CLASSIFICATION OF ELECTRICITY CUSTOMERS BASED ON REAL CONSUMPTION VALUES USING DATA MINING AND MACHINE LEARNING TECHNIQUES AND ITS CORRESPONDING APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUHAMMET TUĞBERK İŞYAPAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2013

Approval of the thesis:

CLASSIFICATION OF ELECTRICITY CUSTOMERS BASED ON REAL CONSUMPTION VALUES USING DATA MINING AND MACHINE LEARNING TECHNIQUES AND ITS CORRESPONDING APPLICATIONS

submitted by **MUHAMMET TUĞBERK İŞYAPAR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. Ferda Nur Alpaslan _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Mehmet Reşit Tolun _____
Computer Engineering Department, TED University

Prof. Dr. Ferda Nur Alpaslan _____
Computer Engineering Department, METU

Prof. Dr. Nihan Kesim Çiçekli _____
Computer Engineering Department, METU

Assoc. Prof. Dr. Tolga Can _____
Computer Engineering Department, METU

Dr. Nuri Alpay Karagöz _____
Head of R&D Department, Innova IT Solutions Inc.

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : MUHAMMET TUĞBERK İŞYAPAR

Signature :

ABSTRACT

CLASSIFICATION OF ELECTRICITY CUSTOMERS BASED ON REAL CONSUMPTION VALUES USING DATA MINING AND MACHINE LEARNING TECHNIQUES AND ITS CORRESPONDING APPLICATIONS

İşyapar, Muhammet Tuğberk

M.Sc., Department of Computer Engineering
Supervisor: Prof. Dr. Ferda Nur Alpaslan

September 2013, 170 pages

Classifying electricity customers based on real power consumptions has been particularly important in the last decade following the liberalization of the electricity markets in numerous countries and ubiquitous use of Automatic Meter Reading devices that collect consumption data at hourly intervals. Collection of vast amounts of consumption data has made it possible to identify customer classes by clustering. Classification of customer load profiles provides the basis of several applications offering solutions to encountered problems in the area. Dedicated tariff design, load forecasting and fraud detection are among deployable applications. Evaluated scalability of the methods reveals an implicit estimation about the size of local regions to which the framework can be applied within reasonable time. Forming further relations among local regions holding consumption data may be useful in developing national energy policies. This study covers a comprehensive scope of the recent work in the problem domain and puts forward foundations of corresponding applications by processing real consumption data of customers of an electricity distribution company in Turkey.

Keywords: Load Profiles, Clustering Algorithms, Classification, Evaluation, Scalability

ÖZ

ELEKTRİK ABONELERİNİN GERÇEK TÜKETİM VERİLERİNİN VERİ MADENCİLİĞİ VE MAKİNE ÖĞRENMESİ TEKNİKLERİ KULLANILARAK SINIFLANDIRILMASI VE İLGİLİ UYGULAMALARI

İşyapar, Muhammet Tuğberk

Yüksek Lisans., Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ferda Nur Alpaslan

Eylül 2013, 170 sayfa

Elektrik abonelerinin gerçek tüketim verileri kullanılarak sınıflandırılması, son on yılda elektrik pazarının birçok ülkede özelleştirilmesi ve saatlik tüketim verisi toplayabilen Akıllı Sayaçların yaygın kullanımıyla birlikte özellikle önemli hale gelmiştir. Yüksek miktarda toplanan tüketim verisi kümelendirme yöntemleri ile abone sınıflarının belirlenmesini mümkün kılmıştır. Abone gruplarını temsil eden yük profillerinin sınıflandırılması, alanda karşılaşılan problemlere çözüm üretecek uygulamaların temelini oluşturur. Özel tarife tasarımı, yük tahmini ve kaçak elektrik kullanımının tespiti geliştirilebilecek uygulamalar arasındadır. Kullanılan yöntemlerin ölçeklenebilirlik değerlendirmesi, sistemin makul sürede uyarlanabileceği yerel bölgelerde tutulacak veri miktarı hakkında dolaylı bir tahmin yürütme imkanı sunar. Yerel bölgeler arasında kurulacak ileri ilişkiler ulusal enerji politikaları geliştirmede faydalı olabilir. Bu çalışma, problem alanındaki güncel çalışmaları kapsamlı bir biçimde inceler ve Türkiye'de faaliyet gösteren bir enerji dağıtım şirketinin abonelerine ait verileri işleyerek ilgili uygulamaların dayanaklarını ortaya koyar.

Anahtar Kelimeler: Yük Profilleri, Kümelendirme Algoritmaları, Sınıflandırma, Değerlendirme, Ölçeklenebilirlik.

To my uncle who lived as an honest worker, a loving father, and a caring person.

To all we lost in the June Days.

ACKNOWLEDGEMENT

I wish to express my gratitude to my supervisor Prof. Dr. Ferda Nur Alpaslan for her advice, guidance, and insight throughout the research.

I would like to thank INNOVA IT Solutions Inc for their interest in this study, and all opportunities they have provided at their company and Boğaziçi Electricity Distribution Company for supplying the consumption data.

I want to thank Karaca for being, as I do not know any better way to thank a person. His sensibility and understanding have helped me greatly in the formation of this thesis.

I am keen on showing my appreciation to Yiğit for his valuable helps regarding the format of the thesis. I would like to thank my friends Hande and Esra, whose presence has called forth my best throughout all these years.

I would like to show my gratitude to my -extended- family, as nobody else could be happier having read this acknowledgment.

Last but not the least, I want to thank my three cat-mates; Kara Bela Dominique, Prenses Pantufya and Aşkiti Pancar. They have been through all stages of this study and now I cannot help admitting how much I love them.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF NOTATIONS

N	Number of data points
K	Number of clusters
I_{max}	Maximum number of iterations until convergence
I	Current iteration number
Q	Dimensionality
X	Set of data points
X_i	Data point indexed by i such that $i \in [1..N]$
X_i^q	Feature q of data point X_i where $q \in [1..Q]$ and $i \in [1..N]$
Ψ	Set of clusters
C_j	Cluster indexed by j such that $j \in [1..K]$
R_j	Center of the cluster indexed by j such that $j \in [1..K]$
R_j^q	Feature q of center vector of cluster C_j where $q \in [1..Q]$ and $j \in [1..K]$
D	Distance matrix
d_{ij}	Distance matrix element indexed by i and j where $i \in [1..N]$ and $j \in [1..K]$
Ω_j	Set of data points belonging to cluster C_j
Ω_j^m	Data point indexed by m in Ω_j where $m \in [1..N_j]$ and $j \in [1..K]$
N_j	Size of cluster C_j
t_z	Number of cluster-changing data points at iteration z where $z \in [1..I_{max}]$
c_1	Centroid initializer constant used in K-means
c_2	Centroid initializer constant used in K-means
M	Initial cluster count variable of K-means++ where $M \in [1..K]$

σ_{WFA}	Weighted fuzzy average
T	Number of training iterations of Hopfield network
$S^{(t)}$	State vector used in Hopfield network at time t where $t \in [1..T]$
$s_i^{(t)}$	State node indexed by i in $S^{(t)}$ such that $i \in [1..N]$ and $t \in [1..T]$
W	Weight vector used in Hopfield network.
w_{ij}	Weight indexed by i and j such that $i, j \in [1..N]$
K'	Desired number of clusters
θ_c	Minimum inter-cluster distance threshold
θ_m	Maximum number of merges threshold
θ_s	Maximum intra-cluster standard deviation threshold
θ_N	Minimum cluster size threshold
ε	Error threshold
Q_{width}	Width of Self Organizing Map
Q_{height}	Height of Self Organizing Map
r	Neighborhood radius function of Self Organizing Map
f	Distance falloff function of Self Organizing Map
η	Learning rate used in the training of Self Organizing Map
n	Neighborhood function of Self Organizing Map
l	A node of the Self Organizing Map
b_{X_i}	Best matching unit of the data point X_i in Self Organizing Map

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Classifying electricity customers based on real power consumptions has been noteworthy in the last decade following the liberalization of electricity markets in numerous countries. Economic processes have been encouraging evermore participants including distributors and retailers to take part in the market. Customers could benefit from the situation by choosing the optimal price for the service they receive. In order to the pave the way for such a development, correct knowledge of customer classes should be identified.

Customer classes can be used in developing business strategies as the business owner can deduce the most profitable groups via continuously monitoring their customers. Temporal changes of customers among identified classes shall constitute an important factor in forecasting revenues and in evaluating quality of the services provided. Similarly, seasonal and occasional behavior changes in electricity consumption are significant facts that shape the composition of customer classes in such a way that classification of customers must be conducted separately for weekdays, weekends and holidays together with seasons.

Electrical energy cannot be stored for re-use. Therefore, forecasting future electricity consumptions based on previous real consumption amounts regarding each customer class is of the utmost importance. Formed customer classes should be compact and well-separated from each other and be as robust as possible against the presence of noise so as to develop applications upon them. In order to stimulate consumption or disposal of then-current electrical energy, appropriate strategies could be established based on characteristics of each customer class. These characteristics are useful in planning dedicated tariff structures and future investments, as well. In short, discovered customer classes are of significant informative value for characterizing and manipulating the whole electricity market.

The most contemporary way to collect real electricity consumptions of customers is employing *Automatic Meter Reading* (AMR) systems that collect consumption and status data from electric metering devices and transfer the data to central databases in predefined intervals. By inspecting the data obtained in a relatively long period,

representative load diagrams for every customer can be computed. Representative load diagrams of a set of customers can be clustered into several consumer classes, each of which is characterized by a representative load profile by using a set of clustering algorithms proposed in the data mining literature.

Formation of representative load profiles offers fundamental solutions to problems associated with the electricity domain. Possible positive uses of classification based on consumption shapes include designing class-specific tariffs to give customers new degrees of freedom; forecasting future electricity consumption to plan the amount of electricity the country needs so that dependency to other countries that sell energy can be diminished and waste can be prevented; and detecting fraud to develop policies against all kinds of misuse.

1.2 Overview and Scope of the Study

Using the knowledge of customer classes obtained by computational techniques that process data from AMR devices offers various advantages over forming macro categories based on contractual and billing data or surveys. Currently, macro categories such as household and commercial are used as part of the national energy policies of Turkey to classify electricity customers. Each macro category comes with dedicated tariffs.

A study conducted in Finland puts forward that classes emerged through the use of data mining techniques are more accurate than macro categories and members of a major category could indeed be distributed across obtained classes (Räsänen et al., 2010). It is worth noting that Finland electricity distribution companies traditionally utilized several more macro categories than the ones in Turkey. Furthermore, intuitively, there is no direct correlation between categories separating a house and an office and the related electricity consumption patterns. Discovering natural clusters on real consumption data makes it possible for more criteria to emerge that can classify customers more accurately compared to major categories.

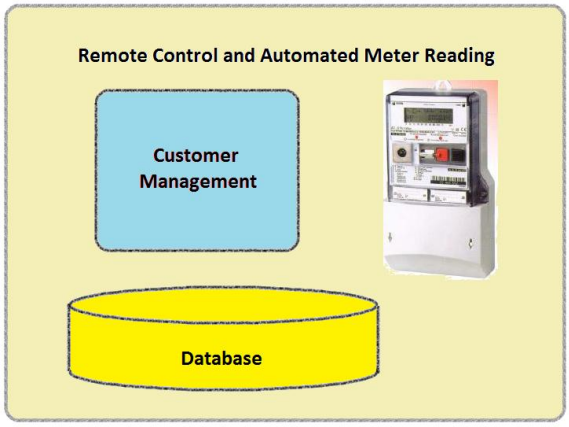

Figure 1.1 Components of load profile classification.

Components necessary to conduct load profile classification is illustrated in (Fig. 1.1). Data collected over AMR devices are transferred remotely to a central database for storage. Customer management deals with data collected over at least one year so as to perform classification to discover representative load profiles of all customers and to deploy applications based on the results of the classification.

The study involves a two-step processing of data by first clustering load diagrams, then classifying the recently-formed labeled data for generalization purposes. It combines supervised and unsupervised learning as in (Pao and Sobajic, 1992). Components included in the customer management task in (Fig. 1.1) are visualized in the following.

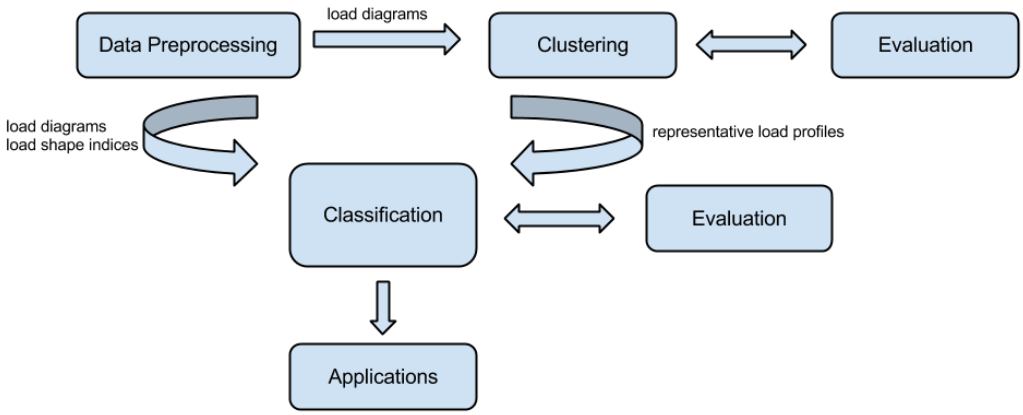

Figure 1.2 A modular representation of components of customer management.

This study aims to highlight components of customer management shown in (Fig 1.2) and focuses primarily on discovering natural clusters of customers computed on real consumption data for different seasons along with weekdays, weekends, and holidays;

and evaluating the quality of performance of generated clustering in such a way that discovered clusters shall be compact and well-separated from each other to be reckoned as having informative value.

Data pre-processing is a crucial step which helps making sense of vast amounts of consumption data by forming representative load diagrams of customers for weekdays, weekends and holidays of both winter and summer seasons. Load diagrams are calculated by taking averages of per hour consumption amounts at specified date intervals. Meanwhile, missing values are replaced; outliers are detected and smoothed; and obtained diagrams are normalized to go through the clustering step. Data pre-processing step also includes computation of load shape indices that characterize the consumption summarized in a load diagram.

Next, representative load diagrams are clustered by several clustering methods. Centers of generated clusters are adopted as representative load profiles of customers residing in them. Clustering algorithms are evaluated by numerous validity indices to reveal the quality of performance regarding produced clusters. Clustering and evaluation are the two steps that work in a coherent fashion so that designs of algorithms are improved following their evaluation. Moreover, performance quality evaluation provides means for comparing clustering algorithms.

Evaluation of clustering algorithms is also carried out based on scalability criterion. This work aims to reveal scalability of clustering algorithms by testing them on larger amounts of data, which shall be of great value to predict average sizes of data sets for which automatic customer class formation is proved to be effective. Consequently, higher-level relations can be formed among several local regions holding the data of a particular number of customers for possible use in developing national energy policies.

Following the discovery of natural clusters on consumption data that is processed into forms of occasional load diagrams of customers, classification is performed to highlight the rules regarding consumption shapes that make a customer be part of a particular cluster. This paves the way for devising applications such as customer recognition that is used to place new customers in correct classes, forming class-specific tariffs, and proposing tariffs to new customers by using the results of customer recognition. Foundations for load forecasting and fraud detection applications are also explained in the study, although their deployable applications are not designed or evaluated due to lack of available consumption data collected over several years.

As a summary, this study is bound within the scope of clustering and classification of electricity customers that are represented by seasonal load diagrams for weekdays, weekends and holidays computed over collected data in a one-year period. An elaborated evaluation of employed methods regarding both accuracy and scalability is presented as well. Possible uses of load profile classification are explained through proposed applications.

1.3 Contributions

This study presents a prototype of customer management for an electricity distribution company. Literature review included in the next chapter covers numerous pioneer works in the area with contributions from all over the world. However, problem domain is yet to draw a similar attention in Turkey. For this reason, the scope of the study is kept quite comprehensive in order to provide a large-scale example work.

Prominent algorithms of the literature are evaluated on the data provided by Boğaziçi Electricity Distribution Company in Istanbul. Following modifications and design improvements are among contributions of the thesis.

- Clustering algorithms that are originally not covered by load profiling literature such as K-means++ and Similarity-Based K-means using point symmetry distance are analyzed.

- Particular clustering algorithms are modified such as using centroid initialization proposed by K-means++ in algorithms such as K-medians and WFA-K-means and allowing new cluster formation in later iterations of Follow-the-Leader algorithm.

- Effects of designing hybrid distance heuristics on the performance of clustering algorithms are thoroughly discussed and a hybrid Pearson distance framework is proposed, which supersedes the performance of well-known Euclidean metric in particular cases.

- Scalability of clustering algorithms is analyzed elaborately. In order to decrease running times and increase scalability of Hierarchical algorithms, two hybrid methods, Hierarchical–K-means++ and Hierarchical–Follow-the-Leader, are designed.

Regarding all methods covered in this study, more recent work is given particular importance in the content of this study so as to motivate the authorities in Turkey to adopt load profile classification in electricity domain.

1.4 Structure of the Thesis

Following chapters discuss load profile clustering and classification in a detailed fashion. Chapter 2 covers the related work and recent improvements in the problem domain. Chapter 3 presents characteristics of used data sets and data pre-processing. Detailed descriptions of clustering algorithms are included in Chapter 4.

Performance quality of each clustering method is evaluated by means of various validity indices in Chapter 5, which also includes illustrations of generated clusters by each algorithm. Chapter 6 presents a discussion about running time complexities and scalability of clustering methods to increasing amounts of data and offers solutions that decrease running times and increase scalability.

