criteria: label user as 1 if $||\mathbf{p}^i - \mathbf{q}|| \leq \epsilon_2$; label user as 2 if $||\mathbf{p}^i - \mathbf{q}|| > \epsilon_2$. And the classification accuracy is at least $1 - 2\delta_2$. In addition, $Pr[$labeled as 1$|$true 1$] \geq (1 - 2\delta_2)$ and $Pr[$labeled as 2$|$true 2$] \geq (1 - 2\delta_2)$.*

Lemma 2 essentially further develops the discussion in Theorem 1. In Theorem 1, the implicit underlying user group with $\epsilon_2/2 \leq ||\mathbf{p} - \mathbf{q}|| \leq \epsilon_2$ is not discussed. However, in Lemma 2, all users are taken into consideration. The proof of Lemma 2 is given as following.

Proof. Assume there are N_1 users who are truly label 1 users, i.e., whose power usage distribution satisfies $||\mathbf{p}^i - \mathbf{q}|| \leq \epsilon_2$ and N_2 users who are truly label 2 users with $||\mathbf{p}^i - \mathbf{q}|| > \epsilon_2$. The total $N = N_1 + N_2$ users' power usage distributions $\{\mathbf{p}^i\}$ are then input to $AlgoDist()$ for classification. Denote $Pr[x \in \omega_1, x \in \psi_1]$ as the joint probability that user x is truly label 1 user (i.e. $x \in \psi_1$) and has been correctly classified as label 1 (i.e. $x \in \omega_1$) by AlgoDist(). Denote its power usage distribution

as $\mathbf{p}$. According to Step 1 of Algorithm 6 and Theorem 1, we have

$$\begin{aligned} Pr[x \in \omega_1, x \in \psi_1] &= Pr[x \in \omega_1, ||\mathbf{p}-\mathbf{q}|| \leq \epsilon_2/2] \\ &+Pr[x \in \omega_1, \epsilon_2/2 < ||\mathbf{p}-\mathbf{q}|| \leq \epsilon_2], \end{aligned} \tag{3.15}$$

$$\begin{aligned} Pr[x \in \omega_1, x \in \psi_1] &\geq 1-\delta_2 \\ &+Pr[x \in \omega_1, \epsilon_2/2 < ||\mathbf{p}-\mathbf{q}|| \leq \epsilon_2], \end{aligned} \tag{3.16}$$

and

$$Pr[x \in \omega_1, x \in \psi_1] \geq 1-\delta_2. \tag{3.17}$$

Meanwhile, we have

$$Pr[x \in \omega_1, x \in \psi_2] \leq \delta_2. \tag{3.18}$$

Therefore, in the $\{LabelSet1\}$ produced by Step 3 of Algorithm 6, there are at least $N_1(1-\delta_2)$ correctly labeled users and at most $N_2\delta_2$ falsely labeled users. Likewise, considering Step 2 of Algorithm 6 and Theorem 1, we have

$$Pr[x \in \omega_1, ||\mathbf{p}-\mathbf{q}|| \geq 2\epsilon_2] \leq \delta_2, \tag{3.19}$$

$$Pr[x \in \omega_2, ||\mathbf{p}-\mathbf{q}|| \geq 2\epsilon_2] \geq 1-\delta_2, \tag{3.20}$$

$$\begin{aligned} &Pr[x \in \omega_2, ||\mathbf{p}-\mathbf{q}|| \geq 2\epsilon_2] \\ &+ Pr[x \in \omega_2, \epsilon_2 \leq ||\mathbf{p}-\mathbf{q}|| \leq 2\epsilon_2] \geq 1-\delta_2, \end{aligned} \tag{3.21}$$

$$Pr[x \in \omega_2, ||\mathbf{p}-\mathbf{q}|| \geq \epsilon_2] \geq 1-\delta_2, \tag{3.22}$$

and

$$Pr[x \in \omega_2, x \in \psi_2] \geq 1-\delta_2. \tag{3.23}$$

Meanwhile, we have

$$Pr[x \in \omega_1, x \in \psi_1] \geq 1 - \delta_2 \tag{3.24}$$

and

$$Pr[x \in \omega_2, x \in \psi_1] \leq \delta_2. \tag{3.25}$$

Therefore, in the $\{LabelSet2\}$ produced by Step 4 of Algorithm 6, there are at least $N_2(1-\delta_2)$ correctly labeled users and at most $N_1\delta_2$ falsely labeled users.

Now consider situation of overlapped labeled users in Step 5 of Algorithm 6. Given that the value of δ_2 is usually very small, i.e., high confidence of the classification results, the worst case of the overlapped labeled users are: $N_1(1-\delta_2)$ correctly labeled-as-1 users in the $\{LabelSet1\}$ completely contain $N_1\delta_2$ falsely labeled-as-2 users in the $\{LabelSet2\}$; and $N_2(1-\delta_2)$ correctly labeled-as-2 users in the $\{LabelSet2\}$ completely contain $N_2\delta_2$ falsely labeled-as-1 users in the $\{LabelSet1\}$. To express this using the set notations, we define two sets that are obtained in the $\{LabelSet1\}$ produced by Step 3 of Algorithm 6, where

$$\Omega_{11} = \{x | x \in \psi_1, x \in \omega_1\} \tag{3.26}$$

and

$$\Omega_{12} = \{x | x \in \psi_2, x \in \omega_1\}. \tag{3.27}$$

Then we have

$$\{LabelSet1\} = \Omega_{11} + \Omega_{12}, \tag{3.28}$$

$$Card(\Omega_{11}) \geq N_1(1-\delta_2), \tag{3.29}$$

and

$$Card(\Omega_{12}) \leq N_2\delta_2. \tag{3.30}$$

Define two sets that are obtained in the $\{LabelSet2\}$ produced by Step 4 of Algorithm 6 as

$$\Omega_{21} = \{x | x \in \psi_1, x \in \omega_2\} \tag{3.31}$$

and

$$\Omega_{22} = \{x | x \in \psi_2, x \in \omega_2\}. \tag{3.32}$$

Then, we have

$$\{LabelSet2\} = \Omega_{21} + \Omega_{22}, \tag{3.33}$$

$$Card(\Omega_{21}) \leq N_1 \delta_2, \tag{3.34}$$

and

$$Card(\Omega_{22}) \geq N_2(1 - \delta_2). \tag{3.35}$$

Define

$$\Delta\Omega_1 = \{x | x \in \Omega_{21}, x \in \Omega_{11}\} \tag{3.36}$$

and

$$\Delta\Omega_2 = \{x | x \in \Omega_{22}, x \in \Omega_{12}\}. \tag{3.37}$$

Define two sets that are obtained in the $\{LabelSet3\}$ produced by Step 5 of Algorithm 2 as

$$\Omega_1 = \{x | x \in \psi_1, x \in \omega_1\} \tag{3.38}$$

and

$$\Omega_2 = \{x | x \in \psi_2, x \in \omega_2\}. \tag{3.39}$$

Then, according to the Algorithm 6, we have

$$\Omega_1 = \Omega_{11} - \Omega_{11} \bigcap \Omega_{21} \tag{3.40}$$

and

$$\Omega_2 = \Omega_{22} - \Omega_{22} \bigcap \Omega_{12}. \tag{3.41}$$

The worst case happens when $\Omega_{21} \subseteq \Omega_{11}$ and $\Omega_{12} \subseteq \Omega_{22}$, and thus

$$\Omega_{11} \bigcap \Omega_{21} = \Omega_{21} \tag{3.42}$$

and

$$\Omega_{22} \bigcap \Omega_{12} = \Omega_{12}. \tag{3.43}$$

Therefore, we have

$$\begin{aligned} \min Card(\Omega_1) &= \min Card(\Omega_{11} - \Omega_{21}) \\ &= \min Card(\Omega_{11}) - \max(\Omega_{21}) \\ &= N_1(1-\delta_2) - N_1\delta_2 = N_1(1-2\delta_2). \end{aligned} \tag{3.44}$$

Similarly, we have

$$\min Card(\Omega_2) = N_2(1-2\delta_2). \tag{3.45}$$

Hence, we have

$$Pr[\text{labeled as } 1, \text{true } 1] \geq (1-2\delta_2) \tag{3.46}$$

and

$$Pr[\text{labeled as } 2, \text{true } 2] \geq (1-2\delta_2). \tag{3.47}$$

The worst case of correctly labeled users in the final result $\{LabelSet3\}$ is

$$\begin{aligned} &\min Card(\Omega_1) + \min Card(\Omega_2) \\ &= N_1(1-2\delta_2) + N_2(1-2\delta_2). \end{aligned} \tag{3.48}$$

In the worst case, the accuracy is

$$\begin{aligned} &\frac{\min Card(\Omega_1) + \min Card(\Omega_2)}{Card(\{LabelSet3\})} \\ &= \frac{(N_1 + N_2)(1 - 2\delta_2)}{N} = 1 - 2\delta_2. \end{aligned} \tag{3.49}$$

So the classification accuracy is at least $1 - 2\delta_2$. □

3.3.4 The Overall Algorithm

In this subsection, we derive the overall algorithm quality ϵ, δ of AlgoDSMC() and its sufficient condition. Since the overall objective is to estimate α, it is straightforward to see that the overall algorithm quality is $\epsilon = \epsilon_1$ and $\delta = \delta_1$. Given the parameters, $\epsilon_1, \delta_1, \epsilon_2, \delta_2$, for the sub-algorithms, ϵ_1 and δ_1 are passed to the AlgoPercent(ϵ_1,δ_1). Within the AlgoPercent(ϵ_1,δ_1), the small number of users, m_1 is sampled from the entire input users. For each one of the m_1 users, the AlgoDist($P\{\mathbf{x}_i\}$, $\mathbf{p}^1$, m_2, ϵ_2,δ_2) is called where $P\{\mathbf{x}\}$ corresponds to the distribution of the testing one user out of the m_1 users.

In AlgoPercent(ϵ_1,δ_1) with given error and confidence parameters, we compute that the sub-sampling number of users needs to be at least $m_1 = -\frac{\log \delta}{2\epsilon_1^2}$, under the assumption that AlgoDist($P\{\mathbf{x}_i\}$, $\mathbf{p}^1$, m_2, ϵ_2,δ_2) gives 100% correct classifications. However, given the error and confidence parameters ϵ_2, δ_2, the AlgoDist() again utilizes the sublinear algorithm and may misclassify a $\mathbf{p}^1$ user into $\mathbf{p}^2$-type. This means that AlgoDist() will not give 100% correct classifications, which nullifies the previous assumption made when analyzing AlgoPercent(ϵ_1,δ_1) due to the cascading relationship between AlgoPercent() and AlgoDist(). Then we have to modify the parameter settings of AlgoPercent() by taking the property of AlgoDist() into consideration. As a result, the subsample number m_1 has to be modified in order to maintain $\epsilon = \epsilon_1, \delta = \delta_1$.

We derive the constraint for the subsample number m, in order to make the probability of the failure of the overall algorithm, $Pr[|\hat{a} - a^*| \geq \epsilon]$, bounded by δ.