Chapter 7 includes classification of load profiles by using load shape indices and actual features with class labels inherited from generated clusters. Throughout this chapter, prototypes of applications based on load profile classification are exemplified.

The last chapter, Chapter 8, contains conclusions of the study along with discussions and guidelines for future work.

CHAPTER 2

RELATED WORK

2.1 Introduction

Data mining techniques have been rapidly applied to various problem domains including power systems since the turn of the millennium. According to the survey conducted in (Mori, 2006) problem areas include security assessment (48.8%), fault detection (11.6%), power systems control (9.3%), load forecasting (6.9%), and load profiling (6.9%). The problem of classifying electricity customers based on their electrical behaviors is heavily inspected in load profiling area.

Application of data mining to customer segmentation is useful as a customer profile analysis for the establishment of business strategies (Chicco et al., 2001; Kitayama et al., 2002). Relationship between the amount of the electric power use and the load factor reveal information about the regular and preferred, i.e. eligible customers. Temporal changes between the preferred and regular segments can be used in evaluation of the planned service (Kitayama et al., 2002). Formed customer classes can also be used in obtaining better support management decisions in terms of planning of bids and energy offers in real-time energy markets (Gabaldón et al., 2010), optimum fixed or real-time price offering (Mahmoudi-Kohan et al., 2010), and short-term (24-hour-ahead) price prediction (Zareipour et al., 2011).

Surveys were employed to identify electricity consumption behaviors of customers in late 1990's. Data collected by surveys are manipulated by statistical techniques to arrive at major customers classes varying from residential customers to extra high voltage customers, to which distinct prices for consumption at particular hours are assigned so that more effective load management alternatives to reduce the system peak demand is carried out (Chen et al., 1997).

Predefined macro categories formed through surveys or simple contractual data does not take into account possible misinformation, irregular consumer behaviors, changes between customer classes and regional differences (Chicco et al., 2005b; Mutanen et al., 2011). Electricity consumption data of customers is gathered by Automatic Meter Reading (AMR) systems which record true values of power usage in predefined time intervals such as every 15 minutes or one hour (Piao et al., 2010) to overcome disadvantages of traditional techniques. Mathematical comparison techniques indicate the

superiority of AMR to prior means of data collection (Chicco, 2005a) and developed countries such as Finland require its distribution operators to equip at least 80% of the customers with AMR by the end of 2013 (Mutanen et al., 2011).

2.2 Load Profile Clustering and Classification

Data collected via AMR devices are stored to form effective customer classes with the help of appropriate automatic clustering techniques instead of the traditional few parameters and commercial codes (Chicco et al., 2001).

Load diagrams for each customer can be identified by properly averaging the consumption data and normalizing the result in the interval [0, 1] to obtain representative load diagrams which are used as inputs to clustering algorithms. Clustering results lead to formation of class representative load diagrams for each customer class built on the basis of load diagrams aggregated in the same customer class which may be used in several applications.

Clustering proceeds in two phases: *pre-classification phase* which includes data preparation, obtaining measurements, bad data detection (Gerbec et al., 2005), data reduction (Chicco et al., 2006), identification of loading conditions (seasons, weekends, holidays), and definition of representative load patterns for each customers and *classification phase* which includes feature selection like picking time-domain data (Verdú et al., 2006), load shape factors including indices such as lunch impact and night impact (Figueiredo et al., 2005), harmonic coefficients; customer classification by a suitable technique; class representative load patterns computation and classification adequacy measurement (Chicco et al, 2005b).

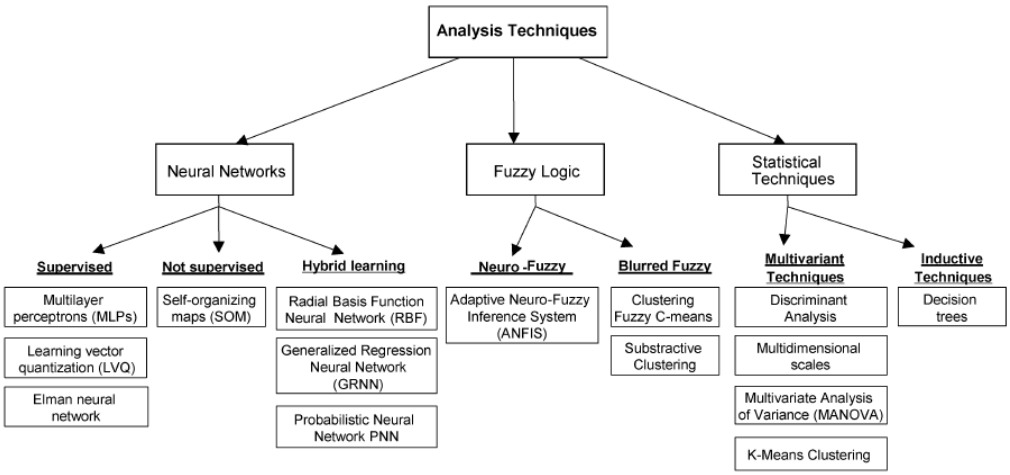

Figure 2.1 Analysis techniques heavily used in load profiles classification domain (Verdú et al., 2006)

Various algorithms are proposed in load profiling area. These data analysis techniques can be categorized depending on computational and mathematical backgrounds, as illustrated in (Fig. 2.1). Several methods are described in the following.

K-means clustering requires the knowledge of number of clusters to operate and applies in cycles to assign data points to clusters and to update cluster centers according to the optimum distance arrangement between cluster centers and data points (Chicco et al., 2003). Although traditional K-means algorithm sets initial cluster centers randomly, centroids can also be initialized using constant values (Tsekouras et al., 2008).

WFA K-means uses a type of fuzzy averaging that puts the center prototype among the more densely situated points and it approximates to the true WFA value following five iterations (Mahmoudi-Kohan et al., 2008). Performance of the algorithm can be improved with the use of new distance computation metrics that take into account the variance of a particular feature from the corresponding average (Mahmoudi-Kohan et al., 2009b).

Fuzzy K-means (Fuzzy C-means, FCM) is an algorithm that is based on the idea that every data point belongs to a cluster at some degree specified by membership grades which is updated according to some objective function at each iteration. When the algorithm converges, final cluster centers are chosen as typical load profiles of the samples (Chang and Lu, 2003).

Modified Follow-the-Leader method forms new clusters or assigns data points to existing clusters depending on its operating distance threshold by traversing data instances in cycles until cluster stabilization is attained. Modified Follow-the-Leader algorithm uses a weighted Euclidean heuristic in the calculation of distance that amplifies the impact of

high-variance features and requires a user-defined similarity threshold to build and refine clusters (Chicco et al., 2004).

Hierarchical clustering generates dendrograms that keep the history of the clustering process, which begins with assigning each data point to singleton clusters and continues by iteratively merging clusters according to various linkage criteria such as average linkage and Ward's linkage (Chicco et al., 2003).

Self organizing map (SOM) is an unsupervised neural network that projects a multi-dimensional dataset into a reduced dimensional space. SOM updates not only the weights of the winning unit, but also the weights of its neighbors in inverse proportion of their distances (Chicco et al., 2004). SOM is often combined with other clustering algorithms to serve as an intermediate step of the clustering process (Räsänen et al., 2010). SOM is also used for anomalous behavior filtering to detect outliers due to external factors (Verdú et al., 2006).

Probabilistic neural networks (PNN) represent a form of unsupervised learning and comprises of the input layer that takes the average profile of the respective activity as input, the radial basis layer has its neurons corresponding to the average load profile of an individual cluster obtained by FCM, and the output layer. PNN classifies an input pattern into the most suitable one of the obtained classes (Gerbec et al., 2005).

Entropy-based clustering is a multi-step hierarchical process that iteratively merges most similar clusters to reduce number of clusters based on the notions of similarity between two load patterns and similarity between two clusters that are devised via between-cluster entropy computation (Chicco and Akilimali, 2010).

TS-part algorithm does not require the exact knowledge of number of clusters and proceeds iteratively by computing the best split point for large clusters via considering the inter-cluster and intra-cluster distances (Gullo et al., 2009). ISODATA is another algorithm that proceeds iteratively by applying a variant of K-means, splitting or merging the clusters depending on the computed Euclidean distance (Mutanen et al., 2011). For a comprehensive review of clustering algorithms, refer to (Jain et al., 1999) and (Xu and Wunsch, 2005).

Decision trees are used for classification purposes. Several entropy definitions that deal with normalized monthly energy usages, monthly and seasonal changes, and normalized average seasonal energy usage are used in a standard C4.5 induction algorithm (Chang and Lu, 2003). Hybrid induction algorithms produce rules from committees of decision trees (Piao et al., 2010).

SVM-based classification methods determine separating hyper planes to distinguish data classes in such a way that hyper planes have the maximum possible distance from either of the data classes (Zareipour et al., 2011).

2.3 Evaluation of Algorithms

Algorithms are evaluated according to quality of performance via several validity metrics. Two of the most commonly used metrics are Mean Index Adequacy (MIA) and Clustering Dispersion Indicator (CDI). MIA is used to assess the compactness of clusters while CDI is used to measure the degree to which clusters are separated from each other (Ramos and Vale, 2008; Mahmoudi-Kohan, 2009a). Less values of MIA and CDI metrics indicate that the algorithm exhibits a higher quality of performance.

Among other validity metrics, there are Similarity Matrix Indicator (SMI) (Chicco et al., 2003), Davies-Bouldin Index (DBI), Modified Dunn Index (MDI), Scatter Index (SI) (Chicco et al., 2006), Mean Average Percentage Error (MAPE) (Gerbec et al., 2005), the ratio of Within-Cluster Sum of Squares to Between-Cluster Variation (WCBCR) and the number of dead clusters (Tsekouras et al., 2008). Computational complexities of algorithms can be compared using heuristics such as the relative computation time index (RCTI) (Chicco et al., 2003).

There are particular trade-offs in adopting an algorithm related to characteristics of the data regarding the content and the size, and the objectives of the application (Mahmoudi-Kohan, 2009a). Moreover algorithms follow certain tendencies. Advantages include the speed of statistical algorithms (K-means and Follow-the-Leader), possibility of reallocations in successive iterations (K-means, and Follow-the-Leader), no need of training (K-means, Follow-the-Leader, and Hierarchical algorithm), interpretation and visualization of results (SOM and Hierarchical algorithms), and stability of the obtained solution (Follow-the-Leader, and Hierarchical algorithm). Disadvantages include randomness of the solution obtained in K-means and SOM, non-reallocation of data points in successive steps (Hierarchical algorithm), lack of objective function (Hierarchical algorithm), and implementational difficulties of the modified Follow-the-Leader approach (López et al., 2011).

Design of algorithms can be extended to arrive at hybrid methods (Räsänen et al., 2010; López et al., 2011) and frameworks that apply clustering algorithms in successive cycles (Figueiredo et al., 2005) or frameworks that apply a set of algorithms to a particular dataset many times to discover optimum algorithmic parameters and then evaluate them according to several metrics (Tsekouras et al., 2008). However most studies indicate that new algorithm designs should compete with the robust and adequate performances of the Modified Follow-the-Leader approach and Hierarchical algorithms (Chicco et al., 2003; Chicco et al., 2005b).

2.4 Applications

Tariff design based on generated customer classes offers solutions to increase profits of retailers in the market by maximizing a profit function based on customer revenues, supply cost, and risk penalty (Mahmoudi-Kohan et al., 2010). Both annual and seasonal frameworks can be generated as part of the business strategy regarding retailers (Panapakidis et al., 2012). Revenue assessment through area-based and customer class-based load profiling techniques puts forward the superiority of customer classes generated by suitable clustering methods (Chicco, 2005a).

Customer classes can be used in tariff design to reduce peak hour demands (Morais et al., 2013). Time-of-use pricing, critical peak pricing, real-time pricing, real-time pricing with load limit and extreme real-time pricing are among pricing techniques that can be applied to generated clusters of customers. Experiments with customers reveal that users tend to prefer dynamic pricing, which offers them the opportunity to adjust their consumption levels. However, there is a trade-off concerning daily habits that some customers may not be keen on changing (Paetz et al., 2012).

Consumption patterns should be modeled before estimating future demands of electricity. Besides clustering algorithms, statistical constructs such Markov models can be used for this purpose (Labeeuw and Deconinck, 2013). Together with representative load profiles, models of daily consumption over weeks may be necessary (Zhang et al., 2013). Load forecasting can be realized by using regression methods (Cho et al., 2013), artificial neural networks (Felea et al., 2012), association rules (Piao et al., 2008), decision trees and SVM classifiers.

Fraud detection can be realized by using fuzzy logic constructs such as rough sets (Cabral and Gontijo, 2004). Distance-based, set-based, density-based, depth-based and model-based techniques are used to discover customer irregularities before processing these outliers by several prediction and classification methods including Support Vector Machine and Extreme Learning Machine to detect actual frauds (Nizar and Dong, 2006).

Ideas from similar other domain such as telecommunication industry may come handy in power domain. SVM, neural networks and decision trees can be combined so that normal clusters, outlier clusters and inconsistent clusters of customers are discriminated. (Farvaresh and Sepehri, 2011).

CHAPTER 3

DATA PREPARATION

3.1 Introduction

Data preparation is a crucial step before clustering and classification. Representative load diagrams for each customer regarding every external load condition should be computed before any further process that intends to discover representative load profiles or deploy an application.

While forming load diagrams, fundamental tasks of data pre-processing is carried out including dealing with missing values, detecting outliers, smoothing and normalization.

3.2 Characteristics of Used Data Sets

Electricity consumption amounts measured per hour for the interval 01 January 2012 to 31 December 2012 by Boğaziçi Electricity Distribution Company in Istanbul, which provides electricity service to European side of Istanbul, are used in this study.

Consumption data belongs to mostly high voltage customers including factories, offices and grand houses, which may exhibit distinct consumption styles.

Table 3.1 Number of customers whose data available at months of 2012.

	Months (2012)											
	01	02	03	04	05	06	07	08	09	10	11	12
Size	259	268	327	329	329	321	245	242	242	249	247	244

As data in Table 3.1 suggests, number of instances of monthly data is not evenly distributed throughout the months. Moreover, missing values exist regarding particular hours of the days to be processed during the course of data preparation.

3.3 Data Preparation Tasks

Data preparation involves identifying loading conditions and dividing data instances accordingly for each customer, dealing with missing attribute values, outlier detection and smoothing, generating monthly and seasonal load diagrams, and finally normalizing seasonal load diagrams so that customer representative load diagrams are formed for clustering and classification (Kotsiantis et al., 2006).

Statistical operators used in data preparation are briefly described in the following. Let X be the set of instances indexed as x_i and let the cardinality of X be N. Then, the mean of the values in the set is computed subsequently in equation (3.1).

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad , \text{ where } i \in [1..N]. \tag{3.1}$$

Let X' be the set that is composed of elements of X, sorted in an ascending order. Median operator computes the most centrally located data instance as follows.

$$If\ |X'|\ is\ odd, then\ i = \frac{N+1}{2} and\ median(X) = x'_i\ . \tag{3.2}$$

$$If\ |X'|\ is\ even, then\ i = \frac{N}{2}\ and\ median(X) = \frac{1}{2}\ (x'_i + x'_{i+1})\ .$$

Trimmed mean, also known as truncated mean, computes the average value of a data set by excluding a particular percentage of extremum values of the sorted data set X'. Let p be the percentage of data excluded at high and low ends. Trimmed mean is computed as follows.

$$n = \left\lfloor N\ \frac{p}{100} \right\rfloor \tag{3.3}$$

$$\bar{\bar{X}} = \frac{1}{N-2n} \sum_{i=n+1}^{N-n} x'_i$$

Median operator in equation (3.2) is an extended version of trimmed mean in (3.3), in which percentage of the excluded extremum values are adjusted to leave out only the most centrally located value in the set. Both operators are utile in outlier detection and exclusion.

Inter-quartile range measures the difference between upper and lower quartiles of the data, which corresponds to the absolute value of difference between 25th and 50th percentile marked by Q_1 and Q_3 respectively. IQR is computed in the subsequent equation.

$$Let\ l = \left\lceil N\frac{25}{100} \right\rceil\ and\ u = \left\lceil N\frac{50}{100} \right\rceil \tag{3.4}$$

$Q_1 = x'_l\ and\ Q_3 = x'_u$.

$IQR(X) = |Q_1 - Q_3|$.

Note that median of the set corresponds to 50th percentile and is denoted by Q_2.

3.3.1 Identifying Weekdays, Weekends and Holidays

Date categorization is among important loading conditions. It is evident that there are changes in electricity consumption amounts and shapes between working days and holidays.

Information on weekdays, weekends and holidays of year 2012 is used to divide consumption data into three parts for every month. Official national days and feasts of Turkey are selected as holidays. Conflictions among holidays and other days are resolved by excluding data of holidays from weekdays and weekends data.

While dividing data by day information, missing attribute values that are indicated by empty strings in the original format of the obtained data are marked with a special character to be processed subsequently. Moreover, more than one copy of the data of all customers is eliminated.

3.3.2 Missing Values

Missing values occur due to device or human errors while gathering, transferring or processing data. Frequency of missing values in the utilized consumption data is low.

If a missing value is observed for an hour of a particular day while forming monthly and seasonal representative load diagrams, it is replaced by the truncated mean of the remaining instances.

More sophisticated methods that deal with missing values are not used in the study due to low frequency of missing values in the consumption data, except consumption data of two customers containing more than 80% missing values of all their measurements. These customers are excluded from further steps of the study.

Missing values can be filled by classification algorithms. Pre-classification configuration imposes that the attribute with the missing value is selected as the class label while remaining are the attributes of the training data composed of instances that strictly do not have their class labels missing. Once the classification is performed, the instance with the

missing value is used as the test data and the predicted class becomes the value that fills the missing attribute.

Regression techniques, instead of classification algorithms, can also be used in the mentioned configuration to fill a missing value.

3.3.3 Outlier Detection and Smoothing

Inter-day outliers regarding consumption values per hour can be detected during the computation of monthly and seasonal load diagrams.

In order to detect outliers, following statistical property is employed.

$$Values\ of\ X\ residing\ outside\ [Q1 - 1.5\ IQR, Q3 + 1.5\ IQR]\ are\ outliers. \quad (3.5)$$

While computing load diagrams, outliers are detected for each hour consumption value and their value is diminished to be at most $Q3 + 1.5\ IQR$ or at least $Q1 - 1.5\ IQR$ depending on whether they are higher than the upper limit or lower than the lower limit indicated in equation in (3.5).

Proposed outlier detection and smoothing method applies to less than 5% of all customers regarding all months. More sophisticated methods used in outlier detection include clustering algorithms or other statistical criteria such as Peirce's criterion. Since the frequency of outliers is so low, sophisticated methods are not used in this step of data preparation.

Dealing with outliers, informational loss caused by simply eliminating them should be carefully considered. Outliers are only smoothed while computing hourly average values during the course of load diagram formation in order to prevent them from misleading the results of relatively less robust methods such as the mean operator in (3.1).

Diagrams residing within divided consumption data sets that could be detected as outliers may actually exhibit merely distinct consumption shapes, whose discovery is of informative value. These kinds of outliers are neither excluded, nor smoothed in data preprocessing so as to evaluate clustering algorithms based on their powers of discriminating outliers in separate clusters.

3.3.4 Forming Monthly Load Diagrams

Monthly profiles of customers are formed by using mean, trimmed mean, and median operators, respectively.

Mean is the regular statistical average operator working on consumption amount for every day of the month. Median chooses the middle consumption data for every hour of the sorted data instances. Trimmed mean, however, sorts data instance for every hour of the day separately, leaves aside 10% of the values lying closer to the extremum, and runs the regular statistical mean operator on the remaining instances.

Median and trimmed mean are more robust to outlier instances because the first one searches the most centrally located instance and the second excludes near-extremum values from the computation. Trimmed mean operator runs a higher probability of capturing the average consumption throughout the month since it works on a set of possibly non-outlier instances, while median operator outputs only one instance, which may result in information loss.

During the course of monthly representative load diagrams computation, missing attribute values are replaced with the trimmed mean value computed over all instances in the month, and intra-day hourly outlier values are detected and smoothed through re-adjusting their values to the maximum of the interval of values that are not considered as outliers.

Regarding each month, data of a customer is generated for weekdays, weekends and holidays and by separately using mean, trimmed mean and median operators.

3.3.5 Presence Check Across Monthly Load Diagrams

Having computed the monthly representative load diagrams, presence of a customer's diagram in all months of the year should be checked before the formation of seasonal diagrams. As data in Table 3.1 suggests, number of load diagrams varies throughout months.

Therefore, presence of load profiles in every month should be summarized in a table structure starting from January 2012. Number of data instances that appear for the first time in a particular month starting from the beginning of the year is included in the subsequent table.

Table 3.2 Number of customers whose data first appears in indicated months.

	Months (2012)												
	01	02	03	04	05	06	07	08	09	10	11	12	Total
Size	258	10	62	5	1	3	2	0	0	7	1	0	349

Table 3.2 shows that data of a customer may appear for the first time as late as in November 2012 and a total of 349 distinct customers' consumption data is present in the whole data set. Note that the table does not indicate whether existence of a customer's data that first appeared in the first month will pursue in subsequent months. The presence check table, which is not included in this section due to its size, holds the mentioned information.

Information on customers' presence throughout the months of 2012 is useful while computing seasonal load diagrams in the next section.

3.3.6 Forming Seasonal Load Diagrams

Load diagrams are further divided by winter and summer loading conditions, which is expected to differ due to effects of external factors such as temperature on electricity usage. Summer is assumed to be between April and September, while winter is assumed to be between October and March.

The presence check described in the previous section is used in forming seasonal load diagrams in such a way that if data associated to a company does not appear in at least three months for a season, it is discarded from further clustering operations. In total, 43 and 31 monthly load diagrams are excluded from winter and summer seasons, respectively. In other words, missing values are not filled but discarded during the computation of seasonal load diagrams, since it may be incorrect filling missing values for only six months at this scale of aggregation. Therefore, the final cardinality of winter load diagrams is 306 and that of summer diagrams is 316.

Although, cardinalities of both seasonal diagrams should be kept the same in order to arrive at more conclusive results, data used in the study is limited. Hence, a relaxation to this objective is preferred for a slightly more comprehensive evaluation of clustering algorithms. Moreover, discovery of distinct diagrams residing in two seasonal sets are crucial to generate models for synthetic data to be used in running time and scalability evaluation in Chapter 6.

Seasonal load diagrams are formed by averaging hourly consumption data of monthly load diagrams. If a customer's data appears in all six months of a season, averaging is

carried out by mean, trimmed mean and median operators. Trimmed mean is modified to exclude only two extremum hourly consumption amounts of all months of the season. If the customer's data does not appear in all six months, then only mean and median operators are used in order not to lose any more information from the existing data.

At the end, winter and summer representative load diagrams for weekdays, weekends and holidays are obtained.

3.3.7 Normalization

Normalization of attribute values for all computed load diagrams residing in winter and summer weekdays data sets is necessary so that bias due to measured consumption amounts can be prevented during distance computation and shapes of consumptions can be used as discriminative features.

Min-max normalization of a vector is computed as follows.

$$x_i^{norm} = \frac{x_i}{\max\{x_i\}} \quad for\ all\ x_i \in X. \tag{3.6}$$

All diagrams are normalized by their own peak demand as indicated by equation in (3.6) because normalizing every data instance by a global constant would result in losing details about consumption shapes of particular customers. In pre-processing of load diagrams, z-score normalization is also used (Kotsiantis et al., 2006).

3.4 Load Shape Indices

Load shape indices offer characterizing consumption shapes of a customer under particular loading conditions in less number of features than the 24 actual features corresponding to a day. These indices can be used in clustering and classification of representative load diagrams.

A list of load shape indices defined in (Verdú et al., 2004) is included subsequently.

Table 3.3 Load shape indices.

	Computation
f_1	$P_{av,day}/P_{max,day}$
f_2	$P_{av,daylight}/P_{max,day}$
f_3	$P_{min,day}/P_{av,day}$
f_4	$1/3\ (P_{av,night}/P_{av,day})$
f_5	$1/5\ (P_{av,lunchtime}/P_{av,daylight})$
f_6	$P_{min,daylight}/P_{av,daylight}$
f_7	$P_{av,night}/P_{av,daylight}$
f_8	$1/5\ (P_{max,daylight}/P_{max,day})$
f_9	$P_{min,day}/P_{min,daylight}$

Notation used in Table 3.3 is explained as follows. Day is the set of all 24 features of a diagram, daylight is the time interval from 7.00 in the morning until 18.00 in the evening, night is from 22.00 in the evening till 6.00 in the morning, and lunchtime is between 12.00 and 14.00. $P_{min,\langle time\rangle}$ refers to the minimum consumption amount for the interval $\langle time\rangle$, and $P_{max,\langle time\rangle}$ and $P_{av,\langle time\rangle}$ stand respectively for maximum and average values of consumptions observed in course of the period marked with $\langle time\rangle$.

Load shape indices that are included in Table 3.3 yield information about daytime, lunch and night impacts on daily consumption along with effects of the minimum and the maximum demands in particular hours of the day. Constants in the formulas of f_4, f_5 and f_8 are used to diminish the effect of these indices while being used by methods of clustering and classification and to prevent these indices from taking unity values.

Load shape indices are computed on seasonal load diagrams prior to normalization. Note that shape indices normalize a customer's electricity consumption on an average day under particular loading conditions by using only the data within their scope, just like min-max normalization covered in the previous section.

3.5 Visualization of Seasonal Load Diagrams

Seasonal representative load diagrams can be visualized by using Follow-the-Leader clustering algorithm, whose details and evaluation is included in the subsequent chapters. All diagrams residing within the same cluster are displayed in the same color so that distinct consumption shapes among customers can be visually identified.

Regarding all included diagrams, horizontal axis holds hours of the day and vertical axis holds normalized consumption values.

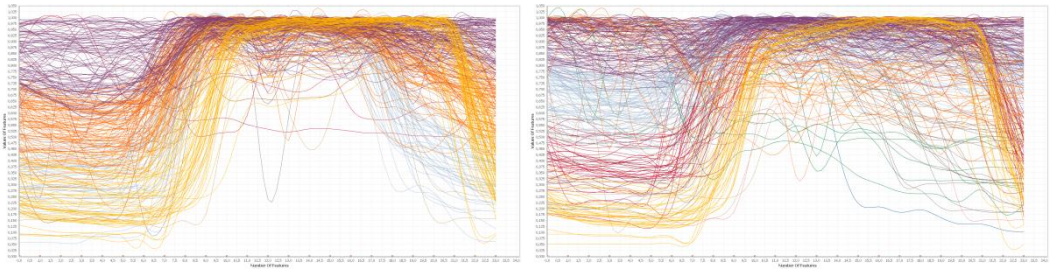

Figure 3.1 Normalized load diagrams of winter weekdays and weekends as clustered by Follow-the-Leader algorithm.

Inspecting the contents of winter weekdays and weekends data, which are respectively illustrated on the left and the right in (Fig. 2.1), reveals that weekends data includes more distinct consumption shapes than weekdays data. Although some of the profiles seem to be conserved in both data sets, visualization of winter data indicates the information gain by separating weekdays and weekends data.

Summer weekdays and weekends data are illustrated in the following.

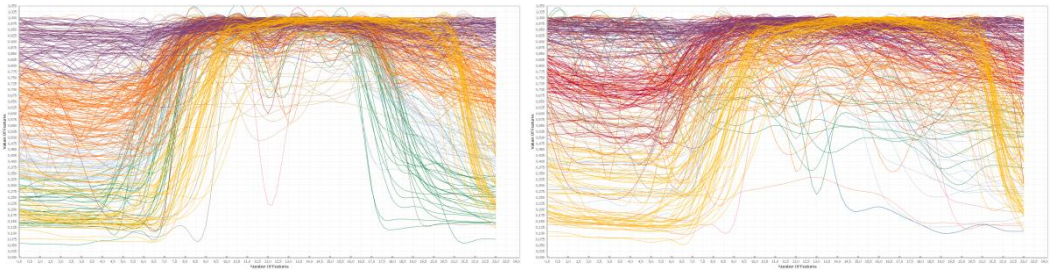

Figure 3.2 Normalized load diagrams of summer weekdays and weekends as clustered by Follow-the-Leader algorithm.

Illustration in (Fig. 3.2) also confirms that weekends and weekdays data sets indeed include distinct shapes of consumption. A quick look in both winter and summer load diagrams indicates the seasonal impact on load profiles, despite the fact that some clusters of diagrams exhibit little variance between seasons.

3.6 Summary

Data preparation is an obligatory task before proceeding with clustering and classification of electricity consumption. The goal of data preparation is forming normalized customer representative load diagrams in order to make sense of large amounts of data.

Load diagrams are formed by aggregating daily consumption data into monthly and seasonal diagrams. Meanwhile, weekdays, weekends and holidays serve as load conditions that separate monthly profiles into three groups. Seasonal load diagrams impose the winter-summer division over monthly profiles.

While aggregating daily profiles into higher levels in the hierarchy, hourly values are combined by mean, trimmed mean and median operators. During the process, missing values are filled and inter-day outliers are detected and smoothed. In addition to this, presence check regarding all customers' data in months of the year is conducted prior to the formation of seasonal load diagrams.

Normalization is a tool used on winter and summer load diagrams so that all data is mapped within [0, 1] interval. This makes it possible to compare consumption shapes rather than measured values.

Instead of actual 24 features of load diagrams, normalizing load shape indices can be used for clustering and classification purposes, which are covered in the subsequent chapters.

CHAPTER 4

CLUSTERING ALGORITHMS

4.1 Introduction

Clustering of customers of an electricity distribution company based on real consumption values per hour in the format of load diagrams per season is the most crucial operation that reveals the natural groupings inside seasonal data sets to be used in the computation of class-representative load profiles.

All clustering algorithms covered in this chapter works on pre-processed and normalized data and yields clusters whose representative centers constitute the load profiles regarding all customers. Further classification and application deployment steps strictly depend on clustering operation carried out by the algorithms described in detail throughout the chapter.

4.2 Problem Definition and Distance Frameworks

Clustering process involves dividing data instances into distinct sets, each of which are characterized by their centers. Instances residing in the data set are Q-dimensional vectors, whose position in the data set of cardinality N is marked with indices i as described in (4.1).

$$X = \{ X_i \mid i \in [1..N] \} \tag{4.1}$$

$$X_i = \left[X_i^1 \; X_i^2 \; \ldots \; X_i^Q \right]$$

Clusters have Q-dimensional center points and the set of all clusters, Ψ contain K elements.

$$\Psi = \left\{ C_j \mid j \in [1..K] \right\} \tag{4.2}$$

$$C_j = \left[R_j^1 \; R_j^2 \ldots R_j^Q \right]$$

Algorithms assign data points to particular clusters by their innate comparison mechanisms. The following notation represents set of data points that a cluster has.

$$\Omega_j = \{X_i \mid X_i \; belongs \; to \; C_j \; and \; i \in [1..N]\} \; , \; where \; j \in [1..K]. \tag{4.3}$$

Clustering algorithms make heavy use of distance between two vectors to compare data points and clusters. Several distance measures exist, among which Euclidean, weighted Euclidean and a hybrid version of Pearson distance heuristics are covered in this study.

Euclidean distance between two Q-dimensional vectors is computed as follows.

$$d_E(V_i, Z_j) = \|V_i - Z_j\| = \sqrt{\sum_{q=1}^{Q} \left(V_i^q - Z_j^q\right)^2} \tag{4.4}$$

Squared Euclidean distance is computed as the square of Euclidean distance.

$$d_{SE}(V_i, Z_j) = \|V_i - Z_j\|^2 = \sum_{q=1}^{Q} \left(V_i^q - Z_j^q\right)^2 \tag{4.5}$$

Weighted Euclidean distance proposed in (Chicco et al., 2004) is computed subsequently.

$$Let \; V = \{V_i \mid i \in \mathbb{Z}^+\}. \tag{4.6}$$

$$d_{WE}(V_i, Z_j) = \sqrt{\sum_{q=1}^{Q} \frac{\sigma_q^2}{\bar{\sigma}^2} \left(V_i^q - Z_j^q\right)^2}$$

Feature variance, σ_q^2 in equation (4.6) is the variance of the q^{th} feature over all vectors residing in the set V and $\bar{\sigma}^2$ is the mean value of all feature variances. If two features of the two vectors are at the same Euclidean distance, yet one of them has higher variance, then the weighted Euclidean distance between the two vectors regarding the feature with high variance will be larger than the one with less variance. In short, features with higher variances are used as amplifying factors in distance computation.

Squared weighted Euclidean distance is simply the square of the weighted Euclidean distance as computed in the following.

$$d_{SWE}(V_i, Z_j) = \sum_{q=1}^{Q} \frac{\sigma_q^2}{\bar{\sigma}^2} \left(V_i^q - Z_j^q\right)^2 \tag{4.7}$$

Hybrid Pearson distance combines the squared Euclidean distance and an adaptive tuning function taking Pearson correlation as a parameter by a simple multiplication. This measure is inspired by the measure developed in (Chouakria-Douzal and Nagabushan, 2007) for classifying time series.

$$corr(V_i, Z_j) = \frac{\sum_{q=1}^{Q}(V_i^q - \bar{V}_i)(Z_j^q - \bar{Z}_j)}{\sqrt{\sum_{q=1}^{Q}(V_i^q - \bar{V}_i)^2 \sum_{q=1}^{Q}(Z_j^q - \bar{Z}_j)^2}} \quad (4.8)$$

$$d_{HP}(V_i, Z_j) = \frac{2}{1 + e^{k\, corr(V_i, Z_j)}}\, d_{SE}(V_i, Z_j) \;\; , \;\; where\ k \geq 0.$$

In case of data vectors with distinct shapes, or uncorrelated data, distance measure reduces to squared Euclidean distance. However, when vectors are correlated value of the distance measure decreases and in case of negative correlation measured distance increases, assuming that all two vectors compared are of the same squared Euclidean distance. Parameter k in equation (4.8) controls the effect of correlation on the distance measure, which increases as the value of this parameter increases.

Distance between two vectors is computed as follows, depending on the adopted distance framework.

$$d(V_i, Z_j) = \begin{cases} d_E(V_i, Z_j) \;, & for\ Euclidean\ framework. \\ d_{SE}(V_i, Z_j) \;, & for\ squared\ Euclidean\ framework. \\ d_{WE}(V_i, Z_j) \;, & for\ weighted\ Euclidean\ framework. \\ d_{SWE}(V_i, Z_j) \;, & for\ squared\ weighted\ Euclidean\ framework. \\ d_{HP}(V_i, Z_j) \;, & for\ hybrid\ Pearson\ framework. \end{cases} \quad (4.9)$$

Distance formula in equation (4.9) makes a selection among Euclidean distance in (4.4), squared Euclidean distance in (4.5), weighted Euclidean distance in (4.6), squared weighted Euclidean distance in (4.7), and hybrid Pearson distance in (4.8) on the basis of the data analyst's choice of distance framework. Parameters of the distance function, V_i and Z_j are substituted by actual data point features or cluster centers during the course of clustering.