Theorem 2. *Given ϵ, δ for the overall algorithm quality, ϵ_2, δ_2 for AlgoDist() and suppose δ_2 is small enough such that $\epsilon > 6\delta_2$, to guarantee that we have a probability of $1-\delta$ success that the percentage of $\mathbf{p}^1$-type users will not deviate from the true α for more than ϵ, i.e., $Pr[|\hat{\alpha} - \alpha| \geq \epsilon] = Pr[|\overline{\Lambda} - \alpha| \geq \epsilon] \leq \delta$, the number of users we need to sample must be at least $m \geq -\frac{\log \delta}{2(\epsilon - 6\delta_2)^2}$.*

Proof. Denote $p_{11} = Pr[x \in \omega_1 | x \in \psi_1]$ as the probability that user x is truly label 1 user and has been classified as label 1. Similarly, define $p_{21} = Pr[x \in \omega_1 | x \in \psi_2]$ as the probability that user x is truly label 2 user and has been classified as label 1. By Lemma 2, we have $1 - p_{11} \leq 2\delta_2$ and $p_{21} = 1 - p_{22} \leq 2\delta_2$. Follow the discussion in Section 3.3.2 and Lemma 2, we have

$$E[\overline{\Lambda}] = \frac{1}{N}(\sum_{i=1}^{N_1} p_{11} \cdot 1 + \sum_{j=1}^{N_2} p_{21} \cdot 1) = \alpha p_{11} + (1-\alpha)p_{21}. \tag{3.50}$$

Therefore,

$$|\overline{\Lambda} - E[\overline{\Lambda}]| = |\overline{\Lambda} - \alpha + \alpha(1 - p_{11} + p_{21}) - p_{21}|, \tag{3.51}$$

$$|\overline{\Lambda} - E[\overline{\Lambda}]| \geq |\overline{\Lambda} - \alpha| - |\alpha(1 - p_{11} + p_{21})| - |p_{21}|, \tag{3.52}$$

$$|\overline{\Lambda} - E[\overline{\Lambda}]| \geq |\overline{\Lambda} - \alpha| - \alpha|1 - p_{11}| - \alpha|p_{21}| - |p_{21}|, \tag{3.53}$$

$$|\overline{\Lambda} - E[\overline{\Lambda}]| \geq |\overline{\Lambda} - \alpha| - \alpha \cdot 2\delta_2 - \alpha \cdot 2\delta_2 - 2\delta_2, \tag{3.54}$$

and

$$|\overline{\Lambda} - E[\overline{\Lambda}]| \geq |\overline{\Lambda} - \alpha| - 6\delta_2. \tag{3.55}$$

Hence, given $|\overline{\Lambda} - \alpha| \geq \epsilon$, it is sufficient to say that $|\overline{\Lambda} - E[\overline{\Lambda}]| \geq \epsilon - 6\delta_2$, which implies

$$Pr[|\overline{\Lambda} - \alpha| \geq \epsilon] \leq Pr[|\overline{\Lambda} - E[\overline{\Lambda}]| \geq \epsilon - 6\delta_2]. \tag{3.56}$$

Denote $\tilde{\epsilon} =\geq \epsilon - 6\delta_2 > 0$, by the Hoeffding Inequality, we have

$$Pr[|\overline{\Lambda} - E[\overline{\Lambda}]| \geq \tilde{\epsilon}] \leq e^{-2\tilde{\epsilon}^2 m}. \tag{3.57}$$

To ensure that $e^{-2\tilde{\epsilon}^2 m} \leq \delta$, we need to have

$$m \geq -\frac{\log \delta}{2\tilde{\epsilon}^2}. \tag{3.58}$$

That is

$$m \geq -\frac{\log \delta}{2(\epsilon - 6\delta_2)^2}. \tag{3.59}$$

If this holds and using $\overline{\Lambda}$ as the estimator of α, then

$$Pr[|\hat{\alpha} - \alpha| \geq \epsilon] = Pr[|\overline{\Lambda} - \alpha| \geq \epsilon] \leq \delta. \tag{3.60}$$

□

3.4 Evaluation

In this section, we evaluate our proposed differentiating user service model and associated algorithms. Due to the nondisclosure agreement, all results are computed from the simulated data that are generated according to the real data analysis. The proposed algorithms can be directly applied to real data without modification. Note that our sublinear algorithm has already provided a theoretical bound. We thus primarily investigate the relationship among the number of data we need to process, the errors and confidence interval. More specifically we evaluate:

1. The relations among the number of sub-samples m, m_2 and the error ϵ and confidence δ;

2. The proposed algorithm accurately estimates of α value within an acceptable error bound;

3. The differentiating user services model performs better than the traditional single rate fixed-price approach; In addition, analyzing the impact brought by the factor α and total user number N;

4. The proposed sublinear algorithm significantly reduces the computation load.

We evaluate the proposed differentiating user services model and the sublinear algorithm at scale $S = t_2 = \{0hour, 1hour, ..., 23hour\}$. The evaluation data set is simulated based on the real data analysis and similar to the generation procedure of the benchmark distributions. According to the data trace study in Section 3.1, we simulate the dataset based on Gaussian distributions. Specifically, we first generate the benchmark distribution of $\mathbf{p}^1$-type user similarly as the Household 1 plotted in Fig. 3.1 in Section 3.1. The total number of users is set to $N = 100,000$. α is varying from 0.1 to 0.8. The usage distribution of one $\mathbf{p}^1$ user is generated in this way: 1) each dimension of $\mathbf{p}^1$ (recall that $\mathbf{p}^1$ is the corresponding expectation of the benchmark distribution of $\mathbf{p}^1$-type, as defined in Section 3.2.2) is added with a random Gaussian noise drawn from $Gauss(0, 0.5)$, resulting in a noisy vector $\tilde{p}^1$; 2) generate a sequence of random variables $\tilde{k}_i^1, i = 1, ..., 24$ that conforms to the Gaussian distribution $Gauss(\tilde{p}_i^1, 0.1)$ respectively; 3) form the vector $\tilde{y} = (\tilde{k}_1^1, ..., \tilde{k}_{24}^1)$ as one instance of daily usage that belongs to the $\mathbf{p}^1$-type user; 4) repeat 2) and 3) for 365 times and obtain the set of vectors $\{\tilde{y}\}$ that represents the usage distribution of the $\mathbf{p}^1$ user. Finally, we repeat the procedure from 1) to 4) for $\alpha \cdot N$ times and obtain the data points of $\mathbf{p}^1$-type users. The $\mathbf{p}^2$-type users are simulated in the same fashion except that the Gaussian noise conforms to $Gauss(0, 0.1)$ and their user number is $\beta \cdot N$.

For Objective 1), we use the data set generated with $N = 100,000$ and $\alpha = 0.7$. The parameters $\epsilon_1 = 0.05$, $\delta_1 = 0.05$, $\delta_2 = 0.005$ and $\epsilon_2 = 0.5$ are fixed. Then we vary m_2 for $m = 2000, 3000, 10000, 15000$. We define the estimation error as the absolute

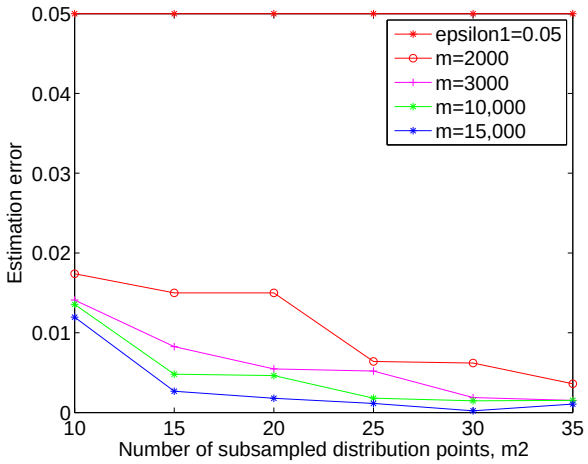

Figure 3.3: Estimation errors $|\hat{\alpha} - \alpha|$ vs. sub-sampling number m_2 from the entire distribution.

value of the difference between the estimated $\hat{\alpha}$ and the true α. By inputting the data set and the parameters into our proposed sublinear algorithm, we obtain the results shown in Fig. 3.3. As can be seen, as the sub-sample number m_2 grows larger, the estimation error generally becomes smaller, which is consistent with the spirit of the sublinear methods: the more we sub-sample, the more precise results we will obtain. However, the speed of ameliorating the result to become closer to the true value is much slower than the increase speed of the required subsamples: if we want to further reduce the error that is already small, we need to pay much more price, i.e., giving a much larger step of increments for m_2.

For Objective 2), the total number of users is set to $N = 100,000$. α is varying from 0.1 to 0.8. We fix the parameters for the AlgoPercent() and AlgoDist() as: $\epsilon_1 = 0.05$, $\epsilon_2 = 0.5$, $\delta_1 = 0.05$, $\delta_2 = 0.005$, $m = 50,000$, $m_2 = 60$. Notice that under this parameter setting, the overall algorithm quality is $\epsilon = \epsilon_1 = 0.05$ and $\delta = \delta_1 = 0.05$. The data sets are then input into our overall algorithm. The estimated results are shown in Fig. 3.4. As can be seen, our algorithm estimates the α values precisely within the error bounds throughout all the simulated values. The maximum absolute error percentage is 1.42%, and the minimum absolute error percentage is 0.10%. And

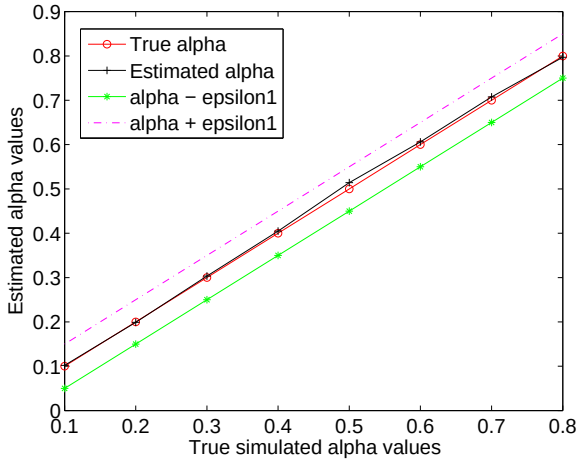

Figure 3.4: Estimated α values vs. simulated true α values.

the subsamples used are only 50% of the total users.

For Objective 3), we first investigate how the proportion of different types of users impact on the net profit. Using the data sets generated in the objective 1) with different α values, we set the parameters of the proposed differentiating user services model as: $N = 100,000$, $c_f = 1$, $c_p = 3$, $c_0 = 0.8$. $a_f = 0.1$, $a_d = 0.9$, $b_f = 0.8$, $b_d = 0.2$, $a_p = 2$ and $a_o = 0.5$. These parameters are chosen with a reference to the real electricity markets and bills. According to electricity usage patterns during the past years in Houston, the peak hour set is $\mathbf{P} = \{10hour, 11hour, 12hour, 18hour, 19hour, 20hour, 21hour\}$. The estimated α values are input into our differentiating user services model. To compare our model with the other two traditional pricing plans, we choose $c_f = 1$ to be the pricing factor that applies to all the users uniformly for the pricing plan referred as the fixed price service. We also choose $c_p = 3$ and $c_0 = 0.8$ for the plan that charges users uniformly but with varying price at peak and off-peak hours, referred as the differentiated charge. The experiment results are shown in Fig. 3.5, where 3 mechanisms of pricing are explained: (1) charging uniformly according to usage regardless of time/hour when the usage happens (referred as fixed price service); (2) charging according to usage but with different rates at peak vs. off-peak

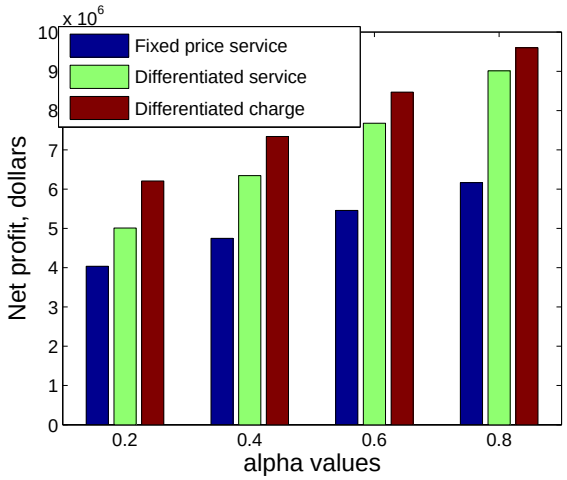

Figure 3.5: Net profits from the differentiating user services vs. net profits from non-differentiating user services with varying proportion of user types.

hours (referred as differentiated charge); (3) charging according to different types of user (referred as our proposed differentiating user service). It can be seen that the proposed model favors over the $\mathbf{p}^1$-type users who generally use more power especially during the peak hours and bring more profits. The differentiated charge obtains the highest profits because it forces all the users to pay much more money at the peak hours, which is usually not suitable for the $\mathbf{p}^2$-type users. Fig. 3.5 indicates that: fixed price service is inefficient in the sense of static pricing; differentiated charge is inefficient in the sense of over charging (certain types of user would not accept this kind of service); differentiating user service is a reasonable pricing compared with the rest two. The profits and percentage analysis can be further utilized in the future to evolve into an advanced dynamic model with dynamic pricing factors, from which the reactions of users can be revealed.