Clustering operation outputs final set of data points belonging to a cluster, Ω_j for all clusters and final cluster representative centers R_j computed over several iterations until convergence. Algorithmic steps differ among clustering methods, which are described throughout the chapter.

4.3 Description of Algorithms

Mechanisms of the algorithms are highlighted in this section. K-means, K-means++, K-medians, WFA-K-means, Hopfield-K-means, K-medoids, Similarity-based K-means, ISODATA, Fuzzy K-means, Follow-the-Leader, Hierarchical clustering, and Self-Organizing Map clustering are covered respectively. Notations used in this chapter are included in the List of Notations.

4.3.1 K-Means Clustering

K-means is an unsupervised partitional classification algorithm, which requires the exact information of number of clusters in order to operate. In thesis work, K-means algorithm is employed in its classical form with mere modifications regarding the computation of initial cluster centroids. The steps of the algorithm are as follows:

- **Step 1**: Set up K centroids initially. In order to deterministically set up cluster centers, method described in (Tsekouras et al., 2008) is used so that several re-runs of the algorithm due to random initialization can be avoided. The formula in equation (4.10) initializes cluster centers.

  $$R_j^q = c_1 + c_2\,(j-1)\,\frac{1}{K-1}\,,\quad \forall q \in [1\,..\,Q]\ and\ \forall j \in [1\,..\,K] \tag{4.10}$$

  The values of $c_1$and c_2 are determined experimentally by running and evaluating the algorithm several times on a particular dataset. Mark all clusters as updated.

- **Step 2:** Compute distance matrix. If this step is to be performed for the first time, construct an N-by-K matrix. For each updated cluster, compute the corresponding entry in the distance matrix, as indicated below.

  $$if\ C_j\ is\ updated,\ d_{ij} = d\big(X_i, C_j\big)\,,\ \forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K] \tag{4.11}$$

  All clusters are considered as not updated following distance matrix computation unless this is the first iteration, which requires all clusters to be marked as updated.

- **Step 3:** Make assignments. Distances of each data point to cluster centers are calculated by using the corresponding distance metric formula. Then, data points are assigned to their closest clusters by traversing the distance matrix.

  $$C_{j*} = \underset{C_j}{\operatorname{argmin}}\,\{d\big(X_i, C_j\big)\} \tag{4.12}$$

  $Let\ C_{j'}\ be\ the\ cluster\ that\ X_i\ formerly\ belongs\ to. If\ C_{j*}\ is\ not\ equal\ to\ C_{j'}$;

  $(i)\quad \Omega_{j*} = \Omega_{j*} \cup \{X_i\}\ and$
  $(ii)\quad \Omega_{j'} = \Omega_{j'} - \{X_i\}\ and$
  $(iii)\quad t_z = t_z + 1\ and$
  $(iv)\quad C_{j*} and\ C_j' are\ marked\ as\ updated,$

  $\forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K]$.

If a data point is assigned to another cluster than its previous one, then both clusters are marked as updated. Assign number of cluster-changing data points to the variable, t_z.

- **Step 4:** Compute cluster centers. Traditionally, the centroid of a cluster is computed by taking average of each feature of every data point belonging to it as it is indicated in equation (2.4).

$$R_j^q = \frac{1}{N_j}\sum_{X_i \in \Omega j} X_i^q \ , \ \forall i \in [1 \,..\, N] \,, \forall j \in [1 \,..\, K], and \ \forall q \in [1 \,..\, Q]. \qquad (4.13)$$

- **Step 5**: If no data points have changed clusters, namely t_z is equal to 0 or maximum number of iterations, I_{max} is reached, then stop. Else, increment the current iteration number, I and continue with **Step 2**.

K-means clustering provides the basis for all partitional clustering algorithms. Subsequent algorithms examined in the thesis modify particular steps of the K-means algorithm, yet keeping the main mechanisms and work flow quite the same. The modification to the original K-means clustering employed in this section is the deterministic choice of initial cluster centers, which turns out to be an improvement in the quality of clustering, as further discussed by the evaluation section.

4.3.2 K-Means++ Clustering

K-means++ clustering algorithm (Arthur and Vassilvitskii, 2007) is a modified version of the classical K-means algorithm, offering a solution to cluster centers initialization problem. Although randomness is still included in that method, it could be controlled better because centroid initialization has been put into a direct relation with inter-point distances. Furthermore, a more accurate and timesaving clustering is provided by the algorithm in most cases.

K-means++ method follows the same steps as K-means algorithm except for the sole modification in the centroid initialization step as follows.

- **Step 1**: Set up K centroids initially.
    - **Step 1.1**: Select a random number i uniformly from the interval $[1 \,..\, N]$.
    - **Step 1.2**: Set $R_1 = X_i$ and set $M = 1$.
    - **Step 1.3**: Find the minimum distance between each data point and already processed clusters and store the accumulated squared value of this distance in an array as described in the following equation. Set $sum = 0$ before every time this computation is performed.

$$sum = sum + (\min \{d(X_i, C_m)\})^2 \quad (4.14)$$

$$minDist[i] = sum, \qquad \forall i \in [1..N] \; and \; \forall m \in [1..M]$$

- **Step 1.4**: Select another data point as the center of the next cluster. In order to achieve that, generated random number shall be multiplied with the distance weights computed in the previous step. Computations regarding this step are included in the following.

$$Generate\ a\ random\ real\ number\ r \quad (4.15)$$
$$uniformly\ from\ the\ interval\ [0, 1].$$

$$r = r * sum$$

$$\exists i \in [1..N],\ if\ minDist[i] \geq r\ then\ R_S = X_i and\ M = M + 1\ .$$

- **Step 1.5**: If M becomes K, then cluster center initialization step is successfully terminated. Else continue with **Step 1.3**.

One disadvantage regarding the algorithm is that, several re-runs may be necessary in order to obtain the clustering that has the best performance quality with respect to others due to randomness factor. However, K-means++ clustering algorithm yields better clustering than the classical K-means algorithm with respect to several performance indices discussed thoroughly in the next chapter of the thesis.

4.3.3 K-Medians Clustering

K-medians clustering algorithm (Anderson et al., 2006) follows the same steps as K-means++ clustering with a sole modification in the computation of cluster centers. Centroids are chosen to be combinations of feature-based median values of the data points that the clusters have. The modified step is described subsequently.

- **Step 4:** Compute cluster centers using the median operator, as follows.

$$Sort\ elements\ of\ \Omega_j\ in\ ascending\ order\ based\ on\ feature\ q. \quad (4.16)$$

$$If\ N_j\ is\ even\ then\ R_j^q = \frac{\Omega_j^m + \Omega_j^{m+1}}{2}\ where\ m = \left\lfloor \frac{N_j}{2} \right\rfloor,$$

$$If\ N_j\ is\ odd\ then\ R_j^q = \frac{\Omega_j^{m+1}}{2}\ where\ m = \left\lfloor \frac{N_j}{2} \right\rfloor,$$

$$\forall i \in [1..N]\ , \forall j \in [1..K], and\ \forall q \in [1..Q].$$

Effect of the outlier values is diminished in cluster center computation by utilizing median operator instead of the statistical mean operator.

4.3.4 WFA K-Means Clustering

Weighted fuzzy average (WFA) K-means clustering algorithm (Mahmoudi-Kohan et al., 2008) follows the same steps as K-means++ clustering with only modifying the centroid update step by computing the weighted fuzzy average of data points belonging to a cluster to be its center instead of taking feature-by-features averages of the points. Cluster center computation of WFA K-means algorithm is described in the following.

- **Step 4:** Compute cluster centers by the WFA operator included in equation (4.17), assuming that σ_{WFA} is initially 0 as in (Looney, 2002).

$$\sigma_{WFA} = \sigma_{WFA} + d(C_j, \Omega_j),\ \forall j \in [1..K] \tag{4.17}$$

$$R_j^q = \frac{\sum_{X_{i^*} \in \Omega_j} X_{i*}^q \, e^{\frac{X_{i*}^q - \frac{1}{N_j}\sum_{X_{i^*} \in \Omega_j} X_{i*}^q}{-2\sigma_{WFA}}}}{\sum_{X_{i^*} \in \Omega_j} e^{\frac{X_{i*}^q - \frac{1}{N_j}\sum_{X_{i^*} \in \Omega_j} X_{i*}^q}{-2\sigma_{WFA}}}} \quad , \quad \forall j \in [1..K], and\ \forall q \in [1..Q].$$

By using weighted fuzzy average instead of the mean operator, more robust clustering results are expected to be obtained, since weighted fuzzy average diminishes the effect of outliers on cluster centers. However, computation costs associated to centroid update sub-task will increase due to the complexity of the WFA operator.

4.3.5 Hopfield-K-Means Clustering

Hopfield K-means clustering algorithm (López et al., 2011) uses a Hopfield network in the computation of the initial cluster centers. It employs K-means++ centroid initialization algorithm to establish the initial state vector. Although randomness is still a factor at the state initialization step of the neural network training, states are expected to converge eventually when the minimum energy configuration is obtained (Takefuji et al., 1992). Then, the centroid initialization shall be performed.

Hopfield artificial neural networks contain nodes that are connected to all others, forming a complete graph. The algorithm follows the same steps as classical K-means except the change in the first step that computes initial centroids. Corresponding modification is described in the following.

- **Step 1**: Set up K centroids initially.

- **Step 1.1**: Build a Hopfield network consisting of N neurons, each of which corresponds to a data point. Neurons have states, which are characterized by output functions that return the indices of clusters the data points are assigned to. A state vector pertaining at a particular instance of the training process characterizes the state of the system. State vector is initialized randomly by using K-means++ centroid initialization algorithm, prior to the first iteration of training.

$$S^{(t)} = [s_i^{(t)}] \quad (4.18)$$
$$where\ s_i^{(t)} \in [1\,..K]\,,\quad \forall i \in [1\,..N]\ and\ \forall t \in [1\,..T].$$

$$Let\ \omega = \left\{ S^{(t)} \middle|\ t \in [1\,..T] \right\}\ be\ the\ set\ of\ all\ state\ vectors.$$

$$\omega = \emptyset\ initally$$

$$Use\ K-means++\ initialization\ result\ r \in [1..K]\ such\ that$$
$$s_i^{(0)} = r\ \ if\ \ X_i \in C_r\ ,\ \forall i \in [1\,..N].$$

$$\omega = \omega \cup \left\{S^{(0)}\right\}\ and\ t = 1.$$

- **Step 1.2**: Every neuron is connected to each other by means of weight functions. Convergence of the training step is controlled by making use of an energy function, which takes the current state vector as its argument and measures the degree of similarity of states based on inter-data point distances. Energy function requires the knowledge of similarity between states, which is attained through utilizing the simple function included below.

$$W = \left[w_{ij}\right]\ where\ w_{ij} = distance\left(X_i, X_j\right),\ \ \forall i\ \forall j \in [1\,..N]. \quad (4.19)$$

$$f\left(s_i^{(t)}, s_j^{(t)}\right) = \begin{cases} 1\ , if\ s_i^{(t)} equals\ s_j^{(t)} \\ 0\ , otherwise \end{cases}$$

$$E\left(S^{(t)}\right) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\, f\left(s_i^{(t)}, s_j^{(t)}\right)$$

- **Step 1.3**: Change the state at the position implied by the training iteration number. A neuron can change its state if the corresponding change implies a decrease in the energy function. Compute the new state vector by reflecting the performed changes. If none of the states has been updated, then the new state vector is unchanged.

$$Set\ s_{i'}^{(t+1)} = k,\ if\ \exists k\ E\left(S^{(t+1)}\right) \leq E\left(S^{(t)}\right)\ and\ N_{S_{i'}^{(t)}} \geq 1\ , \qquad (4.20)$$

$$Set\ s_{i'}^{(t+1)} = s_{i'}^{(t)},\ otherwise,$$

$$where\ i' = (T\ mod\ K) + 1\ and\ k \in [1\,..\,K].$$

$$Compute\ S^{(t+1)} = \left[s_i^{(t+1)}\right]\ such\ that\ s_i^{(t+1)} = s_i^{(t)}\ ,$$

$$\forall i \in [1\,..\,N] - \{i'\}.$$

$$\omega = \omega \cup \left\{S^{(t+1)}\right\}\ and\ t = t + 1.$$

- **Step 1.4**: If maximum training iteration number has not been reached, continue with **Step 1.3**. Else, continue with **Step 1.5**.

- **Step 1.5**: Find the state vector with minimum energy. For each constituent states in the state vector with minimum energy, assign the corresponding data point to the cluster whose index is the same as the output value of the constituent. Then, update the cluster centers in accordance with the traditional K-means cluster centroid update rule. The processes regarding this step are included in the following.

$$S^{(t')} = \underset{S^{(t)} \in \omega}{\operatorname{argmin}} \left\{E\left(S^{(t)}\right)\right\}\ where\ t' \in [1\,..\,T] \qquad (4.21)$$

$$If\ s_i^{(t')} = j\ \ then$$

$$(i)\quad X_i\ belongs\ to\ C_j,$$
$$(ii)\quad \Omega_j = \Omega_j \cup \{X_i\}, and$$
$$(iii)\quad N_j = N_j + 1; \qquad \forall i \in [1..N], j \in [1..K]$$

$$R_j^q = \frac{1}{N_j} \sum_{X_i \in \Omega j} X_i^q\ , \quad \forall i \in [1\,..\,N]\ , \forall j \in [1\,..\,K], and\ \forall q \in [1\,..\,Q].$$

The training of the Hopfield network is terminated at the completion of the above steps. Energy function of the Hopfield network decreases steadily over iterations until it reaches a local optimum that corresponds to the solution of the centroid initialization problem. Maximum number of iterations regarding the training should be set reasonably, so that convergence of the energy values can be observed. Having set the initial cluster centers, algorithm continues with the same steps as the classical K-means until the final clustering of data points is attained.

4.3.6 K-Medoids Clustering

K-medoids clustering selects the most centrally located data points within clusters as cluster centers called *medoids*. The algorithm employed is based on *Partitioning Around Medoids (PAM)* method (Kaufman and Rousseeuw, 1987) with modifications. Initial medoids are computed by K-means++ centroid initialization algorithm. Cluster medoids are updated by choosing the most centric data point in then-currently existing cluster, instead of computing a swapping cost for each data point and crosschecking as proposed by the original algorithm.

Main progress of the algorithm is described below.

- **Step 1**: K-means++ centroid initialization algorithm is run. Data points selected as cluster centers by this method are marked as medoids and are added to corresponding clusters.

  $$if\ X_i equals\ R_j\ ,\ then \quad (4.22)$$
  $$X_i is\ marked\ as\ a\ medoid\ and$$
  $$\Omega_j = \Omega_j \cup \{X_i\}\ \ ;\ \ \forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K].$$

- **Step 2:** Compute the distance matrix. This step is the same as in the K-means clustering algorithm.

  $$if\ C_j\ is\ updated, d_{ij} = d\left(X_i, C_j\right)\ ,\ \forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K]. \quad (4.23)$$

- **Step 3:** Make assignments. This step is the same as in the K-means method except the following modification asserting that data points that are marked as medoids cannot change clusters in the assignment step. Assignment of non-medoid data points to clusters is carried out by searching the minimum distance between clusters and data points, as in K-means clustering.

  $$If\ X_i is\ a\ medoid, then\ continue\ with\ the\ next\ data\ point. \quad (4.24)$$

- **Step 4:** Update cluster medoids. A new medoid is chosen if a non-medoid data point within a cluster is more centrally located than the current medoid according to the following formula.

$$\bar{d}(X_i) = \sum_{X_{i'} \in \Omega_j} d(X_i, X_{i'}) \;\; such\ that\ X_i \in \Omega_j \,, \quad (4.25)$$
$$where\ i \in [1\,..\,N] and\ j \in [1\,..\,K].$$

$$Set\ R_j = X_{i'}\,,$$
$$if\ \exists X_{i'} \left(X_{i'}\ is\ not\ a\ medoid\ and\ X_{i'} \in \Omega_j \;\; and\ X_{i'} = \underset{X_i \in \Omega_j}{\operatorname{argmin}} \left\{ \bar{d}(X_i) \right\} \right),$$
$$for\ all\ j \in [1\,..\,K].$$

- **Step 5**: If no data points have changed clusters, namely t_z is equal to 0 or maximum number of iterations, I_{max} is reached, then stop. Else, increment the current iteration number and continue with **Step 2**.

Medoid initialization method inherited from K-means++ contributes to reduce the running time of the algorithm and to yield high-quality partitioning of the data set by carefully selecting initial cluster centers. Employing medoids instead of centroids in the algorithm, improves the quality of generated clusters, since medoids achieve better isolation of outliers and noise that may exist in the dataset.

4.3.7 Similarity-Based K-Means Clustering

Similarity-based K-means clustering (Bandyopadhyay and Saha, 2007) is a two-step algorithm. In the first-step or the *coarse-tuning phase*, K-means++ performs an initial clustering of data points. In the second-step or the *fine-tuning phase*, distance metric is changed as the point symmetry distance, which is not a mathematical metric yet a heuristic that aims at gathering data points close to each other within certain threshold limits by making use of the concept of symmetry in multidimensional space.

Definition of the point symmetry distance is included below, as outlined in (Su and Chou, 2001). In order to compute the distance, first the symmetric point of a data instance with respect to a cluster center should be computed. The symmetric point, which does not have to reside physically in the data set, is calculated as follows.

$$X_{i*}^q = 2R_j^q - X_i^q,\ \forall q \in [1\,..\,Q]\ \ where\ i \in [1\,..\,N]\ and\ j \in [1\,..\,K]. \quad (4.26)$$

Next, the two minimum squared Euclidean distances regarding the symmetric data point is computed, as in the following.

$$d_1 = \min\ \{\|X_{i*} - X_i\|^2\}, \quad where\ i \in [1\,..\,N]. \quad (4.27)$$

$$Let\ X_{i'} = \underset{X_i}{\operatorname{argmin}}\ \{\|X_{i*} - X_i\|^2\},\ \ where\ i \in [1\,..\,N].$$

$$d_2 = \min\ \{\|X_{i*} - X_i\|^2\}, \quad where\ i \in [1\,..\,N] - \{i'\}.$$

Computation proceeds with the actual formula of the point symmetry distance between a cluster and a data point, which combines the distance calculations regarding the symmetric point and the Euclidean distance between the cluster and the data point as in (4.28)

$$psd(X_i, C_j) = 0.5\,(d_1 + d_2)\,\|X_i - C_j\|^2, \quad where\ i \in [1..N]\ and\ j \in [1..K]. \quad (4.28)$$

Further clustering based on distance metric switch is performed in order to refine clustering results obtained at the end of the coarse-tuning. Therefore, further clustering is called as *fine tuning*. In fine tuning phase, firstly the distance metric is changed to the point symmetry heuristic and the algorithm follows the main steps of the K-means clustering except the centroid initialization.

- **Step 1:** Initial cluster centroids are inherited from the coarse tuning phase and all clusters are marked as updated so that distance matrix computation is carried out.

Cycles of assignment and update steps are the same as the ones in the original K-means clustering except a sole modification in the assignment step. This modification ensures that a data point may change its cluster only if the computed point-symmetry distance is below a certain threshold value θ_{SMK}. The threshold value can be specified by the data analyst or can be computed automatically according to the following heuristics.

$$\theta_{SMK} = \max\left\{\left\| X_i - \left(\operatorname{argmin}_{X_j}\left\{\|X_i - X_j\|^2\right\}\right)\right\|^2\right\}, \quad (4.29)$$

$\forall i \in [1..N]\ and\ \forall j \in [1..N]$.

Hence, the threshold is set to the maximum nearest neighborhood distance regarding all points in the set unless its value is explicitly determined. This rule of thumbs contributes greatly in reducing the time the analyst has to spend on running the algorithm, since it offers the chance of deciding the value automatically for any number of clusters to be generated.

4.3.8 ISODATA Clustering

Iterative Self-Organizing Data Analysis Technique (ISODATA) is a clustering method based on thresholds of minimum cluster size, maximum standard deviation within a cluster, minimum inter-cluster distance, desired number of clusters to be obtained by the end of the operation, and maximum number of merge operations that could be performed during clustering (Ball and Hall, 1965). Yet, ISODATA does not require the exact number of clusters to function. The algorithm operates by merging or splitting clusters and re-distributing data points among clusters depending on thresholds.

ISODATA makes heavy use of the notion of average distance within a cluster, which is included in equation (4.30).

$$\bar{d}(C_j) = \frac{1}{N_j} \sum_{X_i \in \Omega_j} d(X_i, C_j), \quad where\ i \in [1\,..\,N]\ and\ j \in [1\,..\,K]. \tag{4.30}$$

System-wide average distance is calculated by employing the average distance within clusters. It is a weighted average function so that large clusters will have more impact on the value of the total average distance as put forward in (4.31).This distance is a parameter for determining clusters that are less compact than others to decide the next action regarding them.

$$avgDist = \frac{1}{N} \sum_{j=1}^{K} N_j\ \bar{d}(C_j) \tag{4.31}$$

Intra-cluster deviation is another important control parameter of ISODATA. It measures distance between the cluster center and the data points belonging to the cluster feature-by-feature as computed in the following.

$$\sigma_j^q = \sqrt{\frac{1}{N_j} \sum_{X_i \in \Omega_j} \left(R_j^q - X_i^q\right)^2}\ , \tag{4.32}$$

$\forall q \in [1\,..\,Q],\ where\ i \in [1\,..\,N]\ and\ j \in [1\,..\,K]$.

The largest standard deviation among all clusters and the index of the feature with the largest deviation is computed and stored for use by the algorithm, as indicated in equation in (4.33).

$$\sigma_j = \max_q\left\{\sigma_j^q\right\}, and\ ind_j = \operatorname{argmax}_q\left\{\sigma_j^q\right\}, \tag{4.33}$$

$where\ q \in [1\,..\,Q]\ and\ j \in [1\,..\,K]$.

ISODATA proceeds as described in the following steps.

- **Step 1**: Set initial centroids. Begin the algorithm with arbitrary number of clusters stored in the variable K. Compute initial cluster centers exactly as in K-means++ clustering.

- **Step 2**: Make assignments. Each data point is assigned to a cluster that is closest to it in terms of calculated distances. This step is the same as **Step 3** of K-means clustering.

- **Step 3**: Check cluster sizes. If a cluster has less elements than minimum cluster size, then remove this cluster, decrement number of clusters and assign its data points to the closest one of the remaining clusters by going back to **Step 2**.

- **Step 4**: Update cluster centroids. Set the center of each cluster to the mean of the data points belonging to it. This step is the same as **Step 4** of the K-means clustering.

- **Step 5**: Compute system-wide average distance as defined by the equation (4.31).

- **Step 6**: If this is the last iteration, set minimum inter-cluster distance to 0. Proceed with merge operation described in **Step 9**.

- **Step 7**: If current number of clusters is less than or equal to half of the desired number of clusters, then proceed with the split operation described in **Step 10**.

- **Step 8**: If current number of clusters is greater than or equal to twice as the desired number of clusters and current iteration number is even, then proceed with the merge operation described in **Step 9**.

- **Step 9**: Merge clusters.

    - **Step 9.1**: Check each pair of clusters and add the pairs that can be merged into a list as described in the following. If there are no pairs of clusters to be merged, continue with **Step 11**.

      $$If\ d\left(C_{j_1}, C_{j_2}\right) \leq\ \theta_c\ and\ \left\{\langle C_{j_1}, C_{j_2}\rangle\right\} \notin L\ then \tag{4.34}$$

      $$L = L\ \cup \left\{\langle C_{j_1}, C_{j_2}\rangle\right\}\ ,\ \ \forall j_1 \in [1\,..\,K]\ \ \forall j_2 \in [(j_1+1)\,..\,K].$$

    - **Step 9.2**: Sort the list of the elements to be merged in a descending order based on distance between cluster pairs. If number of pairs in the list exceeds the maximum number of merges, then re-organize the list so that it will contain only the number of elements up to the merge threshold.

      $$Sort\ L\ in\ a\ descending\ order\ of\ d\left(C_{j_1},\ C_{j_2}\right) \tag{4.35}$$
      $$for\ all\ \langle C_{j_1}, C_{j_2}\rangle \in L.$$

      $$If\ |L| > \theta_m\ then\ L = \bigcup_{s=1}^{\theta_m} L_s\ .$$

    - **Step 9.3**: Merge each pair of clusters in the list. For each pair, create a new cluster having its centroid as the weighted average of the centroids

of the two clusters. Then remove the old clusters and adjust the number of clusters accordingly.

$$Let\ C_{j*}\ be\ the\ new\ cluster. \tag{4.36}$$

$$R_{j*}^{q} = \frac{1}{N_{j_1}+N_{j_2}} \left(N_{j_1} R_{j_1}^{q} + N_{j_2} R_{j_2}^{q} \right) , \forall q \in [1 \,..\, Q] ;$$

$$Remove\ C_{j_1}\ and\ C_{j_2} ;$$

$$K = K - 1 , \quad \forall \langle C_{j_1}, C_{j_2} \rangle \in L.$$

- **Step 9.4**: Continue with **Step 2**.

- **Step 10**: Split clusters.

  - **Step 10.1**: Check each cluster whether it fulfills the preconditions of the split operation. Add splitable clusters to a list. If no cluster can be split, continue with **Step 11**.

$$If\ \sigma_j > \theta_s\ and \tag{4.37}$$
$$\left(\left(\bar{d}(C_j) > avgDist\ and\ N_j > 2\theta_N \right) or\ K \leq \frac{K'}{2} \right)$$
$$then\ L = L \cup \{C_j\} \ , \ \forall j \in [1 \,..\, K].$$

  - **Step 10.2**: Split each cluster in the list into two new clusters. Compute the features of the centroids of the new clusters to be the same on each two clusters as the split cluster, except the feature with the highest standard deviation. This feature has two different values in the new centroids as the following description indicates. Remove the split cluster and adjust the number of clusters accordingly.

$$Let\ C_{j_1} and\ C_{j_2}\ be\ the\ new\ clusters. \tag{4.38}$$

$$R_{j_1}^{ind_j} = R_j^{ind_j} + 0.5\ \sigma_j\ \sigma_j \quad and \quad R_{j_2}^{ind_j} = R_j^{ind_j} - 0.5\ \sigma_j\ \sigma_j ;$$

$$R_{j_1}^{q} = R_j^{q} \ \ and \ \ R_{j_2}^{q} = R_j^{q} \quad \forall q \in [1 \,..\, Q] - \{ind_j\} ;$$

$$Remove\ C_j ;$$

$$K = K + 1 ; \ \forall C_j \in L.$$

  - **Step 10.3**: Continue with **Step 2.**

- **Step 11**: If no data points have changed clusters, namely t_q is equal to zero, and no cluster has been removed due to having insufficient number of elements; or maximum number of iterations, I_{max} is reached then stop. Else, increment the current iteration number, I and continue with **Step 2**.

ISODATA method efficiently groups data points into clusters, detects outliers, and is robust to noise given that parameters are carefully selected. Since the algorithm requires several parameters, it is data analyst's responsibility to adjust them to obtain the desired clustering configuration. Although this seems to be a disadvantage at the first glance, one could take advantage of this situation by manipulating the mechanisms of the algorithm, since many trade-offs that the partition of the data exhibits can be analyzed through the parameters.

4.3.9 Fuzzy K-Means Clustering

Fuzzy K-means clustering imposes a fuzzy partition of the data set in which a data point belongs to a cluster by a degree computed by the algorithm (Cannon et al., 1986). Likewise, cluster centroids are calculated by taking averages of all data points weighted by their degree of belonging to clusters. This approach is also called *soft clustering*, as an antagonist analogy to all clustering algorithms that have been covered so far, since they are examples of *hard clustering*, which imposes that a data point belongs to a particular cluster and does not belong to any other.

The algorithm follows the steps of the K-means clustering with modifications in distance and cluster representative vector computation. Fuzzy K-means utilized in the thesis is slightly modified so that instead of random initialization of the degree matrix. First, K-means++ cluster centroid initialization algorithm is run and then degree vector is computed.

Fuzzy K-means is controlled by the minimization of the objective function defined in the following equation.

$$J = \sum_{i=1}^{N}\sum_{j=1}^{K}\left(u_{ij}\right)^{m} d(X_i, X_j) \tag{4.39}$$

Degree of belonging of a data point to a cluster is computed by making use of fuzziness factor, which is a parameter of the algorithm, as indicated by the following equation.

$$u\left(X_i, C_j\right) = \frac{1}{\sum_{j'=1}^{K}\left(\frac{d\left(X_i, C_j\right)}{d\left(X_i, C_{j'}\right)}\right)^{\frac{2}{m-1}}}\ , \tag{4.40}$$

$where\ i \in [1\,..\,N], j \in [1\,..\,K], and\ 1 < m < \infty.$

The algorithm proceeds by following the steps included subsequently.

- **Step 1**: Initial steps.
    - **Step 1.1**: Run K-means++ initial centroid computation algorithm and compute initial cluster centers.
    - **Step 1.2**: Compute the degree matrix based on the formula in (4.40) by taking into account the two special cases introduced in the following.

$$if\ u(X_i, C_j)\ equals \infty\ then\ set\ u(X_i, C_j) = 0.9999\ , \tag{4.41}$$

$$if\ u(X_i, C_j)\ equals\ 0\ then\ set\ u(X_i, C_j) = \frac{1-0.9999}{K-1}$$

This modification is necessary since utilizing K-means++ algorithm in initial centroid computation implies that certain data points shall be chosen as cluster centers. Therefore, for these data points, the degree of belonging to the clusters, whose centroid set by their coordinates will be infinity, and the degree they belong to others will be zero, as calculated by (4.40). This is a direct side effect of using a hard clustering algorithm as part of a fuzzy clustering algorithm. In order to prevent that, above modification is put in use. Next, the degree matrix is constructed and computed as described below.

$$Construct\ an\ N-by-K\ matrix, called\ U = [u_{ij}]. \tag{4.42}$$

$$u_{ij} = u(X_i, C_j),\ \forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K].$$

- **Step 2:** Compute cluster centroids. Use an average of features of all data points weighted by their degree of belonging while computing the representative center of a cluster, as follows.

$$R_j^q = \frac{\sum_{i=1}^{N} X_i^q (u_{ij})^m}{\sum_{i=1}^{N} (u_{ij})^m} \quad ,\ \forall q \in [1\,..\,Q], \forall j \in [1\,..\,K]. \tag{4.43}$$

- **Step 3:** Update the degree of belonging matrix by utilizing the original formula in (4.42).

$$u_{ij} = u(X_i, C_j),\ \forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K]. \tag{4.44}$$

- **Step 4**: Compute change in the objective function. If the change goes below a predefined threshold that should be small enough to ensure a local optimum solution; or the value of the objective value starts to increase; or maximum

number of iterations is met, then exit the loop and proceed with **Step 5**. Else, increment the iteration number and go to **Step 2**.

$$\Delta J = J^{(t)} - J^{(t+1)}\ ,\ where\ J^{(t+1)} = \sum_{i=1}^{N}\sum_{j=1}^{K}\left(u_{ij}\right)^{m} d\left(X_i, X_j\right). \quad (4.45)$$

$$If\ \Delta J < \varepsilon\ \ or\ \Delta J < 0$$
$$then\ a\ local\ optimum\ state\ is\ reached.\ Exit\ the\ loop.$$

- **Step 5:** Make assignments. A data point is assigned to a cluster to which it belongs with the strongest degree.

$$If\ j^{'} = \underset{j}{\operatorname{argmax}}\left\{u_{ij}\right\},\ then\ X_i\ belongs\ to\ C_{j'}\ \ and\ \ \Omega_{j'} = \Omega_{j'} \cup \{X_i\}\,; \quad (4.46)$$
$$\forall i \in [1\,..\,N]\ and\ \forall j \in [1\,..\,K].$$

Fuzzy K-means clustering performs the soft partition of the data set by the steps described above. It is more time-consuming than hard k-means clustering algorithms, since it is mandatory to update the computationally expensive degree matrix at each cycle of the loop, regardless of whether a cluster is updated or not. Furthermore, its convergence takes longer whenever the value of the fuzziness factor is decreased. Yet, it does not require assignment of data points while performing the main steps of the algorithm, which could be regarded as a positive factor reducing the total computational cost.

4.3.10 Follow-The-Leader Clustering

Follow-the-Leader clustering (Chicco et al., 2004) relies on a distance threshold and performs iteratively on a data set by incrementing clusters and refining them based on the threshold and the distance between data points and clusters. The algorithm does not require exact number of clusters, yet solely has the threshold as its parameter.

The steps of the algorithm are covered in the following.

- **Step 1**: Set the centroid of the first cluster to the features of the first data point in the data set and add the data point to the cluster. Set the number of clusters to 1, number of changes over clusters to 1, and the number of iterations to 0.

$$R_1^q = X_1^q\ ,\quad \forall q \in [1\,..\,Q]. \quad (4.47)$$
$$\Omega_1 = \{X_1\}.$$
$$K = 1.$$
$$changeOverCount = 1.$$
$$I = 0.$$
$$next = 2.$$

- **Step 2**: Take action regarding the next data point as indicated in subsequent steps.

    - **Step 2.1**: Check the distance of the next data point from the existing cluster centers and identify the cluster at minimum distance.

      $$minDist(X_{next}) = \min\{d(X_{next}, C_j)\},\ and \quad (4.48)$$
      $$C_{j'} = \text{argmin}_{C_j}\{d(X_{next}, C_j)\},\quad where\ j \in [1\,..\,K].$$

    - **Step 2.2**: If the iteration number is zero and computed minimum distance exceeds the threshold, then create a new cluster for the data point, set the coordinates of the centroid to its features, and add it to the cluster. Continue with **Step 2.7**.

      $$If\ I\ equals\ 0\ and\ minDist(X_{next}) > \rho\ \ then \quad (4.49)$$

      $$R_{K+1}^{q} = X_{next}^{q}\ \ ,\ \ \forall q \in [1\,..\,Q]\,,$$
      $$\Omega_{K+1} = \{X_{next}\}\,,$$
      $$K = K + 1\,, and$$
      $$changeOverCount = changeOverCount + 1.$$

    - **Step 2.3**: If the iteration number is zero and computed minimum distance is less than or equal to the threshold, then add the data point to the cluster that is closer to it. Update the cluster centroid afterwards. Continue with **Step 2.7**.

      $$If\ I\ equals\ 0\ and\ minDist(X_{next}) \leq \rho\ \ then \quad (4.50)$$

      $$\Omega_{j'} = \Omega_{j'} \cup \{X_{next}\}\,,$$
      $$R_{j'}^{q} = \frac{1}{N_{j'}} \sum_{X_i \in \Omega_{j'}} X_i^q \quad , \forall d \in [1\,..\,Q].$$

    - **Step 2.4**: If the iteration number is not zero, further cluster creation is allowed in successive cycles, and minimum distance is greater than the threshold; then create a new cluster for the data point, set the center coordinates to the features of the data point, and add it to the cluster. Continue with **Step 2.7**.