For Objective 4), we compare the computation load for direct computation with the proposed sublinear algorithm under some different parameter settings specified in the objective 1). Taking the repeating procedure in AlgoDist() into account, the overall amount of data needed to be processed by the proposed sublinear algorithm is expressed as $2 \times m \times m_2 \times 24 \times log(\frac{1}{\delta_2}) \times 8$, while the direct computation of entire

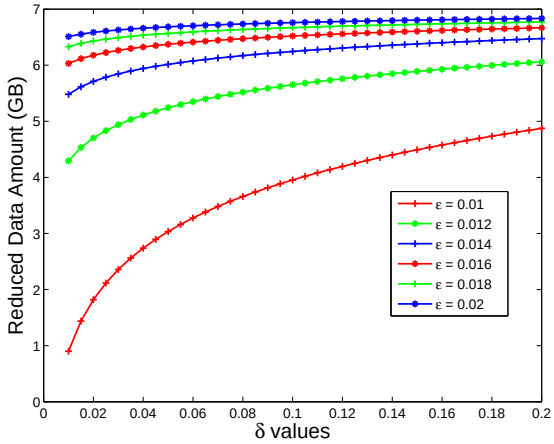

Figure 3.6: Reduced data amount (GB) vs. overall confidence parameter δ.

data needs to process $N \times 365 \times 24 \times 8$. In the objective 3), $N = 100,000$, $\delta_2 = 0.005$, $m_2 = 10, 15, 20, 25, 30, 35$, and m varies as $m = 2000, 3000, 10000, 15000$. We render the numerical computation load in Table 3.2, where the second column corresponds to the direct computation and the 3rd to 6th columns correspond to the proposed algorithm with some specific parameter settings of (m, m_2). As can be seen from the table, the proposed sublinear algorithm greatly reduces the computation load at the price of acceptable estimation error on the percentage α as indicated in the objective 3). We also investigate the numerical computation load by considering the minimum amount of data required in the computation of proposed sublinear methods with varying parameters and given fixed input data. Suppose we have $N = 100,000$ users and therefore 7.008 GB data as input, we vary the parameters ϵ_1, δ_1, δ_2 and fix $\epsilon_2 = 0.5$ (ϵ_2 is problem-oriented and data-driven. In our application, it is set according to the distance between the two benchmark distributions) to see what is the minimum amount of data involved in the computation of proposed sublinear methods in order to guarantee performance. We render the numerical computation load in Table 3.3 as a function of parameters ϵ_1, δ_1 and δ_2. As can be seen in Table 3.3, the smaller error bounds and larger confidence impose more computation load for our

algorithms. Moreover, the error bound parameter ϵ_1, which is highly related to the subsampled number of users m, is the major factor that influences the computation load, compared with the rest two factors. Another interesting observation can be found in the first row of Table 3.3 that even if the parameters change to have higher confidence, i.e., smaller δ_2, the required data to be processed is less. This is so because the smaller δ_2 indicates higher confidence of the success of the classification algorithm AlgoDist(). As a consequence, we no longer need to sample as many users as before to guarantee the overall algorithm, as indicated in Theorem 2. To further demonstrate the computational efficiency of the proposed sublinear methods, we also investigate the amount of data (GB) that is reduced by employing our algorithms instead of direct computation on the original 7.008 GB input data. First, we fix the parameters $\epsilon_2 = 0.5$ and $\delta_2 = 0.001$, and plot the reduced data in GB unit by varying the confidence parameter δ of overall algorithm (note that $\delta = \delta_1$ as discussed before). The results are shown as in Fig. 3.6 with different overall error bound ϵ. We then fix the internal parameters $\epsilon_2 = 0.5$ and $\delta_2 = 0.001$, and plot the reduced data in GB unit by varying the error bound parameter ϵ of overall algorithm (note that $\epsilon = \epsilon_1$ as discussed before). The results are shown as in Fig. 3.7 with different overall confidence ϵ. As can be seen from both Fig. 3.6 and Fig. 3.7, if the confidence or the error bound is relaxed, our proposed method can reduce larger amount of data involved in computation. Another discover is that the error bound will influence the reduced data more gently while the impact of the confidence parameter saturates fast as δ increases. This provides a guidance in practice that if we aim to reduce the computation load, there is no need to relax the confidence too much.

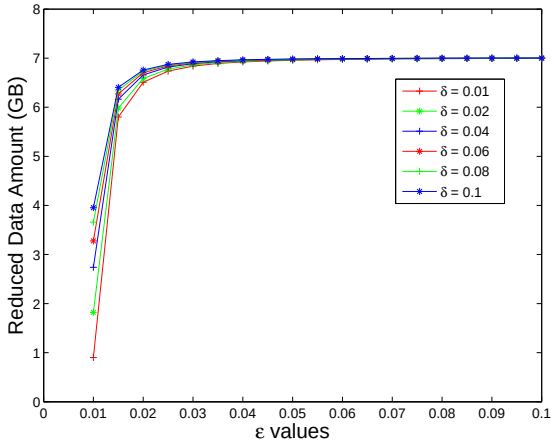

Figure 3.7: Reduced data amount (DB) vs. overall error bound parameter ϵ.

Table 3.2: Comparison of the amount of data (GB unit) needed to be processed between direct computation and proposed sublinear algorithm with different parameter settings of (m, m_2).

m, m_2	$m = N = 10^5$	$(2000, 10)$	$(2000, 20)$	$(2000, 35)$
Data (GB)	7.008	0.041	0.081	0.142
m, m_2	$(3000, 10)$	$(3000, 20)$	$(3000, 30)$	$(2000, 35)$
Data (GB)	0.061	0.122	0.183	0.214
m, m_2	$(10^4, 10)$	$(10^4, 20)$	$(10^4, 30)$	$(10^4, 35)$
Data (GB)	0.204	0.407	0.610	0.712

3.5 Summary

In this chapter, we investigated a differentiating user service model for electricity usage. The model is based on an analysis of a real smart metering data trace where we observed that there exists various usage patterns among the power energy customers. One key problem of a differentiating user service model is that the model computation faces a huge amount of data. There is a large number of customers, and for each customer, his/her electricity usage pattern is represented by long period and multi-dimensional data. As a result, the complexity for a differentiate service model is not in the sense of computation, but in big data. We developed a novel sublinear

Table 3.3: Minimum amount of data (GB unit) involved in the computation of proposed sublinear methods as a function of (ϵ_1, δ_1, δ_2)

$\epsilon_1, \delta_1, \delta_2$	(0.05,0.05,0.006)	$(0.05, 0.05, 0.001)$
Data (GB)	0.2402	0.0328
$\epsilon_1, \delta_1, \delta_2$	$(0.05, 0.005, 0.001)$	$(0.01, 0.05, 0.001)$
Data (GB)	0.0581	3.9732

algorithm where we use a sublinear amount of data and we guarantee a small error bounds and a given confidence. We demonstrated by both theoretical proofs and trace-driven evaluations that our algorithm can effectively reduce the amount of data to be processed to a range that is reasonable for the state-of-the-art computing capability.

Chapter 4

Tensor Voting Framework with Its Applications in Human Mobile Trace Inference

As an example to illustrate tensor methods, we address the tracking problem of inferring human mobility trace under the circumstance that the recorded location information exhibits missing and noisy data. Trace inference provides key location information of objects such as personal devices in wireless networks, and can serve as one of the important networking topics. Based on the tensor voting theory, we propose an efficient tensor voting algorithm and a specified implementation scheme. The model is constructed based on the geometric connections between the input signals and encodes the structure information in the tensor matrix. Thus, the computation is carried out in the form of matrix, which reduces the computation load. The proposed method is applied to real human mobility trace. After trace completion by tensor voting, we develop a systematic framework of feature extraction to analyze the traces. The fractal analysis is incorporated in the feature extraction method to characterize scale-independent features of human mobile trace. The results show that our proposed approach effectively recovers human mobility trace from the incomplete data input and provides comprehensive analysis of the trace. Our key contributions are:

1. Giving the detailed derivation of constructing the normal space based on the eigenvalue problem;

2. Implementing the tensor voting algorithm efficiently in the sparse sense;

3. Applying valid evaluation criteria to quantify the performance of the proposed

tensor voting algorithm;

4. Applying fractal analysis to characterize the trace features for data mining tasks.

This chapter is based on our previous work [39] and organized as follows: In Section 4.1, the literature survey is presented to provide a broad view of tensor methods as well as the fractal analysis utilized in this chapter. In Section 4.2, the mathematical model of tensor voting is introduced to encode the data and perform the structure inferring procedure. The inference algorithm is given in Section 4.3. In Section 4.4, trace analysis is performed based on the efficient feature extraction including fractal analysis. Simulation results are presented in Section 4.5. In Section 4.6, we draw conclusions and summarize this chapter.

4.1 Background

Some tracking problems can be essentially transformed as the image processing problems where the tensor voting technique has been widely utilized. Guy elaborated tensor voting theory with insightful analysis in his Ph.D. thesis [40]. The theory is then ameliorated by other researchers in the past decade [41]. In tracking applications, as the topic of this chapter, tensor voting systematically infers hidden or incomplete structures, for instance, gaps and broken parts in the trace curve. Tensor voting is also referred to as perceptual grouping [42] emphasize the contribution of the Gestalt principles on which the theory is based. In brief, the Gestalt principles state that the presence of each input token (site, pixel, signal, etc.) implies a hypothesis that the structure passes through it. For example, considering the $2D$ imaging process of a chair, if one pixel has recorded the chair signal, it is highly likely that some of its neighboring pixels should have captured the same structure/object signals. In other words, it is human nature to configure simple elements into the perception of complex

structures. In [43] and [44], the object signals observed by fixed cameras are represented using spatiotemporal features which facilitates the application of tensor voting theory. The advantage of applying tensor voting brings several geometric properties including smooth continuous trajectories and bounding boxes with minimum registration error. Although there are extensive research efforts dedicated to tensor voting study in the image processing field [45], little work involving the tensor voting theory has been done in the communication realm according to the best of our knowledge. Moreover, very few research handles with the missing data problem in the tracking context.

Besides tensor voting, tensor theories have been widely applied to emerging hot topics such as big data. Tensor decomposition has been developed to address various data mining tasks as an extension of principal component analysis (PCA) in higher dimensions. In [46], a scalable and distributed version of the Tucker model, MR-T, is implemented using the Hadoop MapReduce framework. Authors in [47] propose a new constrained tensor factorization framework, building upon the Alternating Direction method of Multipliers (ADMoM). Work in [48] permeates benefits from rank minimization to scalable imputation of missing data, via tracking low-dimensional subspaces and unraveling latent structure from incomplete streaming data. Work in [49] addresses the problems of computing decompositions of full tensors by using compressed sensing (CS) methods working on incomplete tensors, i.e., tensors with only a few known elements.

Another topic in tracking problems is trajectory analysis [50], which has wide application scenarios including vehicle traffic management, vessel classification by satellites images and so forth. In [51] a unifying framework is constructed to mine trajectory patterns of various temporal tightness. The proposed framework consists of two phases: initial pattern discovery and granularity adjustment. Authors in [52] employ

the fractal analysis of a fin whale's trajectory tracked by the satellite. The implemented fractal analysis provides the scale-independent measurement to summarize interactions between an organism and its ecosystem and depends on the heterogeneity of the whale's environment and the whale's ability to perceive it. In [53], several real-world human mobility traces are employed to analyze network robustness in the time domain. Authors in [54] propose a novel integrated framework for multiple human trajectory detection, learning and analysis in complicated environments. In [55], a new approach for abnormal loitering detection using trajectory analysis is described and Inverse Perspective Mapping (IPM) is presented to resolve distortion of trajectory direction.

4.2 Tensor Voting Model

In this section, we illustrate the tensor voting framework. In Section 4.2.1, we present the way to encode the normal space with tensor representation. In Section 4.2.2, the fundamental stick tensor voting is addressed as the basis for inferring geometric structure via the encoded normal space. In Section 4.2.3, the initialization procedure for tensor voting is introduced under the circumstance that no prior structure information is known. In Section 4.2.4, the inference method based on tensor decomposition is explained.

In order to infer the hidden or missing structures, we need to model the structures mathematically first. Generally, the structure types in the form of $2D$ images can be classified into two categories: curves and regions. Curves are modeled as the structures that have a 1-d normal space, which is referred to as a stick. Regions are modeled as the structures that have a 2-d normal space, referred to as a ball. Normal space represents the structure types well, but it is required to know how salient the structures are in order to adequately model the structure. Hence, the parameter

defined as saliency associated with each structure type is employed to indicate the size of structure. Both the normal space and saliency information are encoded in a tensor via specific calculation that will be described later.

After developing the mathematical models, we can further explain the hints obtained from the Gestalt principles [56]: (1) a token "communicates" its structure information to its surrounding tokens in a certain way with respect to its normal space, i.e., the surrounding tokens under its influence are supposed to have the same kind of normal space; (2) in the real world, a token may contain a combination of information of both structure types. For instance, a token that actually belongs to a curve has a dominant saliency in the 1-d normal space while it probably has minor saliency in the 2-d normal space.

4.2.1 Encode Normal Space with Tensor

In math, a normal space is an N-by-N matrix for objects in N-dimension, denoted by N_d. Consider a d-dimension normal space in the N-dimension world, which is spanned by the first d out of N orthonormal basis vectors $\mathbf{e}_k$, $k = 1, 2, ..., N$, we have the normal space matrix expressed as

$$\mathbf{N}_d = \sum_{k=1}^{d} \mathbf{e}_k \mathbf{e}_k^T. \tag{4.1}$$

The projection of any vector $\mathbf{v}$ into this normal space is

$$\mathbf{v}_n = \mathbf{N}_d \mathbf{v}. \tag{4.2}$$

It can be easily proved that $\mathbf{v}_n$ is the projection of $\mathbf{v}$ into the normal space by showing that: (1) $\mathbf{v}_n$ can be linearly expressed by the basis vectors $\mathbf{e}_k$, $k = 1, 2, ..., N$; (2) dot product $\langle \mathbf{v}_n - \mathbf{v}, \mathbf{v}_n \rangle = 0$.

The exact tensor matrix $\mathbf{V}$ that encodes both normal spaces and their saliency is assumed to be known. From the previous discussion, we know that $\mathbf{V}$ is a symmetric,

positive semi-definite N-by-N matrix. Suppose its eigenvalues are ordered as $\lambda_1 \geq ... \geq \lambda_N \geq 0$ with corresponding eigenvectors $\mathbf{e}_1, ..., \mathbf{e}_N$. By the knowledge of linear algebra, we know that the eigenvectors of $\mathbf{V}$ are orthogonal with each other (for the eigenvectors that belong to 0 eigenvalue, we can generate those eigenvectors in a way that they meet this requirement). Then if we normalize these eigenvectors into unit vectors, we have a set of orthonormal basis $\hat{\mathbf{e}}_1, ..., \hat{\mathbf{e}}_N$. Furthermore, we have the following derivations as

$$\mathbf{V}\hat{\mathbf{e}}_d = \lambda_d \hat{\mathbf{e}}_d, \tag{4.3}$$

$$\mathbf{V}\hat{\mathbf{e}}_d\hat{\mathbf{e}}_d^T = \lambda_d \hat{\mathbf{e}}_d\hat{\mathbf{e}}_d^T, \tag{4.4}$$

$$\mathbf{V}\sum_{d=1}^{N}\hat{\mathbf{e}}_d\hat{\mathbf{e}}_d^T = \sum_{d=1}^{N}\lambda_d \hat{\mathbf{e}}_d\hat{\mathbf{e}}_d^T, \tag{4.5}$$

$$\mathbf{V} = \sum_{d=1}^{N}\lambda_d \hat{\mathbf{e}}_d\hat{\mathbf{e}}_d^T = \sum_{d=1}^{N-1}(\lambda_d - \lambda_{d+1})\sum_{k=1}^{d}\hat{\mathbf{e}}_k\hat{\mathbf{e}}_k^T + \lambda_N\sum_{k=1}^{N}\hat{\mathbf{e}}_k\hat{\mathbf{e}}_k^T, \tag{4.6}$$

and

$$\mathbf{V} = \sum_{d=1}^{N-1}(\lambda_d - \lambda_{d+1})N_d + \lambda_N N_N. \tag{4.7}$$

From (4.7), we define the saliency s_d in the straight forward fashion: $s_d = \lambda_d - \lambda_{d+1}$, if $d < N$; $s_d = \lambda_N$, if $d = N$. Thus, substituting the saliency into (4.7), we have

$$\mathbf{V} = \sum_{d=1}^{N} s_d N_d. \tag{4.8}$$

4.2.2 Inferring Structure

To simplify the illustration procedure, we will assume that we already know the structure types and saliencies for now. We will discuss how to obtain this information later.

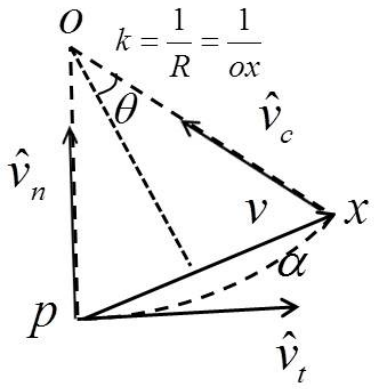

Figure 4.1: Illustration of the fundamental stick vote

We start with the simplest case that is the fundamental unit stick vote when the normal space is 1-d. Consider a voter point p (a token that passes its structure information to others) on a curve. Its normal is a known unit vector $\hat{\mathbf{v}}_n$. We want to know how it influences its neighboring votee point x (a token that receives structure information from voters). Based on the previous discussion, we assume that p and x share the same structure type when we consider p is influencing x. To approximate the path of the same structure type that passes through p and x, we take the arc of the osculating circle centered at o passing through p and x as the most likely smooth path [57]. Figure 4.1 shows the geometric relationship between p and x. $\hat{\mathbf{v}}_t$ is the known tangent vector at p, and $\mathbf{v}$ is the vector from p to x. $\hat{\mathbf{v}}_c$ is the normal vector at x that we want to calculate. θ is the vote angle between $\mathbf{v}$ and $\hat{\mathbf{v}}_t$. We take the influence as 0 when θ is larger than $\pi/4$ for the reason that two points that have an angle larger than 90 degrees between their normals are least likely to influence each other. a is the arc length between p and x. k is the curvature of the osculating circle, which is the reciprocal of the radius $R = ox$. To calculate θ and $\hat{\mathbf{v}}_c$, we have

$$\theta = \arcsin \mathbf{v}^T \hat{\mathbf{v}}_n \tag{4.9}$$

and

$$\hat{\mathbf{v}}_c = \hat{\mathbf{v}}_n \cos 2\theta - \hat{\mathbf{v}}_t \sin 2\theta. \tag{4.10}$$

We also add a decay profile to the tensor to model the decaying influence of the information going through the structure. Therefore, the complete unit stick vote (tensor) that encodes the normal space information received by the votee is

$$\mathbf{V}^p = DF(a, k, \sigma)\hat{\mathbf{v}}_c\hat{\mathbf{v}}_c^T. \tag{4.11}$$

$DF(a, k, \sigma)$ is the decay profile that takes a, k and σ as parameters. σ is the free parameter set by the user to control the scale of voting. The decay profile can be given empirically or based on a traditional choice as

$$DF(a, k, \sigma) = e^{-(\frac{a^2+ck^2}{\sigma^2})}, \tag{4.12}$$

where the parameters can be derived by the geometry as

$$c = \frac{-16log(0.1)(\sigma - 1)}{\pi^2}, \tag{4.13}$$

$$a = \frac{\theta||\mathbf{v}||}{\sin\theta}, \tag{4.14}$$

and

$$a = \frac{2\sin\theta}{||\mathbf{v}||}. \tag{4.15}$$

Usually, a voter's stick vote is not a unit vector. If so, the unit tensor expressed in (4.11) is multiplied with the corresponding saliency of the 1-d normal space of the voter.

Likewise, when inferring structures for the 2-d normal space, we attempt to find the same normal space at the votee. In that case, 2 basis vectors are required for the 2-d normal space. The procedure is equivalent to: (1) find out the basis vectors that span voter's normal space; (2) vote or project these basis vectors, respectively, in the way the fundamental stick vote does to the votee. (3) reconstruct the complete normal space information at the votee by combining the newly generated normal space information, i.e., adding the tensor matrices created by the stick vote, respectively.

To find out the basis vectors of the voter's normal space, Mordohai proposes a method [57] that projects the voter-to-votee vector into voter's normal space and then computes the orthonormal basis vectors for the voter's normal space based on the projection and Gram-Schmidt orthogonalization procedure. This method significantly reduces the computation when calculating the basis vectors of the high dimensional normal space.

Suppose the voter's normal space is known and encoded as the tensor matrix $\mathbf{N}_d^p$, where p represents the voter point and d represents the dimension of its normal space. For any fixed votee point x that receives p's vote, the voter-to-votee vector $\mathbf{v}$ is known. Then the projected vector is

$$\mathbf{v}_n = N_d^p \mathbf{v}. \tag{4.16}$$

Thus, the tangent vector $\mathbf{v}_n$ is computed by

$$\mathbf{v}_t = (I - N_d^p)\mathbf{v} = \mathbf{v} - \mathbf{v}_n, \tag{4.17}$$

where I is the identity matrix of dimension N-by-N. Based on (4.17), these two vectors are then normalized as $\hat{\mathbf{v}}_n$ and $\hat{\mathbf{v}}_t$. The first constructed basis vector for the normal space is selected as $\hat{\mathbf{v}}_{n,1} = \hat{\mathbf{v}}_n$. Next, the Gram-Schmidt procedure is employed to construct the rest $d-1$ orthonormal basis vectors $\hat{\mathbf{v}}_{n,i},\ i = 2, 3, ..., d$. As a consequence, each $\hat{\mathbf{v}}_{n,i}$ is considered as the fundamental stick vote and voted to the votee as the voting procedure described above. Each stick vote results in a tensor matrix $\mathbf{V}_i^p,\ i = 1, 2, ..., d$ for the d-dimension normal space. Finally, these tensors are summed up into one matrix that represents the complete information for the d-dimension normal space at votee x. The proposed method of generating the basis set facilitates the computation because for $i = 2, 3, ..., d$, the basis vector $\hat{\mathbf{v}}_i$ is orthogonal to $\mathbf{v}$, which means the vote angle $\theta = 0$. Hence, (4.10) is simplified during the computation.