$If\ I > 0\ and$ (4.51)
$further\ cluster\ generation\ is\ allowed\ and$
$minDist(X_{next}) > \rho\ ,\ \ then$

$R_{K+1}^{q} = X_{next}^{q}\ \ ,\ \ \forall q \in [1\,..\,Q]\,;$
$\Omega_{K+1} = \{X_{next}\}\,;$
$K = K + 1\,;\ and$
$changeOverCount = changeOverCount + 1.$

- **Step 2.5**: If the iteration number is not zero, minimum distance is not greater than the threshold, and the closest cluster to the data point is its current cluster, then do nothing and continue with **Step 2.7**.

- **Step 2.6**: If the iteration number is not zero, minimum distance is not greater than the threshold, and the closest cluster to the data point is not the cluster that it already belongs to, then remove the data point from its current cluster and add it to the closest cluster. Update both cluster centers. Continue with **Step 2.7**.

  $Let\ C_{j*}\ be\ the\ cluster\ that\ X_{next}\ formerly\ belongs\ to.$ (4.52)

  $If\ C_{j'}\ is\ not\ equal\ to\ C_{j*}\ then$

  $(i)\quad \Omega_{j'} = \Omega_{j'} \cup \{X_{next}\}\ and$
  $(ii)\quad \Omega_{j*} = \Omega_{j*} - \{X_{next}\}\ and$
  $(iii)\quad R_{j'}^{q} = \dfrac{1}{N_{j'}} \displaystyle\sum_{X_i \in \Omega_{j'}} X_i^q \quad \forall q \in [1\,..\,Q]\ and$
  $(iv)\quad R_{j*}^{q} = \dfrac{1}{N_{j*}} \displaystyle\sum_{X_i \in \Omega_{j*}} X_i^q \quad \forall q \in [1\,..\,Q]\ and$
  $(v)\quad changeOverCount = changeOverCount + 1\ .$

- **Step 2.7**: The processing of the data point has been completed. Increment the subject number and continue with **Step 2.1**.

  $next = next + 1.$ (4.53)

- **Step 3**: If all data points are covered, and maximum number of iterations is reached or no cluster change has occurred then exit the loop and continue with **Step 5**. Else go to **Step 4**.

- **Step 4**: If all data points are covered but neither the maximum number of iterations is reached nor any of the data points have changed clusters, then reset

the number of data points and number of changeovers and increment the number of iterations. Continue with **Step 2**.

$$If\ next > N\ and\ I \neq I_{max}\ and\ changeOverCount \neq 0\ then \tag{4.54}$$

$$next = 1\ ,\ \ changeOverCount = 0,\ \ I = I + 1\ .$$

- **Step 5**: The algorithm terminates.

Number of clusters is set in the first iteration of the algorithm by going over every point in the set. This process depends on the order of the data points. Yet, in successive cycles, data points may change clusters and if cluster creation is allowed in successive cycles, points may form new clusters on condition that they are more-than-threshold-distance far away from all existing clusters.

Follow-the-Leader is a computationally efficient, fast converging, and parametrically simple clustering algorithm. It manages outlier detection and is robust to noise. The mere disadvantage of the algorithm for the data analyst is deciding the value of the threshold, since setting a distance threshold for separating points in a multi-dimensional space is not a subtle issue. However, once the threshold is set to a suitable value, the analyst can easily control the quality of clustering based on parameters such as the number of clusters by manipulating the threshold.

4.3.11 Hierarchical Clustering

Hierarchical clustering imposes a top-down or bottom-up ordering of data points based on inter-cluster distance (Chicco et al., 2003). Within the scope of this thesis, a bottom-up (agglomerate) approach is adopted, in which all data points form standalone clusters by themselves in the beginning and as the algorithm progresses, most similar clusters are linked until all of the data points are eventually organized in a single cluster.

The method itself is a deterministic way of distributing data points among clusters, unlike the partitional cluster algorithms examined up to this point. Furthermore, it does not rely on the order of points within the dataset, unlike the Follow-the-Leader approach.

Hierarchical agglomerate clustering algorithms are classified by the heuristics used to decide which clusters are the two most similar ones at a cycle. In order to judge the similarity of algorithms, following distance computation is made by each particular type of the algorithm.

Parametric distance formula used in Hierarchical clustering is included as follows.

$Let\ C_i, C_j be\ the\ clusters\ to\ be\ merged.$ (4.55)
$Let\ C_k be\ a\ cluster\ that\ does\ not\ equal\ to\ neither\ C_i nor\ C_j.$
$Let\ C_t be\ the\ cluster\ obtained\ following\ the\ merge\ operation.$

$$d(C_t, C_k) = distHier(C_i, C_j, C_k) = \alpha_i d(C_i, C_k) + \alpha_j d(C_j, C_k) + \beta\, d(C_i, C_j) + \gamma\, |d(C_i, C_k) - d(C_i, C_k)|$$

$where\ i, j, k, t \in [1\,..\,K], k \neq i, k \neq j, t = \min\{i, j\}, and\ \alpha, \beta, \gamma \in \mathbb{R}.$

Each type of algorithm sets parameters in the distance formula distinctly (Murtagh and Contreras, 2011). Particular settings of parameters are explained in the following.

In single linkage, computation implies that distance between other clusters and the new cluster becomes the minimum of the centers of two clusters that have been merged to a particular cluster. This is ensured by setting the parameters as in below.

$$Single\ linkage\ sets\ \ \alpha_i = 0.5,\ \alpha_j = 0.5,\ \beta = 0,\ \gamma = -0.5. \quad (4.56)$$

Complete linkage imposes the distance of the new cluster to other clusters to be the maximum distance between the centers of each of the two linked clusters to them. The parameters are set as in the following.

$$Complete\ linkage\ sets\ \ \alpha_i = 0.5,\ \alpha_j = 0.5,\ \beta = 0,\ \gamma = 0.5 \quad (4.57)$$

Average linkage computes the distance of the new cluster to others as the mean of the distances between the two linked cluster centers and a particular unlinked cluster. *Weighted pair-group method with averaging* (WPGMA) takes the simple average of the distances as implied by the subsequent equation.

$$WPGMA\ \ ets\ \ \alpha_i = 0.5,\ \alpha_j = 0.5,\ \beta = 0,\ \gamma = 0. \quad (4.58)$$

A more advanced type of linkage, namely *unweighted pair-group method with averaging* (UPGMA), takes the average by taking into consideration the number of elements in the clusters that have been merged. The parameter setting is carried out as follows.

$$UPGMA\ \ sets\ \ \alpha_i = \frac{N_i}{N_i + N_j},\ \alpha_j = \frac{N_j}{N_i + N_j},\ \beta = 0,\ \gamma = 0. \quad (4.59)$$

Linkage criterion may involve taking into account the distance between the linked cluster centers. *Weighted pair-group method using centroids* (WPGMC), a type of centroid linkage method, computes the distance of the new merged cluster to some unlinked cluster as the mean of distances of its constituent cluster centers to the unlinked cluster

diminished by the value of the distance between the linked cluster centroids. WPGMC sets the parameters of the formula as below.

$$WPGMC\ sets\ \alpha_i = 0.5,\ \alpha_j = 0.5,\ \beta = -0.25,\ \gamma = 0. \tag{4.60}$$

Unweighted pair-group method using centroids (UPGMC) is the other centroid-linkage method, which includes the effect of number of elements the linked clusters have in the distance computation as implied by the following equation.

$$UPGMC\ sets\ \ \alpha_i = \frac{N_i}{N_i+N_j},\ \alpha_j = \frac{N_j}{N_i+N_j},\ \beta = -\frac{N_i N_j}{(N_i+N_j)^2},\ \gamma = 0. \tag{4.61}$$

Ward's minimum variance method intends to minimize total intra-cluster variance by controlling the process of the Hierarchical algorithm such that two clusters having the least variance are merged at each cycle. The parameter setting of this criterion is explained below.

$$Ward\ linkage\ sets\ \ \alpha_i = \frac{N_i+N_k}{N_i+N_j+N_k},\ \alpha_j = \frac{N_j+N_k}{N_i+N_j+N_k},\ \beta = -\frac{N_k}{(N_i+N_j+N_k)},\ \gamma = 0. \tag{4.62}$$

The last linkage type covered, *flexible linkage*, does not impose a direct setting of values. Rather, it gives the opportunity to data analyst to choose the value of constants within certain limits, as explained below. By manipulating the constants, a clustering satisfying particular conditions such one having minimum value of an evaluation metric may be obtained.

$$Flexible\ linkage\ is\ constrained\ as\ \ \beta < 1,\ \gamma = 0,\ \ \alpha_i = \frac{1-\beta}{2},\ \alpha_j = \frac{1-\beta}{2}. \tag{4.63}$$

Steps of the agglomerate Hierarchical clustering algorithm are covered subsequently.

- **Step 1**: Initial steps.
    - **Step 1.1**: Build singleton clusters. Add every data point to the cluster formed by it. Set maximum number of iterations to the number of data points.

      $$R_i^q = X_i^q\ and\ \Omega_i = \{X_i\},\ \forall i \in [1\,..\,N]\ and\ \forall q \in [1\,..\,Q]. \tag{4.64}$$

      $$K = N.$$

      $$I_{max} = N,\ \ I = 1.$$

    - **Step 1.2**: Build up the distance matrix. Compute the distance between data points and clusters.

$$Construct\ an\ N \times N\ distance\ matrix\ D = \left[d_{ij}\right]\ , \qquad (4.65)$$
$$d_{ij} = d\left(X_i, C_j\right), \quad \forall i, j \in [1\,..\,N].$$

- **Step 2**: Find closest cluster pairs by making use of the distance matrix. Create a new cluster by merging the obtained pair. Add data points of both clusters to the new cluster. Decrement number of clusters.

$$(i', j') = \ \text{argmin}_{i,j}\{d_{ij}\} \qquad (4.66)$$

$Let\ C_t\ be\ the\ cluster\ obtained\ by\ merging\ C_{i'}\ and\ C_{j'}.$

$\Omega_t = \ \Omega_{i'} \cup \Omega_{j'}\ ,\ \ where\ t = \min\{i', j'\}.$

$K = K - 1.$

- **Step 3**: Update distance matrix by utilizing the Hierarchical distance computation imposed by the type of linkage criterion. For all unlinked clusters, keep distances among each other the same, as included in the current data matrix. Only compute the distances between the new cluster and the unlinked clusters, and store the results in corresponding entries of the new matrix. Set the distance matrix as the new matrix.

$$Construct\ an\ (N-1) \times (N-1)\ distance\ matrix\ D' = \left[d'_{ij}\right]. \qquad (4.67)$$

$d'_{ij} = d_{ij}\ ,\ \ \forall i, j \in [1\,..\,(N-1)] - \{t\}.$

$d'_{it} = \ distHier\left(C_{i'}, C_{j'}, C_i\right),\ \ and\ d'_{ti} = \ d'_{it},\ \forall i \in [1\,..\,(N-1)] - \{t\}.$

$d'_{tt} = \ 0.$

$Set\ D = D'.$

- **Step 4**: Increment number of iterations. If number of iterations equals the maximum number of iterations minus the desired number of clusters, namely the cutoff point is reached, continue with **Step 5**. Else, go to **Step 2**.

$$I = I + 1,\ and\ \ check\ if\ I\ equals\ (I_{max} - K'). \qquad (4.68)$$

- **Step 5**: Update cluster centers by computing feature-by-feature averages of the data points belonging to them.

$$R_j^q = \frac{1}{N_j}\sum_{X_i \in \Omega j} X_i^q \ , \ \forall i \in [1\,..\,N]\,, \forall j \in [1\,..\,K], and\ \forall q \in [1\,..\,Q]. \qquad (4.69)$$

History of the process is kept in a data structure called dendrogram, which is a tree structure having nodes with coordinates of the identities of cluster and the distance level, in which corresponding clusters are merged. Dendrograms have the single cluster obtained at the end of the algorithm in the root position, and the singleton clusters computed at the beginning of the algorithm as leaves.

Quality of Hierarchical clustering depends strictly on the type of the linkage employed in the algorithm. Characteristics of each linkage criterion are covered in the evaluation chapter of the thesis. Hierarchical clustering does not reallocate data points in successive cycles, namely data points only become part of larger clusters hierarchically. If a data point is assigned to a wrong cluster in previous steps, it may not be correctly assigned in subsequent cycles. However, the major advantage of Hierarchical approach is the deterministic computation of data distribution among clusters so that cluster compositions remain the same no matter how many times the algorithm is run.

4.3.12 Self-Organizing Map Clustering

Self-organizing map (SOM) is an unsupervised artificial neural network that produces a projection of multidimensional data in two dimensions (Vesanto and Alhoniemi, 2000). It consists of two layers: input layer through which input vectors are fed and the single computational layer that consists of neurons to be trained. The computational layer, or the map, is generally a two dimensional rectangular or hexagonal lattice formed by adjacent neurons. Input vectors are connected to all neurons in the map. SOM is topology-preserving, meaning that trained map places the input vectors in a topologically-ordered fashion.

Training of SOM is based on a process called *competitive learning*, in which neurons forming the map compete with each other at instances of training, and a particular neuron called *winning neuron* or *the best matching unit* (BMU) becomes activated. BMU has the weights closest to a particular data point. Following its identification, the BMU and all the neighbors that reside within a computed neighborhood are moved towards the data point by a certain amount inversely proportioned by their distance to the data point. Competitive learning ensures the self organization of the map by adjusting the weights as described so that neurons of the lattice can represent the input space after the training.

Neighborhood radius is computed according to the following formula having its parameter as the current iteration number so that radius decreases exponentially over iterations to a value close to 1.

$$r(I) = \frac{1}{2}\left(\max\{Q_{width}, Q_{height}\}\right) e^{\; I \; \log\left(0.5 \max\{Q_{width}, Q_{height}\}\right) \; I_{max}^{-1}} \tag{4.70}$$

Distance between neurons is computed as the Euclidean distance between their positions in the grid, as indicated below.

$$d\left(l_{q_{w_1} q_{h_1}}, l_{q_{w_2} q_{h_2}}\right) = \sqrt{\left(q_{w_1} - q_{w_2}\right)^2 + \left(q_{h_1} - q_{h_2}\right)^2} \tag{4.71}$$

Distance falloff within the neighborhood of a neuron is computed exponentially according to the following formula.

$$f(d, r) = e^{-\frac{d^2}{2\,r^2}}\,, \tag{4.72}$$

where d is the map distance between neurons and r is $r(I)$.

Learning rate determines how fast the weight vector of a neuron in the computational unit converges to a particular input vector during cycles of the training process. It is given as a parameter of SOM clustering algorithm. Learning rate can be selected as a relatively large number for the first phase of training so that self organization process is accelerated. Later, it is set to a smaller value in order to ensure the convergence of the algorithm. This phase of training is called *fine tuning*. Division between the two phases and setting of the learning rate at an instant of training is carried out as follows.

$$\eta_t = \begin{cases} \eta_p e^{-I\,(zI_{max})^{\,-1}} & ,\ if\ I \le zI_{max} \\ \eta_s e^{-(I - zI_{max})((1-z)I_{max})^{-1}} & ,\ otherwise \end{cases} \qquad where\ z \in [0.2, 0.5]. \tag{4.73}$$

SOM provides means of data visualization, by placing closer data points to near locations and more distant points to further locations in the two dimensional grid. In order to determine the clusters, weight vectors of neurons of the trained map should be clustered by making use of the clustering algorithms described so far. Following further clustering, map is divided into regions of clusters.

SOM training is carried out as outlined by the following steps.

- **Step 1**: Assign each data point as an input vector. Input vectors constitute the input layer. Build a lattice of specified size as the competitive layer of the network. Each neuron in the network has a weight vector that has exactly as the same number of dimensions as a data point. Initialize weight vectors to small real random numbers.

$$Let\ \{X_i\}\ be\ the\ set\ of\ input\ vectors,\ \ \forall i \in [1\,..\,N]. \tag{4.74}$$

$$Construct\ a\ lattice\ of\ size\ (Q_{width}, Q_{height}).$$

$$Each\ neuron, l, is\ accessible\ by\ location\ indices\ (q_w, q_h);$$
$$q_w \in [1\,..\,Q_{width}]\ and\ q_h \in [1\,..\,Q_{height}].$$

$$Let\ W = [w_{q_w q_h}]\ be\ the\ weight\ matrix,$$
$$where\ w_{q_w q_h} is\ the\ Q-dimensional\ weight\ vector\ of\ the\ neuron$$
$$at\ location\ adressed\ by\ q_w and\ q_h.$$

$$w^q_{q_w q_h} = rv,\ \ where\ \ q \in [1\,..\,Q]\ and\ rv \in [0,1].$$

- **Step 2**: Grab an input vector and compute its best matching unit in the map.

$$BMU(X_i) = b_{X_i} = l_{q^i_w q^i_h} = \min_{q_w q_h}\{d(X_i, w_{q_w q_h})\}, \tag{4.75}$$
$$where\ q_w \in [1\,..\,Q_{width}]\,, q_h \in [1\,..\,Q_{height}], and\ i \in [1\,..\,N].$$

- **Step 3**: Adjust weights of the winning neuron and its neighbors inversely proportional to their distance of the input vector based on the learning rate of the current iteration.

$$Let\ n(b_{X_i}) = \{l_{q'_w q'_h}\}. \tag{4.76}$$
$$such\ that\ q'_w\ and\ q'_h\ is\ within\ r(I)\ of\ the\ location\ of\ b_{X_i}, i.e\ (q^*_w, q^*_h)$$

$$Set\ \ w^q_{q'_w q'_h} = w^q_{q'_w q'_h} + \eta_t\ f\left(d\left(l_{q^i_w q^i_h}, l_{q'_w q'_h}\right), r(I)\right)\left(X^q_i - w^q_{q'_w q'_h}\right),$$
$$\forall\ l_{q'_w q'_h} \in n(b_{X_i})\ \forall q \in [1\,..\,Q].$$

- **Step 4**: Continue with **Step 2** until all of the input vectors are processed. If all the vectors are finished training, increment the number of iterations by one and start training over the input vectors at **Step 2**. If maximum number of iterations is reached, then stop.