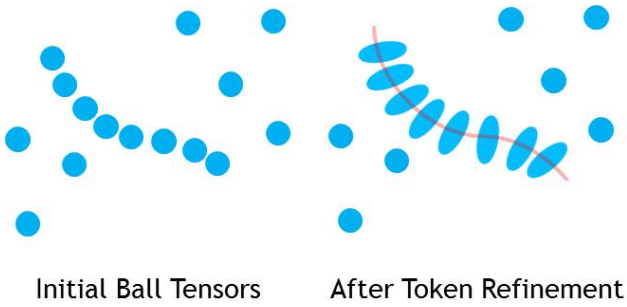

Figure 4.2: Illustration of the token refinement procedure: each point is initialized with a ball tensor; points nearby with each other form stick tensors while points far away from others remain ball tensors.

4.2.3 Token Refinement

So far, the discussions are based on the presumption that the normal space and saliencies information are known at a given voter site. Nevertheless, in most cases, it is impossible to obtain this kind of prior knowledge. Thus, the proper initialization called token refinement, which estimates the prior information, is needed.

Figure 4.2 illustrates the token refinement procedure in a $2D$ space. In the token refinement procedure, each input token is initialized with a unit ball tensor indicating neither direction preference nor prior saliency information. The input tokens are then considered one by one as the voter and voted to its neighboring input tokens. In the end, all the tensors received by each input token are summed up and stored as the known normal space and saliency information. If a cluster of tokens actually belong to the same curve in the real world, and then the way they influence each other using their initial ball tensors will put major emphasis on the stick tensor, namely the 1-d normal space in a $2D$ world. As can be seen in Figure 4.2, the tokens along the curve influence each other during token refinement, resulting in elongating their tensors to become stick tensors in the $2D$ space, while the tokens that sparsely spread out the space receive little information from others, causing the existing ball tensors to remain..

4.2.4 Token Decomposition

After token refinement, the tensor voting procedure can be completed by the method described in Section 4.2.2. The result in the $2D$ image case is that each pixel is associated with a 2x2 matrix $\mathbf{T}$. The ultimate objective is to decide which structure type the candidate pixel should belong to. Hence, the tensor matrix $\mathbf{T}$ needs to be decomposed by (4.7) to extract the saliencies for the 2 structure types. In $2D$ case, it becomes

$$\mathbf{T} = \lambda_1 \hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T = (\lambda_1 - \lambda_2)\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T + \lambda_2(\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2 \hat{\mathbf{e}}_2^T). \tag{4.18}$$

If $\lambda_1 - \lambda_2 > \lambda_2 > 0$, the stick saliency is the dominant one, which indicates the certainty of one normal orientation. Therefore, the token is inferred as the part of a curve, with its estimated normal being $\hat{\mathbf{e}}_1$. If $\lambda_1 \approx \lambda_2 > 0$, the dominant component is the ball saliency, which means there is no preference of orientation. Thus, the token is estimated as the part of a region or a junction where two or more curves intersect with multiple orientations present simultaneously. Note that, if both the saliency values are very small, the candidate token is likely an outlier. This makes tensor voting capable of filtering noise.

4.3 Inference Algorithm

Based on the previous discussion, there are multiple choices with respect to how to implement the tensor voting technique. One feasible way to implement the tensor voting technique is to compute the so called voting field for each token; this method is referred to as the per-voter scheme. After token refinement procedure is complete, the per-voter algorithm examines every input token as a voter and computes the set of votes it casts to all its neighbors. That set of votes is referred as the voting field. The algorithm then integrates all the voting fields and performs tensor decomposition

at each site. In [58], tensor voting is implemented based on the per-voter scheme. In addition, the algorithm is combined with the steerable filter theory [59] to rewrite the tensor voting operation as a linear combination of complex-valued convolutions, which significantly reduces the computation load.

In this chapter, a straightforward implementation method of tensor voting technique referred to as the per-votee scheme is proposed to infer the human mobility trace encoded in the location data with some recordings missing. The per-voter scheme calculates one vote from point to point at one time. In order to reduce the computation, we also implement tensor voting in a sparse sense. When examining one site as a votee, we only consider the influence it received from the neighboring pixels $\{C_{ee}\}$ within the radius of approximately 3σ as reported in [40]. Furthermore, we define the sparse voting region, gather all the tensors received by each votee only in that region and decompose the result tensor matrix $\mathbf{M}_{ee}$ to determine its actual structure type. Finally, we make the voting procedure iterative so that it is able to fill the large gaps. The implementation scheme is summarized in the Algorithm 1. In the initialization stage, each voter matrix is initialized as the identity matrix $\mathbf{M}_{er}$, representing the ball tensor while each votee matrix $\mathbf{M}_{ee}$ is initialized by the zero matrix. During the voting procedure, the voter matrix $\mathbf{M}_{er}$ is firstly decomposed into two normal vectors $\{\mathbf{V}_{n1}, \mathbf{V}_{n2}\}$ which are then voted as the fundamental stick tensor to the votee. As reported in [40], each voter decomposition using (4.8) is computed in $O(N^3)$ time where N is the dimension of input data. And each stick vote indicated in (4.11) is computed in $O(N^3)$ time.

4.4 Trace Analysis

In this section, we carry on to analyze the trace patterns in order to discover representative behaviors or characteristics of the objects. The underlying philosophy of

Algorithm 8: Tensor voting based on the per-votee scheme

input $2D$ image with the incomplete trace points set $\{P_{ij}\}$, set scale of voting σ.
for $r = 1, 2, \ldots,$ *number of iterations* **do**
  initialize voter matrix $\mathbf{M}_{er}$ and votee matrix $\mathbf{M}_{ee}$.
  for $i, j = 1, 2, \ldots,$ *indexes of trace points* **do**
    for *each point p lies in the neighbohood of* $\{P_{ij}\}$ **do**
      if $p \in \{P_{ij}\}$ **then**
        calculate DF and θ by the coordinates of the voting pair $\{p, P_{ij}\}$;
        decompose p's $\mathbf{M}_{er}$ into 2 normal vectors $\{\mathbf{V}_{n1}, \mathbf{V}_{n2}\}$;
        multiple $\{\mathbf{V}_{n1}, \mathbf{V}_{n2}\}$ with DF and project them respectively to P_{ij};
        collect all vectors received at P_{ij} and convert them into tensor.
      $\mathbf{M}_{ee}$ of P_{ij} = sum up all tensors that P_{ij} has received;
    update the $\mathbf{M}_{er}$ for $\{P_{ij}\}$ by $\mathbf{M}_{er} = \mathbf{M}_{ee}$;
  find out a new set of votees $\{C_{ee}\}$ that are defined by the sparse region of $\{P_{ij}\}$;
  set the votee matrix $\mathbf{M}_{ee}$ for $\{C_{ee}\}$ as 2-by-2 zero matrix;
  for *each point in* $\{C_{ee}\}$ **do**
    repeat the similar voting procedure above using the updated voter matrices.
  for *each pixel x of the image* **do**
    if $x \in \{C_{ee}\}$ **then**
      decompose the $\mathbf{M}_{ee}$ of x;
      classify the pixel x according to its saliencies;
  Skeletonize the updated trace points.

trace analysis is that we believe different objects' traces reflect some hidden features that can be differentiated. The objective of mining traces can vary in different contexts. For example, in our study, due to the different road layouts of distinct cities, it is highly likely that the corresponding collected human mobile traces carry different hidden features. Hence, by analyzing the properly extracted features of human mobile traces, different corresponding cities can be inferred as well as the road layout styles. Another application of trace analysis is the traffic control problem. Based on the trace analysis of each vehicle, it is possible to identify outlier vehicles which may cause damage and loss to the traffic.

In order to perform trace analysis, proper hidden features of the complete traces have to be extracted first. Note that feature extraction depends on the specific application. Here we propose three features to be used to characterize a human mobile trace: normalized trajectory mean, Fourier descriptor and fractal dimension. These features can be categorized into two classes: spatial features (the first two) and scale-invariant feature (the fractal feature). It is worth to point out that the fractal feature provides suitable descriptions of inherent nature of the object data recorded by the GPS due to the scale-independent property brought by the fractal analysis. Hence, regardless of any resolution of the GPS measurements or any unit (meters, kilometers, etc.) that is taken to represent the trace, the fractal feature stays consistent.

To ease the illustration of feature extraction procedure, we assume the trace data are in $2D$. Let the set of inferred trace points be $\{T\}$, and we assume there are n points that continuously form the inferred trace, each with two coordinates in the $2D$ space: $(T_x^i, T_y^i), i = 1, 2, ..., n$. The normalized trajectory mean is defined as

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}(T_x^i, T_y^i) - (min(T_x^i), min(T_y^i)). \tag{4.19}$$

The normalized trajectory mean is to roughly indicate the cover range or off-set range of the trace. Sometimes the trace might form a loop and therefore, the Fourier de-

scriptor is needed to characterize the trace in this case. For the coordinates of each trace point, there is a corresponding complex value formed as

$$z^{i-1} = T_x^i + j \cdot T_y^i, \tag{4.20}$$

where j is the imaginary unit. Thus, we can obtain a sequence of $\{z^i\}$ and the discrete Fourier transform of $\{z^i\}$ is

$$a(u) = \frac{1}{n} \sum_{i=0}^{n-1} z^i e^{-j2\pi ui/n}, u = 0, 1, ..., n-1. \tag{4.21}$$

The complex coefficient $a(u)$ is referred to as the Fourier descriptor and its absolute value indicates the magnitude of corresponding frequency component. Note that the Fourier descriptor is independent of how the first trace point is chosen. High frequency components present rapidly varying details in the trace while low frequency components determine the shape of trace at the coarse level. Denote the absolute value of low frequency components and high frequency components as a_L and a_H, respectively. We then define R_f to be the ratio of low vs high frequency components as

$$R_f = a_L/a_H, \tag{4.22}$$

which serves as the Fourier descriptor feature for the further trace analysis.

In order to include scale-independent feature, we briefly introduce the fractal dimension to the feature extraction procedure. The underlying philosophy of fractal dimension is that self-similarity and repeating patterns exist in nature objects. A fractal is a shape made of parts similar to the whole in some way. Usually, a fractal is a set of objects that has a fractal dimension that exceeds its topological dimension. The fractal dimension indicates the effect of space occupation by the complex shape. There are many ways to formulate fractal dimension, such as the Hausdorff dimension, the Richardson Law [52] and so forth. Here we adopt the Richardson Law to define fractal dimension as

$$L(\epsilon) \propto \epsilon^{D-D_f}, \tag{4.23}$$

where D is the topological dimension of the trace and $D = 1$. D_f is the fractal dimension. ϵ is the imaginary unit "rule" used to measure the trace. It can be a line segment for measuring curves, or a square/circle for measuring planar objects, or a cube/ball for measuring solid objects. $L(\epsilon)$ is the number of "rulers" used to continuously cover the entire trace. By taking the log of (4.23), we have

$$\ln(L(\epsilon)) = (1 - D_f) \ln \epsilon. \tag{4.24}$$

Therefore, we can vary the unit "ruler" ϵ and compute the corresponding counts $(L(\epsilon))$ of the trace. The fractal dimension D_f is then calculated by linear regression via (4.24).

So far we have obtained all the features to characterize a trace: the normalized trajectory mean $\mathbf{m}$, the Fourier descriptor feature R_f, and the fractal dimension D_f. These extracted features of traces are then input into our classification model to infer the different corresponding cities. We choose the logistic regression model for our trace classification/city inference task. Since the classification model is not the key point in this chapter, we refer readers to the work in [60] for further information regarding the classification model used here.

4.5 Example and Analysis

The performance of the proposed tensor voting algorithm is verified through extensive experiments on the human mobility data collected in New York city by the GPS [61]. Each of the 39 collected traces is firstly converted into $2D$ binary images of the dimension 314×351. The $2D$ images are then randomly sampled to have missing or broken segments of approximately 7-pixel length on average. Finally, the proposed tensor voting algorithm is applied to the images with missing parts and outputs the inferred complete trace of the human object. Due to the small size of missing gaps,

the iteration number is set fixed to 1 in this chapter for simplicity. For the cases where there are large missing gaps, the iteration number should be set larger when the adjacent iterations return sufficiently similar results. The experiments are conducted extensively to each image with various values of σ, which is the only free parameter that controls the scale of voting, ranging from 1 to 50. One instance of the complete traces and corresponding sampled trace with missing parts are shown in Fig. 4.3 and 4.4, respectively. Fig. 4.5 and 4.6 are the inferred traces by setting the scale of voting parameter $\sigma = 1$ and $\sigma = 2$, respectively. As can be seen in these two figures, the gaps are not fully connected using too small voting scale because the structure elements are unable to influent further points. While in the situation that σ is too large, the gaps can be connected however there are many over-detected segments of the traces, which make the performance inefficient. One particularly interesting point is that the algorithm even recovers the missing arcs appropriately. During the whole procedure of tensor voting algorithm, there is no user effort required to be input to indicate which part of the trace is missing and should be inferred. The inference is completed fully automatically once the only free parameter σ is fixed. This is one aspect of the strength of tensor voting method. The key of implementing tensor voting algorithm is the determination of data-driven parameters: the number of iterations and the scale of voting σ. Empirically, larger values of iterations and σ will make the algorithm capable of filling larger missing gaps, while the smaller values of them are more suitable for the traces of plenty of narrow missing gaps.

There are massive efforts contributed to the performance measure for the tracking problems [62], [63], [64]. To validate the inferred traces and quantify the performance of the proposed algorithm, the similarity metrics proposed in [65] is adopted due to its practical merits. Denote the complete trace points of the ground truth as the set $\{S\}$ while the set of inferred trace points being regarded as set $\{T\}$ as defined in

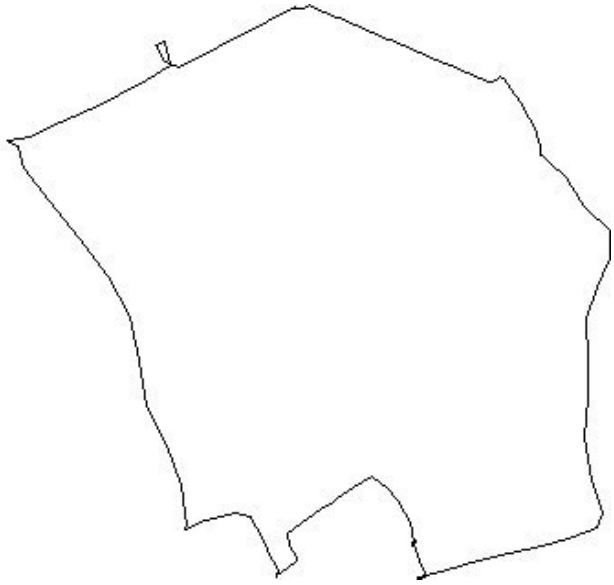

Figure 4.3: One instance of complete human mobility trace converted from the GPS data.

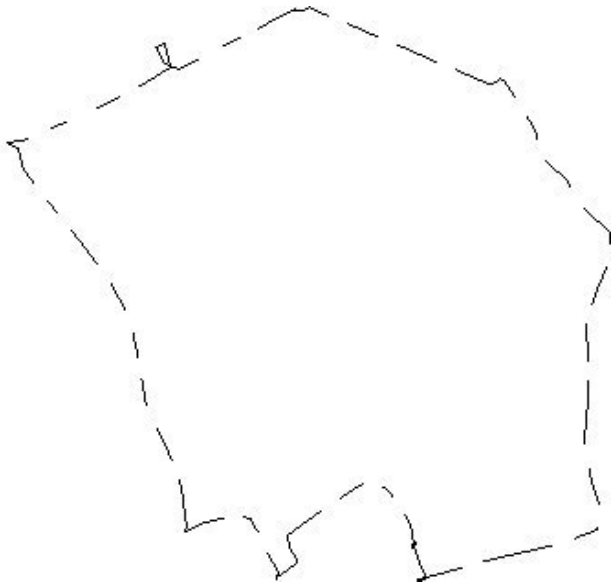

Figure 4.4: Corresponding sampled human mobility trace with missing segments.

Figure 4.5: Inferred result employing tensor voting with voting scale σ=1.

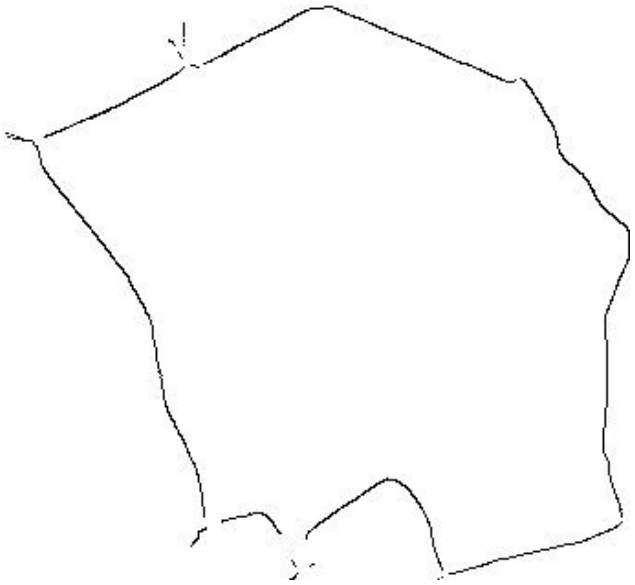

Figure 4.6: Inferred result employing tensor voting with voting scale σ=2.

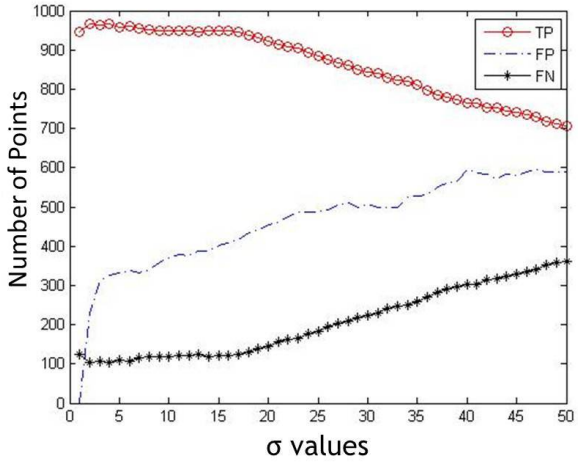

Figure 4.7: Corresponding True Positive, False Positive and False Negative curves.

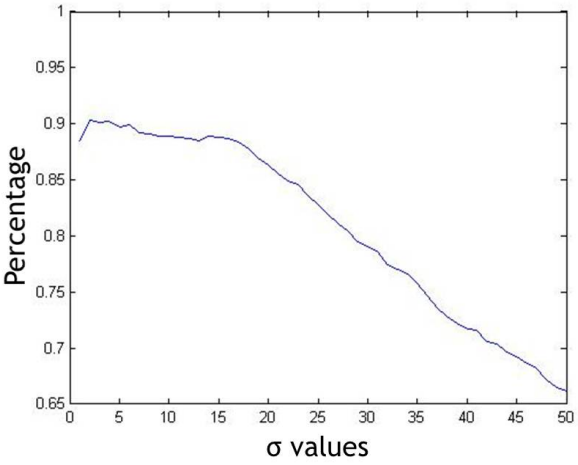

Figure 4.8: Corresponding True Positive ratio curve.

previous sections. The true positive (TP) of the inference algorithm is defined as the cardinality of the set $\{p\}_i$ satisfying

$$\{p\}_i = \{p | p \in \{T\}, p \in \{S\}\}. \tag{4.25}$$

The false positive (FP) and false negative (FN) of the inference algorithm are defined in the similar fashion as

$$FP = Card(\{p\}_j); \{p\}_j = \{p | p \notin \{T\}, p \in \{S\}\} \tag{4.26}$$

and

$$FN = Card(\{p\}_k); \{p\}_k = \{p | p \in \{T\}, p \notin \{S\}\}. \tag{4.27}$$

Since the trace points only occupy a small portion of the entire image, it would be irrational to take the true negative (TN) into account when evaluating the accuracy of the algorithm. In addition, due to the specific setting of the tracking problem, most interests have been focused on the value of TP. Furthermore, in order to compare the performance on various traces, we compute the TP ratio by dividing the TP values by the cardinality of the set $\{S\}$. The quantified performance of the tensor voting algorithm with various σ values is shown in Fig. 4.7 and 4.8. The importance is placed on the TP curve due to the specific settings of the tracking problem. And the optimal TP ratio achieves 90.36% at $\sigma = 2$ while the numbers of FP and FN are relatively small. Considering the average lenght of the missing segments in our experiments is 7-pixel, this result is consistent with the theory represent in [40], which states that the points within the circle centered at a voter of a radius of approximate 3σ would effectively receive the votes. When the σ is chosen too small, the points are under little influence from others, leading to the failure of inferring large gaps. On the other hand, as the σ grows larger, the points far away with each other interfere so much that the reconstructed results are full of noise. The small fluctuations present locally in

the four curves are consequence of different lengths of missing segments that appear in the input images.

To show the advantage of proposed sparse tensor voting algorithm, we also construct the control group or so called victim image using other approach to accomplish the inference task. In this study, we choose to directly connect each pair of the end points of the missing segments, i.e. the control approach assumes the missing parts are all linear segments, which will fail where the true trace exhibits curves. The performance comparison between the proposed tensor voting (TV) algorithm and control method is shown in the Table 4.1, where the experiments are both conducted on the same trace. It can be seen from the Table 4.1 that the proposed algorithm outperforms the control one since the missing segments are not all linear segments. In the situation where most miss segments are curves, the performance difference between the two methods will be more significant.

Table 4.1: Performance comparison between the proposed tensor voting algorithm and victim method.

Method	TP	TP ratio	FP	FN
TV	966	90.36%	230	103
Victim	887	82.97%	293	182

To evaluate the proposed methods for trace analysis, we first present the illustrations of extracting fractal dimensions for each trace. The traces are converted into binary images which are then computed using (4.24) with the varying $\epsilon = 2, 3, 4, 6, 8, 12, 16, 32, 64$ in the unit of pixels. In Fig. 4.9, we illustrate the fractal dimension calculated by linear regression for the same trace used in Fig. 4.3-4.8. As can be seen in Fig. 4.9, the fractal dimension of the investigating trace is larger than 1, indicating that the shape of the trace is "more complex" than a straight line in 2D. For the Fourier descriptor feature R_f, we take the average of first 30% portion of the frequency components in (4.21) as the low frequency components a_L and the average

of last 30% portion of the frequency components as the high frequency components a_H. The feature R_f is then computed according to (4.22). As indicated in Section 4.4, we feed the logistic regression model with the extracted features: the normalized trajectory mean $\mathbf{m}$, the Fourier descriptor feature R_f, and the fractal dimension D_f to classify the traces into different cities. We perform the trace analysis on two data sets with 39 traces from New York (label 0) and 41 traces from Orlando (label 1), respectively. The training and testing data sets are formed by 4-fold cross validation, each time with 60 traces in the training set and 20 traces in the testing set where the traces are divided randomly. The precision of regression results are obtained by averaging each cross validation result, and are shown as the confusion matrices in Table 4.2 where the numerical values refer to the counts of correcly/incorrectly classified traces. As can be seen in Table 4.2, the precision is 80% in the training set and 85% in the testing set, which demonstrates the effectiveness of our trace analysis methods.