Training algorithm outlined above is a form of *sequential training*, in which update of the weight vectors is carried out just after the winning neuron is computed. On the contrary, *batch training* first locates best matching units for all input vectors and then updates weight vectors of all neurons at one step, as included subsequently.

$$w_{q_w q_h}^{q} = \frac{\sum_{i=1}^{N} f\left(d\left(l_{q_w^i q_h^i}, l_{q_w q_h}\right), r(I)\right) x_i^q}{\sum_{i=1}^{N} f\left(d\left(l_{q_w^i q_h^i}, l_{q_w q_h}\right), r(I)\right)}\ , \tag{4.77}$$

$\forall q_w \in [1\,..\,Q_w]\ \forall q_h \in [1\,..\,Q_h]\ \forall q \in [1\,..\,Q],\ where\ l_{q_w^i q_h^i} = b_{X_i}.$

Batch training eliminates the need for learning rate, thus eliminating explicitly defined phases of training, and tends to converge faster than sequential algorithm, while in most cases yielding as good or better clustering results compared to sequential training, which is discussed in the Chapter 5 of the thesis.

Trained map requires further processing for clustering the data set. Once the training is complete, an unsupervised classification algorithm is run for clustering the final weight vectors of the neurons in the competitive layer. Distribution of the actual data set among clusters is computed by grouping the data points whose best matching units reside in the same cluster together. Final clustering obtained this way is evaluated and visualized by diagrams.

4.4 Summary

A comprehensive review concerning mechanisms of prominent clustering algorithms existing in the literature has been made in this chapter, using a genuine notation. Modifications to original versions of the algorithms have been explicitly stated.

Explained methods can be gathered in six distinct algorithm families, based on the similarity of algorithms. K-means family consists of K-means with centroid initialization through constants, K-means++, K-medians, WFA-K-means, Hopfield-K-means, K-medoids, and Similarity-based K-means. All the members in K-means family follow the same algorithmic flow of initialization, updating the distance matrix, assigning data points to cluster, and updating cluster centroids; although they differ among each other substantially in ways they realize the outlined subtasks.

The second family holds only the Fuzzy K-means algorithm allowing variations in fuzziness factor. Although Fuzzy K-means has a similar algorithmic flow to K-means, it offers fuzzy clustering, imposing profound alterations to sub-tasks associated with K-means family.

ISODATA clustering, a variant of K-means algorithm merges and/or split clusters throughout its course of progress and does not hold fixed number of clusters. Judging by the underlying mechanisms, ISODATA is placed in the third family consisting only of itself.

Remaining algorithms form algorithm families by themselves: Hierarchical family containing every linkage criterion, Follow-the-Leader algorithm and SOM clustering.

Evaluation of performance quality of methods and illustrated clustering results are covered in Chapter 5. Associated running times and scalability of the algorithms are analyzed in Chapter 6. Classification of clustering results and corresponding applications are included in Chapter 7.

CHAPTER 5

EVALUATION OF CLUSTERING ALGORITHMS

5.1 Introduction

Quality of a clustering algorithm can be demonstrated via evaluating generated clusters internally by validity indices that provide means of comparison among distinct clustering results and present the infra structure for automatically selecting the optimum number of clusters to be generated on a particular data set.

Design and evaluation of algorithms have a mutual relation between each other such that following the design process, algorithms whose performances are of high quality are considered to be promising; and in order to come up with algorithms that can generate high quality clustering, design of algorithms should be improved.

This chapter presents an elaborated evaluation of all algorithms generating distinct number of clusters for each data set in each distance framework. Evaluation results are analyzed in relation with algorithmic mechanisms outlined in Chapter 4. A comparison of algorithms based on accuracy is also included at the end of the chapter.

5.2 Evaluation Metrics

Quality of performance of a clustering can be measured via several evaluation metrics (Halkidi et al., 2001). Among many, Davies-Bouldin Index, Modified Dunn Index, Clustering Dispersion Indicator, and Similarity Matrix Indicator are used to assess quality along with accuracy.

Distance computation is refined for evaluation purposes so that metric results are filtered-out from the number of clusters, data set size, and dimensionality. Distance between two vectors in multi-dimensional space is calculated throughout this chapter as follows.

$$d(v_1, v_2) = \frac{1}{\sqrt{Q}} \|v_1 - v_2\|, \quad (5.1)$$

$where\ v_1 and\ v_2 are\ both\ Q - dimensional\ vectors.$

Distance in equation (5.1) is the Euclidean distance scaled by the reciprocal of the squared root of dimensionality. Squared distance between two vectors is simply the square of (5.1), as indicated in the following.

$$d^2(v_1, v_2) = \frac{1}{Q} \|v_1 - v_2\|^2 , \quad (5.2)$$

$where\ v_1 and\ v_2 are\ both\ Q - dimensional\ vectors.$

Distance of a data point to a set of data points that belongs to the same cluster is computed as in equation (5.3) by taking the geometric mean of the squared distances between the data point and all members of the set.

$$d(X_i, \Omega_j) = \sqrt{\frac{1}{|\Omega_j|} \sum_{X_{i'} \in \Omega_j} d^2(X_i, X_{i'})} \ , \ where\ i \in [1..N]\ and\ j \in [1..K]. \quad (5.3)$$

Average distance regarding a set of data points is calculated as in equation (5.4) by making use of geometric mean operator. It measures scattering of data points within a cluster.

$$d(\Omega_j) = \sqrt{\frac{1}{2\,|\Omega_j|} \sum_{X_i \in \Omega_j} d^2(X_i, \Omega_j)} \ , \ where\ i \in [1..N]\ and\ j \in [1..K] \quad (5.4)$$

Distance of a cluster to all clusters is computed as in equation (5.5).

$$d(C_j, \Psi) = \sqrt{\frac{1}{|\Psi|} \sum_{C_{j'} \in \Psi} d^2(R_j, R_{j'})} \ , \quad (5.5)$$

$where\ j \in [1..K]\ and\ \Psi\ is\ the\ set\ of\ all\ clusters$

Average distance among all cluster centroids is computed as follows.

$$d(\Psi) = \sqrt{\frac{1}{2\,|\Psi|} \sum_{C_j \in \Psi} d^2(C_j, \Psi)} \ , \ where\ \Psi\ is\ the\ set\ of\ all\ clusters. \quad (5.6)$$

DBI is computed by making use of the sum of average inter-distance of two clusters' data points, which is calculated as in equation (5.7).

$$Z_{j_1 j_2} = \frac{d(\Omega_{j_1}) + d(\Omega_{j_2})}{d(C_{j_1}, C_{j_2})} \ , \ where\ j_1 \in [1..K]\ and\ j_2 \in [1..K]. \quad (5.7)$$

The basic mechanism used by DBI metric to compare two clusters is $Z_{j_1 j_2}$ which is computed in equation (5.7) to measure the ratio of sum of within cluster scatter of two clusters to their inter-distance. More compact clusters whose centers are sufficiently far away from others are awarded with lower values of $Z_{j_1 j_2}$.

For a cluster, maximum value of Z is obtained by traversing through the set of all clusters to find the companion cluster that maximizes the value of the metric, as indicated by (5.8).

$$Z_i = \max_{j' \in [1..K]-\{j\}} Z_{ij'} \quad , where\ i \in [1..K]. \tag{5.8}$$

Davies-Bouldin Index is then simply the average of maximum Z values of all clusters, as computed by the equation in (5.9). DBI examines the quality of each cluster separately by challenging them with other clusters and then presents the average case regarding all clusters.

$$DBI = \frac{1}{K} \sum_{i=1}^{K} Z_i \tag{5.9}$$

Modified Dunn Index is derived from Dunn Index (Chicco et al., 2006), whose lower values indicate better performance, unlike the original metric. The computation of the index involves discovering the maximum intra-cluster scatter and the minimum inter-cluster distance of two unequal clusters as in equation (5.10).

$$MDI = \frac{\max_{j \in [1..K]} \{d(\Omega_j)\}}{\min_{j_1 \neq j_2} \{d(C_{j_1}, C_j)\}} \quad , \quad where\ j_1, j_2 \in [1..K]. \tag{5.10}$$

MDI is the ratio of maximum scatter to minimum inter-cluster distance. High values of the metric are yielded for cases in which clusters are close to each other and data points are scattered within clusters. In other words, if clusters are not well-separated and data points residing in them exhibit high statistical variance, MDI decides that the clustering is of poor quality.

Clustering Dispersion Indicator is computed according to the following formula.

$$CDI = \frac{1}{d(\Psi)} \sqrt{\frac{\sum_{j=1}^{K} d^2(\Omega_j)}{K}} \quad , where\ \Psi\ is\ the\ set\ of\ all\ clusters. \tag{5.11}$$

Equation in (5.11) implies that as average compactness of the clusters decrease together with average inter-cluster distance, CDI metric yields high values and deciding the clustering in question to be of bad quality.

Similarity Matrix Indicator, proposed in (Chicco et al., 2002), is the maximum value of similarity between two distinct clusters. Similarity of the two clusters is computed as in the following.

$$SMI = \max_{j_1 \neq j_2} \left\{ \left(1 - \frac{1}{\ln\left(d(C_{j_1}, C_{j_2})\right)} \right)^{-1} \right\} \quad , \quad where\ j_1, j_2 \in [1..K]. \tag{5.12}$$

If similarity between any two clusters is computed to be high, then SMI value will be high, indicating poor-quality performance.

Note that MDI, and SMI deals with extremum cases present in the clustering, while CDI works on average cases concerning all clusters and data points. On the other hand, DBI deals with both cases. All metrics are biased towards compactness and well separateness of clusters to deem a clustering accurate. Lower metric values indicate better performance for all validity indices.

Two additional measures are used in the analysis of fuzzy K-means clustering. Partition Coefficient (PC), the first one, measures the degree of fuzziness in the membership matrix and is computed as in equation (5.13). PC values reside in the interval [0, 1]. Higher values of PC indicate more decisive partition of the data among clusters and a less fuzzy clustering. Hard clustering attains a PC value of 1.0, in which a data point belongs to a cluster with a probability of 1.

$$PC = \frac{1}{N}\sum_{i\in[1..N]}\sum_{j\in[1..K]} u_{ij}^2 \ . \tag{5.13}$$

The second measure is Classification Entropy (CE), which quantifies the amount of information held in the membership matrix by using Shannon's entropy. As a clustering gets fuzzier, the information that could be encoded increases. Therefore, a CE value that is close to 0 indicates that corresponding clustering is more of an example of hard clustering. On the other hand, if a CE value is closer to 1, the clustering is said to be a representative of soft clustering. Computation of CE is included in the equation in (5.14).

$$CE = -\frac{1}{N}\sum_{i\in[1..N]}\sum_{j\in[1..K]} u_{ij}\log\left(u_{ij}\right) \tag{5.14}$$

Both PC and CE measures do not explicitly assess the validity or quality of the clustering as no computation is performed concerning data points or clusters in equations in (5.13) and (5.14). Instead, they are heuristic measures that provide information about the degree of fuzziness present in the clustering, demonstrating whether a clustering is more hard or soft.

5.3 Evaluation Methodology

All algorithms are run to generate between 5 and 20 clusters on winter and summer weekdays data sets in Euclidean, hybrid Pearson, and weighted Euclidean frameworks. DBI, MDI, CDI, and SMI metric values associated to each case are visualized in the proceeding section together with an example of high quality clustering generated by algorithms, as decided by metric curves.

Algorithms including randomness in their mechanisms; K-means++, K-medians, WFA-K-means, Hopfield-K-means, K-medoids, Similarity-based K-means, Fuzzy K-means,

ISODATA, and SOM clustering; are run with the same parameters 25 times and the clustering with the lowest DBI value that does not include empty clusters is presented.

K-means clustering is run with distinct values of centroid-initializer constants in the interval [0, 1] and the non-empty clusters with the lowest DBI are included.

Maximum iteration of the training of the Hopfield network used by Hopfield-K-means clustering is set to 600 so that convergence of the total energy of states can be observed.

Parameters of ISODATA clustering are adjusted to yield between 5 and 20 clusters Evaluation of this algorithm is carried out only briefly in accordance with other methods, although more detailed evaluation is required to assess the performance of ISODATA to reveal the affects of each parameter of the algorithm on the final clustering.

Fuzziness factor of fuzzy K-means is set as 2.0 throughout the evaluation. Euclidean and weighted Euclidean distance metrics are used in squared forms for fuzzy clustering.

Distance threshold of Follow-the-Leader clustering is adjusted to generate between 5 and 20 clusters. However, some particular number of clusters could not have been yielded by the algorithm, as indicated by the following section of this chapter.

Distance threshold utilized by the fine tuning phase of similarity-based K-means clustering is automatically determined, as proposed in the corresponding section of Chapter 4.

Hierarchical clustering is separately evaluated for every linkage criterion except WPGMC and UPGMC, which fail to generate regular dendrograms (Murtagh and Contreras, 2011) due to non-monotonicity in distance levels of merged clusters as the algorithm progresses. Other criteria exhibit monotonicity in distance levels. For flexible linkage, β is set to 21 distinct values that divide the interval [-0.5, 0.95] and clustering with lowest MDI values is illustrated. Except UPGMA and flexible linkage criteria, every linkage criterion is evaluated only on winter weekdays data due to chapter volume considerations.

Regarding each distance framework, data set, and number of clusters; SOM clustering is run separately in order to demonstrate the affect of random initialization on final clusters, despite the fact that only one trained map could be utilized to generate clusters regarding each case. Hierarchical clustering with UPGMA linkage is selected to carry out the post-clustering of the map. SOM clustering is evaluated only on winter weekdays data in order to save space.

Since $5\sqrt{N}$ map nodes are sufficient to cluster a data set of size N (Vesanto and Alhoniemi, 2000), map size is selected to be $6x15$ to effectively process 306 instances of the winter weekdays data. Both batch and sequential training techniques are run and

evaluated. Concerning sequential training, initial learning rate and secondary learning rate are set to 0.10 and 0.02, respectively. Number of iterations in the training step is set to 90.

Maximum number of iterations until convergence is set to 100 for all algorithms except Hierarchical clustering and SOM clustering.

Clustering of winter and summer weekends and holidays data is excluded from this chapter due to space limitations and prevent distraction.

In clustering diagrams, hours of the day reside in the horizontal axis and normalized consumptions amounts are held in the vertical axis.

5.4 Evaluation Results

Evaluation results of cases outlined in the methodology section is presented in the proceeding sub-sections for every clustering method. For all curves included in this section, red stands for weighted Euclidean, purple stands for Euclidean and orange stands for hybrid Pearson frameworks.

5.4.1 K-Means Clustering Evaluation

Clustering evaluation results for the data set of winter weekdays is included as follows.

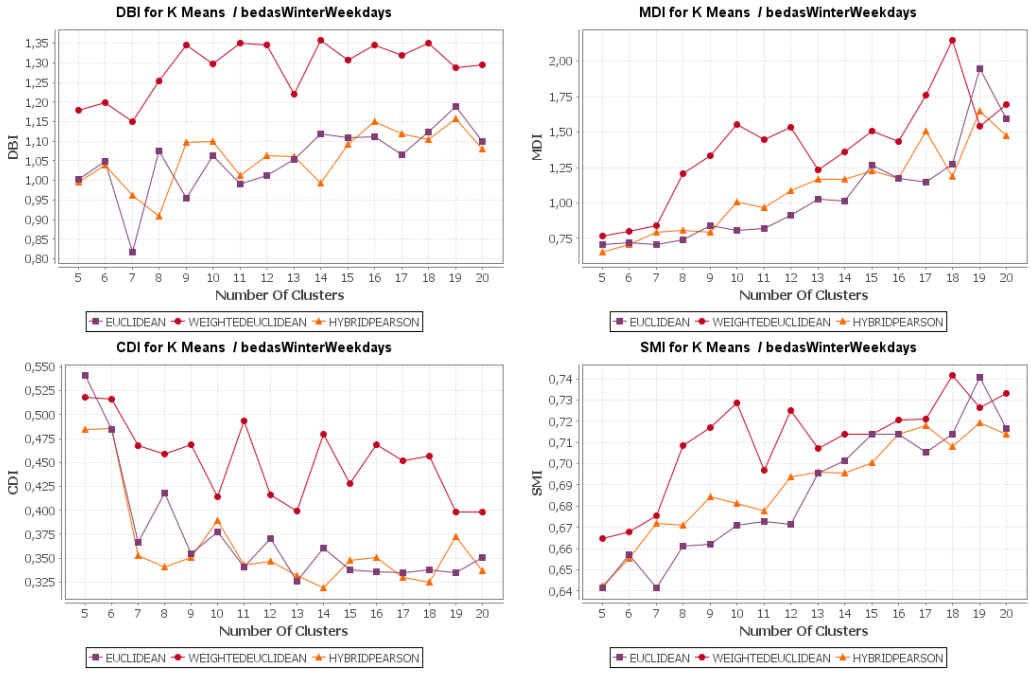

Figure 5.1 Evaluation of K-means for winter weekdays data.