Table 4.2: Trace analysis via logistic regression.

Training set	Estimated label 0	Estimated label 1
True label 0	26	4
True label 1	8	22
Testing set	**Estimated label 0**	**Estimated label 1**
True label 0	7	2
True label 1	1	10

4.6 Summary

This chapter provides an effective approach to infer the human mobility trace as the key object frequently utilized in networking, given that the observed location data exist missing parts. Traditional tracking techniques seldom deal with the problem of missing data setting. By employing the tensor voting technique as one of the artificial intelligence methods, the trace inference is done in an automatic fashion. The

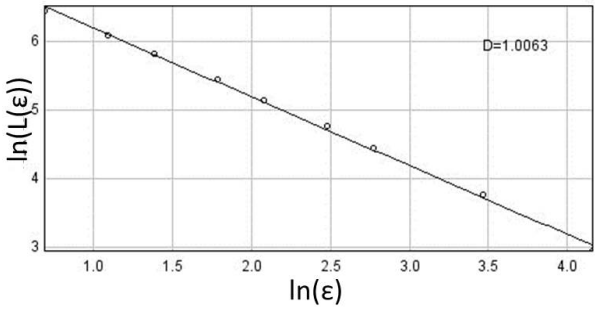

Figure 4.9: Fractal dimension computation via $\ln(L(\epsilon))$ vs $\ln \epsilon$ plot, corresponding to the same trace used in Fig. 4.3.

only free parameter that requires the user input is the voting scale. One advantage of the proposed algorithm is that there is no requirement of user instructions for identifying which part to inferred. The algorithm discovers the missing positions and accomplishes inference. The sparse per-votee implementation scheme significantly reduce the computation, making the algorithm suitable for potentially large-scale data set and online analysis. By employing the similarity metrics between the inferred trace and the ground truth, our algorithm shows its power in recover the human mobility trace accurately. The tensor grouping method can be applied to estimate local dimension in manifold learning or function approximation tasks. The fractal analysis can be designed for image compression, denoising and channel estimation. Future work may also involve with: (1) modifying decay profile to better fit specific problems; (2) combining tensor representations with other information, for instance, combining the second order tensor voting with the first order information as the polarity coefficient to detect end point, which prevents over-detecting points.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this dissertation, we provide conduct a theoretical research in smart grid systems and wireless communications networks with emphases on probabilistic clustering analysis, pricing scheme design, sublinear sampling, tensor voting theory and trajectory pattern recognition.

First, we conduct the investigation on clustering analysis in smart grid systems. Our approach involves several key technical ideas as follows: First, an efficient pragmatic feature extraction by multi-scale analysis of 24 dimensional daily load profiles, beginning at coarse grain levels. Second, a novel iterative probabilistic clustering algorithm based on the modified Mahalanobis distances between load profiles in feature space and log-likelihood optimization. This algorithm converges rather fast to well separated clusters and can be scalable for big data sets. Third, a new strong separation index to characterize the separability of any pair of clusters by precise statistical analysis. Fourth, a hierarchical approach to automatic clustering by starting with the generation of a small number of clusters at coarse grain scales and iteratively attempting sub-cluster splitting at finer scales. Finally, the simulation results on our large benchmark data set of smart meter data indicate that our approach is easily implementable at the computational level and performs quite convincingly on our benchmark data set.

Then, we investigate the pricing scheme based on differentiating users in smart grid networks and develop a sublinear sampling method accordingly. The model is based on an analysis of a real smart metering data trace where we observe that there

exists various usage patterns among the power energy customers. One key problem of a differentiating user service model is that the model computation faces a huge amount of data. There is a large number of customers, and for each customer, his/her electricity usage pattern is represented by long period and multi-dimensional data. We develop a novel sublinear algorithm where we use a sublinear amount of data and we guarantee a small error bounds and a given confidence. We demonstrat by both theoretical proofs and trace-driven evaluations that our algorithm can effectively reduce the amount of data to be processed to a range that is reasonable for the state-of-the-art computing capability.

Third, we research on the trajectory inference and pattern analysis problems involved in the communication networks. By employing the tensor voting technique as one of the artificial intelligence methods, the trace inference is done in an automatic fashion. The only free parameter that requires the user input is the voting scale. One advantage of the proposed algorithm is that there is no requirement of user instructions for identifying which part to inferred. The algorithm discovers the missing positions and accomplishes inference. The sparse per-votee implementation scheme significantly reduce the computation, making the algorithm suitable for potentially large-scale data set and online analysis. By employing the similarity metrics between the inferred trace and the ground truth, our algorithm shows its power in recover the human mobility trace accurately. The tensor grouping method can be applied to estimate local dimension in manifold learning or function approximation tasks. The fractal analysis can be designed for image compression, denoising and channel estimation. Future work may also involve with: (1) modifying decay profile to better fit specific problems; (2) combining tensor representations with other information, for instance, combining the second order tensor voting with the first order information as the polarity coefficient to detect end point, which prevents over-detecting points.

5.2 Future Work

Even thought the clustering analysis, sublinear sampling and tensor voting theory have been investigated in this dissertation, there are still many problems remaining unsolved in there domains.

5.2.1 Future Improvements on the Theory Side

One improvement direction for the proposed probabilistic clustering work is to determine the number of clusters at the beginning of the algorithm. Although in the proposed clustering work, the clustering strategy is treated hierarchically such that the final and optimal number of clusters can be obtained at the end of the proposed clustering framework, there are certain cases where the globally optimal number of clusters is in need. This prior information is critical for many clustering techniques such as K-means. There are some existent methods dealing with this problem. One feasible approach is introduced by Tibshirani [32]. In his pioneering work, he developed the so-called gap statistic to estimate the optimal number of clusters globally. The brief underlying philosophy of gap statistic is to measure the within-cluster scatter as a function of number of clusters. In principle, if the number of clusters is set to 1, the resulting within-cluster scatter will be the largest. However, if the number of clusters is chosen to be equal to the number of data points, the resulting within-cluster scatter will be 0, i.e., each data point itself forms a cluster. Hence, the within-cluster scatter is decreasing over the number of cluster. Obviously, setting large values for the number of clusters is over-demanded while small number of clusters will not guarantee the compactness of the clustered structures. Work in [32] proposes a trade-off scheme. Roughly speaking, the optimal number of clusters should be determined at the value where the decreasing speed of within-cluster scatter starts to slow down. This method is efficient in solving the problem of optimal number of clusters. However, further in-

vestigation is needed to employ it in the proposed probabilistic clustering approach in this dissertation. Moreover, the gap statistic method requires the evaluation of every possible number of clusters, which requires certain computation load. Further investigation may involve how to make this procedure more efficient. Another important topic regarding the proposed clustering technique is the initialization of cluster centroids. Traditional treatment for centroids initialization is to randomly select some data points out of the entire data set. This initialization strategy is, however, unable to be proved as the optimal solution since the random initialization can lead to local optimum. Generally speaking, it is better that the initialization chooses the data points that are more representative. Hence, further research should focus on how to select representative cluster centroids and avoid choosing the ones that are more like outliers. Moreover, the convergence of many existent clustering algorithms has not been proved theoretically even though most of the algorithms achieve convergence in various real world applications.

Regarding the pricing scheme, the objective function of the proposed model is a linear prototype which may not fit into the complexity of other application cases. Future research can focus on how to design a nonlinear objective function while guaranteeing the solution of the optimization problem by proposing some efficient numerical algorithms. Considering the proposed sublinear sampling method, one future improvement may focus on how to theoretically extend the theory to allow for high dimensional data since the original sublinear sampling is designed for one dimensional data input.

As for the proposed tensor voting methods, one key concern is how to reduce the computation load under the situation that the input data dimension is high. If that is the case, the proposed tensor voting methods will cope with eigen-decomposition of high dimensional matrix, which is cumbersome in efficiency. In addition, the decay function needs further research to make it more general for wide applications. Cur-

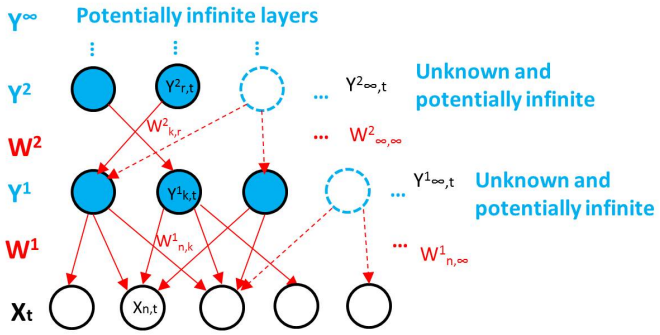

Figure 5.1: Possible generative model of infinite layers for the future work.

rently, the choice of the exponential family as the decay function is quite empirical. Further validation and adjustment is needed in order to make the model more popular.

5.2.2 Future Improvements on the Application Side

The proposed clustering methods are tested using smart meter data. In the future, we wish to test the methods on more data sets from other domains such as classification of user equipment in wireless communications. The developed sublinear sampling method can be adapted in the scenarios in computer science such as the comparison of two string streams. In data science, tensors are utilized to model the data cube in which the inherent property of data is encoded and can be revealed through tensor decomposition. To broad the research field, the future research may focus on the tensor methods with applications in data compression and representation. With the development of big data topics, tensor methods are gaining increasing popularity nowadays. However, in networking research field, tensor methods are seldom discussed. We wish to introduce tensor methods into networking field with new merit and good performance.

5.2.3 Future Improvements on Alternative Models

Statistical models have been applied to the classification and prediction problems in machine learning and data analysis. Some statistical methods make the hypothesis of mathematical models that are controlled by certain parameters to fit the latent structure of observed data. The observed data are assumed to be generated by complex structures which have hierarchical layers and hidden causes. One key challenge faced by modeling the data structure in this way is thus the determination of the numbers of layers and hidden variables. However, it is sometimes impractical and challenging to choose any fixed number for the model structure when making the hypothesis. Therefore, we need flexible non-parametric models that make fewer assumptions and are capable of an unlimited amount of latent structures as illustration in the Fig. 5.1. Hierarchical nonparametric Bayesian model assumes unspecified number of latent variables and produces rich kinds of probabilistic structures by constructing cascading layers. Hence, it is considered to be a powerful technique to cope with the challenge. To continue the research in machine learning and boost our work in smart grid systems, we will further explore several emerging techniques such as the non-parametric Bayesian inference and deep learning to investigate their applicability and efficiency under the smart meter context. As is known, non-parametric Bayesian has the advantage of high automation and needs few assumptions while its hierarchical layers, however, cannot reach as many as possible. Deep learning architecture is capable of providing deep hierarchical layers but needs some pre-determined parameters. One possible direction is to combine these two techniques to achieve higher automation and accuracy.

References

[1] H. Farhangi, "The Path of the Smart Grid," *IEEE Power and Energy Magazine*, vol. 8, no. 1, pp. 18–28, January 2010.

[2] F. Li, W. Qiao, H. Sun, H. Wan, J. Wang, Y. Xia, Z. Xu, and P. Zhang, "Smart Transmission Grid: Vision and Framework," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 168–177, September 2010.

[3] D. Huang, H. Zareipour, W. Rosehart, and N. Amjady, "Data Mining for Electricity Price Classification and the Application to Demand-side Management," *IEEE Transactions on Smart Grid*, vol. 3, no. 2, pp. 808–817, June 2012.

[4] L. Qian, Y. Zhang, J. Huang, and Y. Wu, "Demand Response Management via Real-time Electricity Price Control in Smart Grids," *IEEE Journal on Selected Areas in Communications*, vol. 3, no. 7, pp. 1268–1280, July 2013.

[5] S. Amin and B. Wollenberg, "Toward a Smart Grid: Power Delivery for the 21st Century," *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34–41, September 2005.

[6] S. Chen, K. Xu, Z. Li, F. Yin, and H. Wang, "A Privacy-aware Communication Ccheme in Advanced Metering Infrastructure (AMI) Systems," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, April 2013, pp. 1860–1863.

[7] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, "Smart Grid Technologies: Communication Technologies and Standards," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, November 2011.

[8] V. Ford and A. Siraj, "Clustering of Smart Meter Data for Disaggregation," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, TX, December 2013, pp. 507–510.

[9] C. Joe-Wong, S. Sen, S. Ha, and M. Chiang, "Optimized Day-ahead Pricing for Smart Grids with Device-specific Scheduling Flexibility," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1075–1085, July 2012.

[10] M. Roozbehani, M. Dahleh, and S. Mitter, "Dynamic Pricing and Stabilization of Supply and Demand in Modern Electric Power Grids," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Gaithersburg, MD, October 2010.

[11] Q. Wang, M. Liu, and R. Jain, "Dynamic Pricing of Power in Smart-grid Networks," in *IEEE 51st Annual Conference on Decision and Control (CDC)*, Maui, HI, December 2012, pp. 1099–1104.

[12] C. Tsai, A. Pelov, M. Chiang, C. Yang, and T. Hong, "A Brief Introduction to Classification for Smart Grid," in *IEEE International Conference on SMC*, Manchester, UK, October 2013, pp. 2905–2909.

[13] L. Du, J. Restrepo, Y. Yang, R. Harley, and T. Habetler, "Nonintrusive, Self-organizing, and Probabilistic Classification and Identification of Plugged-in Electric Loads," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1371–1380, September 2013.

[14] C. A. Ramon Granell and D. Wallom, "Impacts of Raw Data Temporal Resolution Using Selected Clustering Methods on Residential Electricity Load Profiles," *IEEE Transactions on Power Systems*, vol. 30, no. 99, pp. 1–8, December 2014.

[15] Y. B. Maria Halkidi and M. Vazirgiannis, "On Clustering Validation Techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2-3, pp. 107–145, December 2001.

[16] T. Hong, J. Wilson, and J. Xie, "Long Term Probabilistic Load Forecasting and Normalization With Hourly Information," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 456–462, January 2014.

[17] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler, "Sublinear-time Approximation of Euclidean Minimum Spanning Tree," in *Proc. 14th Symp. Discrete Algorithms*, January 2003, pp. 813–822.

[18] D. Wang, Y. Long, and F. Ergun, "A Layered Architecture for Delay Sensitive Sensor Networks," in *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, September 2005, pp. 24–34.

[19] F. Ergun, H. Jowhari, and M. Sağlam, "Periodicity in Streams," in *Proceedings of the 13th International Conference on Approximation, and 14 the International Conference on Randomization, and Combinatorial Optimization: Algorithms and Techniques*, Berlin, Heidelberg, April 2010, pp. 545–559.

[20] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White, "Testing that Distributions Are Close," in *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA, November 2000, pp. 259–269.

[21] R. Sanchez Grandia, V. Aucejo Galindo, A. Usieto Galve, and R. Vives Fos, "General Formulation for Magnetic Forces in Linear Materials and Permanent Magnets," *IEEE Transactions on Magnetics*, vol. 44, no. 9, pp. 2134–2140, September 2008.

[22] Q. Li and D. Schonfeld, "Multilinear Discriminant Analysis for Higher-order Tensor Data Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2524–2537, December 2014.

[23] F. Arslan and A. Grigoryan, "Fast Splitting α-rooting Method of Image Enhancement: Tensor Representation," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3375–3384, November 2006.

[24] R. Moreno, M. Garcia, D. Puig, L. Pizarro, B. Burgeth, and J. Weickert, "On Improving the Efficiency of Tensor Voting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2215–2228, November 2011.

[25] M. Zayani, V. Gauthier, I. Slama, and D. Zeghlache, "Tensor-based Link Prediction in Intermittently Connected Wireless Networks," *Computing Research Repository*, 2011.

[26] L. Liu, Z. Han, Z. Wu, and L. Qian, "Collaborative Compressive Sensing Based Dynamic Spectrum Sensing and Mobile Primary User Localization in Cognitive Radio Networks," in *IEEE Globe Communication Conference (Globecom)*, Houston, TX, December 2011.

[27] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: A New Mobility Model for Human Walks," in *INFOCOM*, Rio de Janeiro, Brazil, April 2009, pp. 855–863.

[28] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the Levy-walk Nature of Human Mobility," *IEEE Transactions on Networking*, vol. 19, no. 3, pp. 630–643, June 2011.

[29] W. Fan and A. Bifet, "Mining Big Data: Current Status, and Forecast to the Future," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, pp. 1–5, April 2013.

[30] R. Rubinfeld and A. Shapira, "Sublinear Time Algorithms," *SIAM Journal on Discrete Mathematics*, vol. 25, no. 4, pp. 1562–1588, Februrary 2011.

[31] M. Figueiredo and A. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, March 2002.

[32] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the Number of Clusters in a Dataset via the Gap Statistic," *Journal of the Royal Statistical Society: Series B*, vol. 63, pp. 411–423, November 2000.

[33] A. Mohsenian, V. Wong, J. Jatskevich, R. Schober, and A. Leon, "Autonomous Demand-side Management Based on Game-theoretic Energy Consumption Scheduling for the Future Smart Grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, December 2010.

[34] S. Shao, T. Zhang, M. Pipattanasomporn, and S. Rahman, "Impact of TOU Rates on Distribution Load Shapes in a Smart Grid with PHEV Penetration," in *IEEE PES Transmission and Distribution Conference and Exposition*, New Orleans, LA, April 2010, pp. 1–6.

[35] L. P. Qian, Y. Zhang, J. Huang, and Y. Wu, "Demand Response Management via Real-time Electricity Price Control in Smart Grids," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1268–1280, July 2013.

[36] S. Bu, F. Yu, and P. Liu, "Dynamic Pricing for Demand-side Management in the Smart Grid," in *IEEE Online Conference on Green Communications (GreenCom)*, New York, NY, September 2011, pp. 47–51.

[37] T. Kim and H. Poor, "Scheduling Power Consumption With Price Uncertainty," *IEEE Transactions on Smart Grid*, vol. 2, no. 3, pp. 519–527, September 2011.

[38] F. Carvalho, P. Brito, and H. Bock, "Dynamic Clustering for Interval Data Based on L2 Distance," *Computational Statistics*, vol. 21, no. 2, pp. 231–250, 2006.

[39] E. Pan, M. Pan, Z. Han, and V. Wright, "Mobile Trace Inference Based on Tensor Voting," in *IEEE Global Communications Conference (GLOBECOM)*, Austin, TX, December 2014, pp. 4891–4897.

[40] B. J. King, "Range Data Analysis by Free-space Modeling and Tensor Voting," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY, 2008.

[41] M. Reisert and H. Burkhardt, "Efficient Tensor Voting with 3D Tensorial Harmonics," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Anchorage, AK, June 2008.

[42] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, NY, June 1993, pp. 786–787.

[43] J. Kang, I. Cohen, and G. Medioni, "Continuous Multi-views Tracking Using Tensor Voting," in *Proceedings Workshop on Motion and Video Computing*, Orlando, FL, December 2002, pp. 181–186.

[44] P. Kornprobst and G. Medioni, "Tracking Segmented Objects Using Tensor Voting," in *IEEE Conference on Computer Vision and Pattern Recognition, Proceedings*, vol. 2, Hilton Head Island, SC, June 2000, pp. 118–125.

[45] A. Narayanaswamy, Y. Wang, and B. Roysam, "3-D Image Pre-processing Algorithms for Improved Automated Tracing of Neuronal Arbors." *Neuroinformatics*, vol. 9, no. 2-3, pp. 219–231, September 2011.

[46] L. Li and D. Boulware, "High-order Tensor Decomposition for Large-scale Data Analysis," in *IEEE International Congress on Big Data*, New York City, NY, June 2015, pp. 665–668.

[47] A. Liavas and N. Sidiropoulos, "Parallel Algorithms for Large Scale Constrained Tensor Decomposition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Australia, April 2015, pp. 2459–2463.

[48] M. Mardani, G. Mateos, and G. Giannakis, "Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2663–2677, May 2015.

[49] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based Scientific Computing in Big Data Analysis," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 71–79, September 2014.

[50] N. Anjum and A. Cavallaro, "Multifeature Object Trajectory Clustering for Video Analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1555–1564, November 2008.

[51] J. Lee, J. Han, and X. Li, "A Unifying Framework of Mining Trajectory Patterns of Various Temporal Tightness," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 6, pp. 1478–1490, June 2015.

[52] D. Mouillot and D. Viale, "Satellite Tracking of a Fin Whale (Balaenoptera Physalus) in the North-western Mediterranean Sea and Fractal Analysis of Its Trajectory," *Hydrobiologia*, vol. 452, no. 1-3, pp. 163–171, March 2001.

[53] D. Zhang and J. Sterbenz, "Robustness Analysis of Mobile Ad Hoc Networks Using Human Mobility Traces," in *11th International Conference on Design of Reliable Communication Networks (DRCN)*, Kansas City, MO, March 2015, pp. 125–132.

[54] H. Liu and J. Li, "Unsupervised Multi-target Trajectory Detection, Learning and Analysis in Complicated Environments," in *21st International Conference on Pattern Recognition (ICPR)*, Tsukuba, Japan, November 2012, pp. 3716–3720.

[55] J. Ko and J. Yoo, "Rectified Trajectory Analysis Based Abnormal Loitering Detection for Video Surveillance," in *1st International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, Kota Kinabalu, Malaysia, December 2013, pp. 289–293.

[56] C. Zahn, "Graph-theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Transactions on Computers*, vol. 20, no. 1, pp. 68–86, January 1971.

[57] P. Mordohai and G. G. Medioni, *Tensor Voting: A Perceptual Organization Approach to Computer Vision and Machine Learning*, ser. Synthesis Lectures on Image, Video, and Multimedia Processing. Morgan & Claypool Publishers, November 2006.

[58] E. Franken, M. van Almsick, P. Rongen, L. Florack, and B. ter Haar Romeny, "An Efficient Method for Tensor Voting Using Steerable Filters," in *Proceedings of the 9th European Conference on Computer Vision*, Berlin, Germany, 2006, pp. 228–240.

[59] W. Freeman and E. Adelson, "The Design and Use of Steerable Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, September 1991.

[60] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 957–968, June 2005.

[61] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "CRAWDAD Data Set ncsu/mobilitymodels (v. 2009-07-23)," Downloaded from http://crawdad.org/ncsu/mobilitymodels/, July 2009.

[62] A. Gorji, R. Tharmarasa, and T. Kirubarajan, "Performance Measures for Multiple Target Tracking Problems," in *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, Chicago, IL, July 2011.

[63] C. J. Needham and R. D. Boyle, "Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation," in *Proceedings of the 3rd International Conference on Computer Vision Systems*, Berlin, Germany, 2003, pp. 278–289.

[64] F. Yin, D. Makris, and S. A. Velastin, "Performance Evaluation of Object Tracking Algorithms," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Rio de Janeiro, Brazil, October 2007.

[65] L. M. Brown, A. W. Senior, Y. li Tian, J. Connell, A. Hampapur, C. fe Shu, H. Merkl, and M. Lu, "Performance Evaluation of Surveillance Systems Under Varying Conditions," in *IEEE PETS Workshop*, Breckenridge, Colorado, January 2005.