Evaluation results summarized in (Fig. 5.1) suggest that the performance of weighted Euclidean metric is inferior with respect to Euclidean and hybrid Pearson distance measures, keeping in mind that lower values of the indices imply better performance. Inspecting all values in the graphs, the lowest corresponds to the results obtained via K-means algorithm in Euclidean distance framework for seven clusters. The corresponding clustering is included in the figure below.

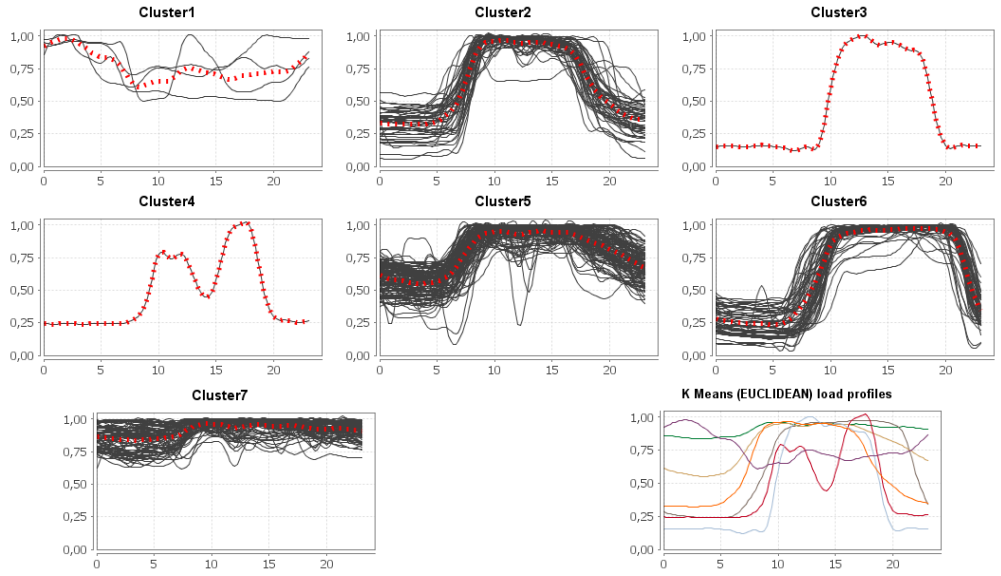

Figure 5.2 Clustering of winter weekdays data by K-means in Euclidean framework.

Summer weekdays clustering evaluation results are included subsequently.

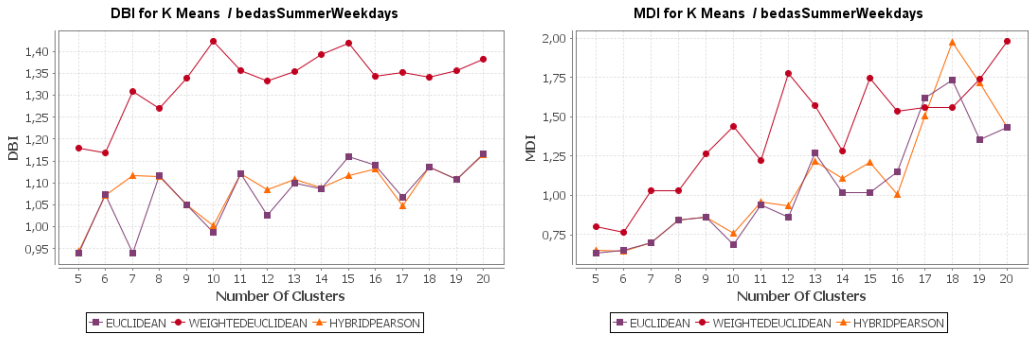

Figure 5.3 Evaluation of K-means for summer weekdays data.

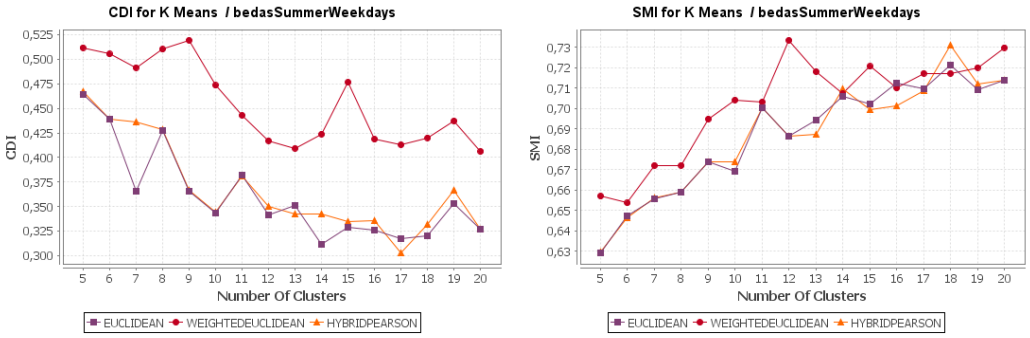

Figure 5.3 Evaluation of K-means for summer weekdays data. (cont'd)

DBI, CDI, and MDI metrics in (Fig. 5.3) agree that 10 clusters could be selected as load profiles and Euclidean distance metric yields only very slightly better results than the modified Pearson while outperforming the weighted Euclidean. Customer distribution within clusters is included below.

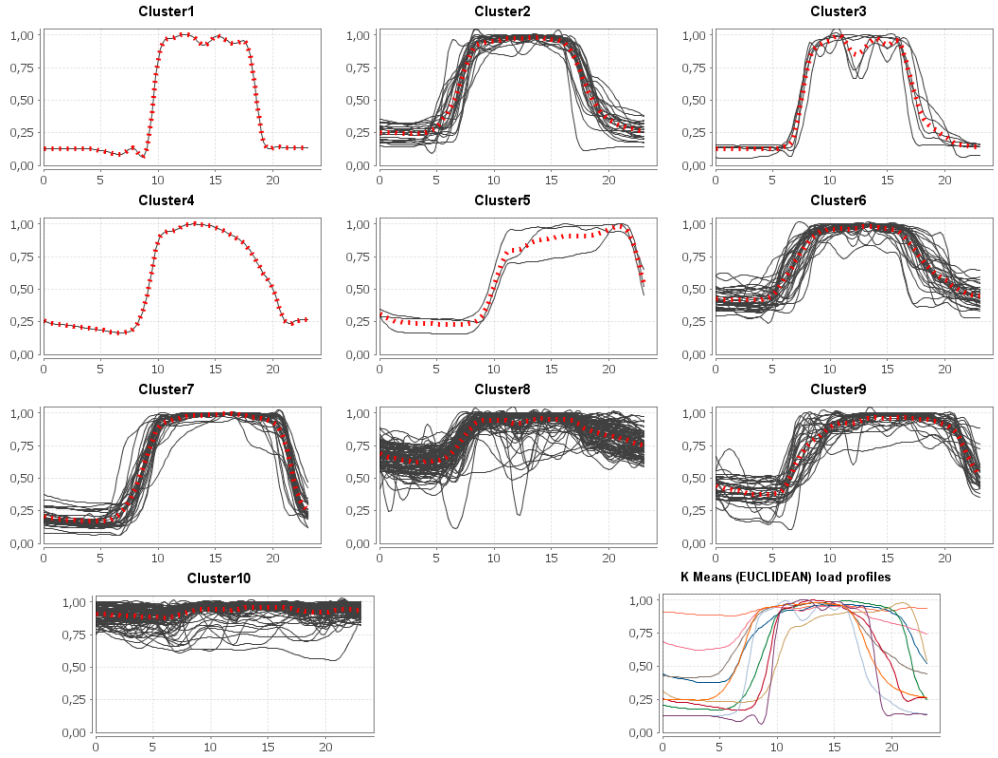

Figure 5.4 Clustering of summer weekdays data by K-means in Euclidean framework.

K-means clustering with deterministic centroid initialization succeeds to form main clusters with several data points as in (Fig. 5.2) and (Fig. 5.4). These large clusters tend to have more or less the same number of elements. Although the algorithm can have some outliers, whose elements vary considerably from the rest of the load profiles yet resemble

each other grouped such as in *Cluster 1* of winter weekdays data set and *Cluster 5* of summer weekdays data set; it may yield redundant clusters having only one element such as *Cluster 3* and *Cluster 4* of the winter weekdays data, and *Cluster 1* and *Cluster 4* of summer weekdays data. Moreover, the algorithm cannot guarantee the absence of noise within clusters such as noisy representative diagrams residing in *Cluster 5* of winter weekdays data and *Cluster 8* of summer weekdays data.

5.4.2 K-Means++ Clustering Evaluation

Evaluation results for the winter weekdays data set are summarized in the following charts.

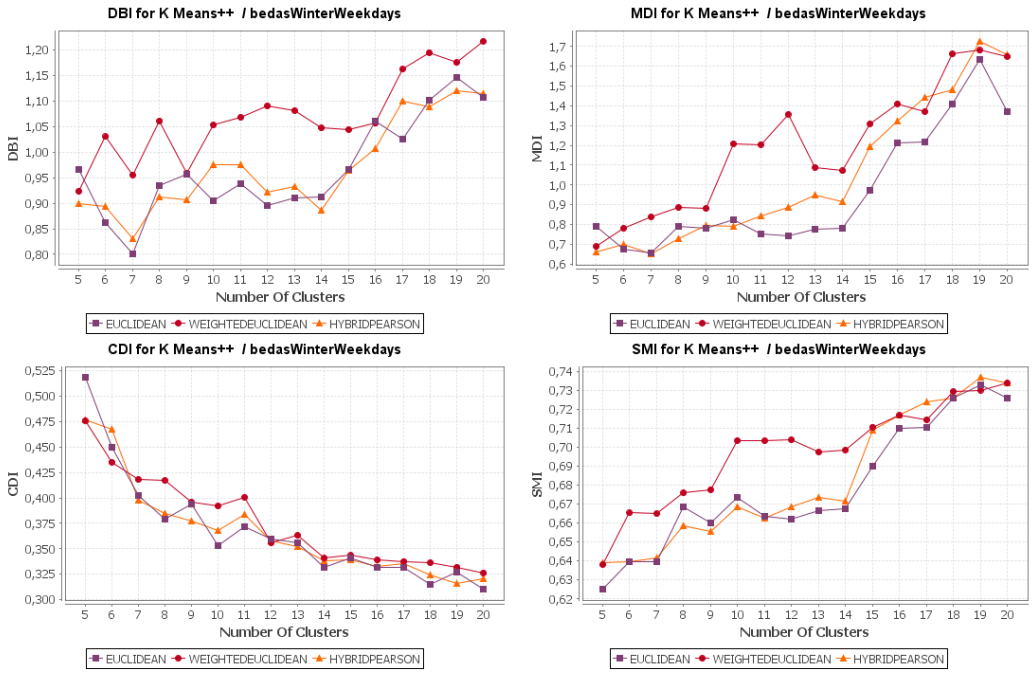

Figure 5.5 Evaluation of K-means++ for winter weekdays data.

Charts included in (Fig 5.5) indicate that the minimum values computed by DBI and MDI indices correspond to seven clusters using Euclidean distance metric followed by the hybrid Pearson. Moreover, CDI metric suggests that greatest decrease in the graph has occurred for six or seven clusters produced by the algorithm employing Euclidean metric. SMI metric also agrees with CDI, as it has almost the same value for six or seven clusters on Euclidean or hybrid Pearson metric. Seven clusters for winter weekdays yielded by the algorithm in a Euclidean distance framework are depicted subsequently.

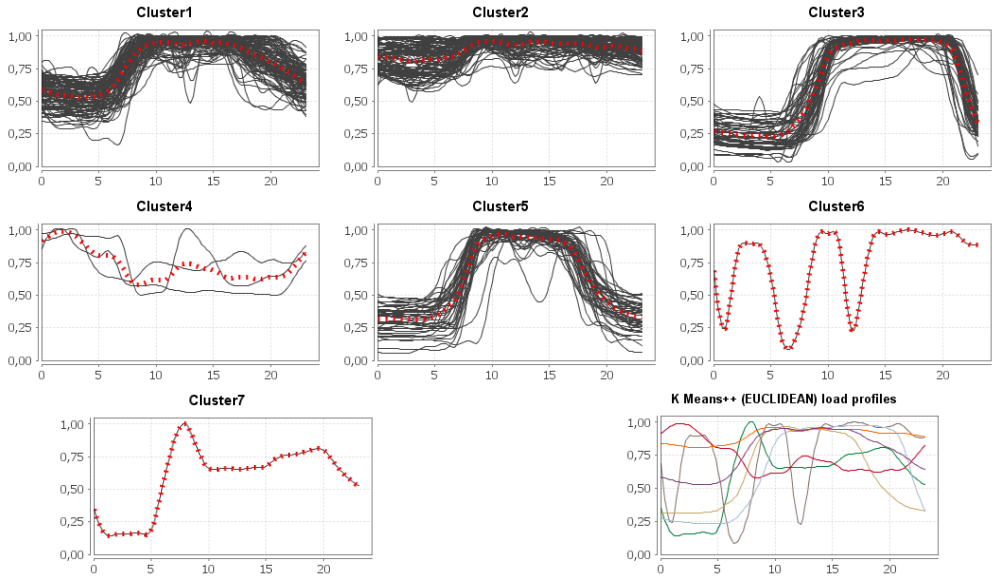

Figure 5.6 Clustering of winter weekdays data by K-means++ in Euclidean framework

In case of seven clusters as illustrated for Euclidean framework in (Fig. 5.6), K-means++ clustering utilizing hybrid Pearson distance measure attains more or less the same results of evaluation as curves in (Fig. 5.5) suggest. Moreover, it is clearer for the hybrid Pearson framework that the knee of DBI, MDI and CDI graphs correspond to seven clusters, rather than controversial case of six clusters for Euclidean framework. Hence clustering of winter weekdays by K-means++ with hybrid Pearson distance is also included below for comparison.

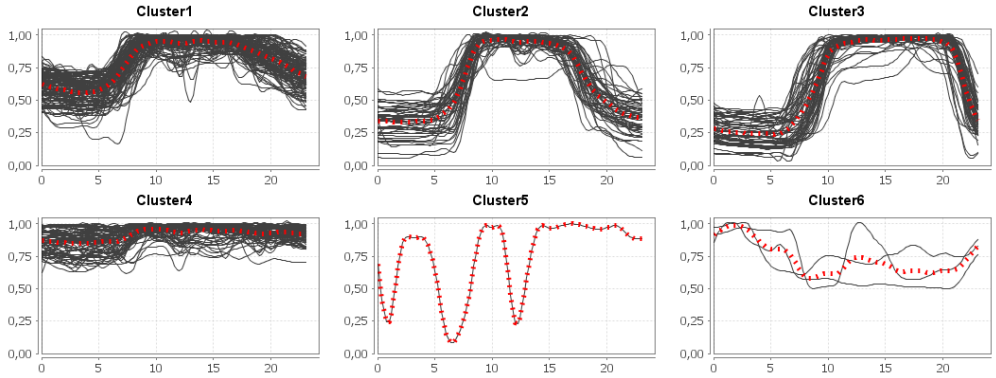

Figure 5.7 Clustering of winter weekdays by K-means++ in hybrid Pearson framework.

Figure 5.7 Clustering of winter weekdays by K-means++ in hybrid Pearson framework. (cont'd)

Distinction between Euclidean and hybrid Pearson frameworks should be sought in clustering of outliers by inspecting (Fig. 5.6) and (Fig. 5.7). Both frameworks have two of their smaller clusters with the same composition, yet Euclidean framework has its singleton *Cluster 7*, and hybrid Pearson framework has its *Cluster 7* with two members as different.

As visual inspection suggests, although some of the large clusters obtained in both frameworks are quite similar, hybrid Pearson framework has its large clusters more compact, due to taking into account the correlation between features of data points. Evaluation metrics are biased in such a way that singleton clusters are considered to be more compact than ones such as *Cluster 7* in (Fig 5.7), which in turn might affect the decision-making in subtle conditions. Hence, visual inspection by the data analyst may be useful in such cases.

Evaluation of summer weekdays dataset is summarized in the following charts.

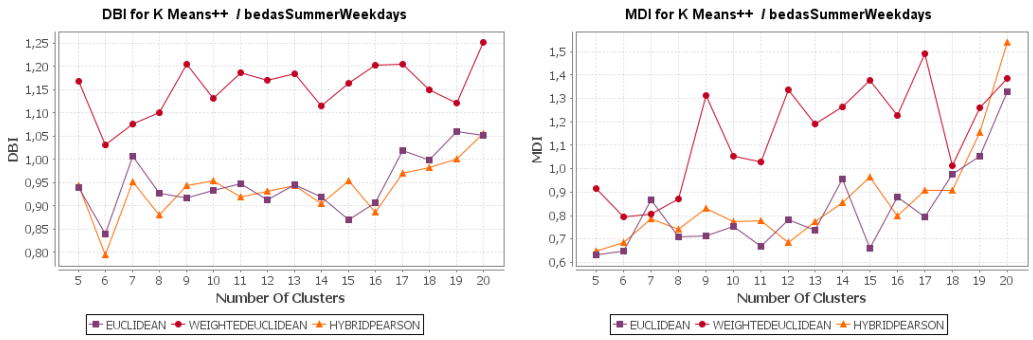

Figure 5.8 Evaluation of K-means++ for summer weekdays data.

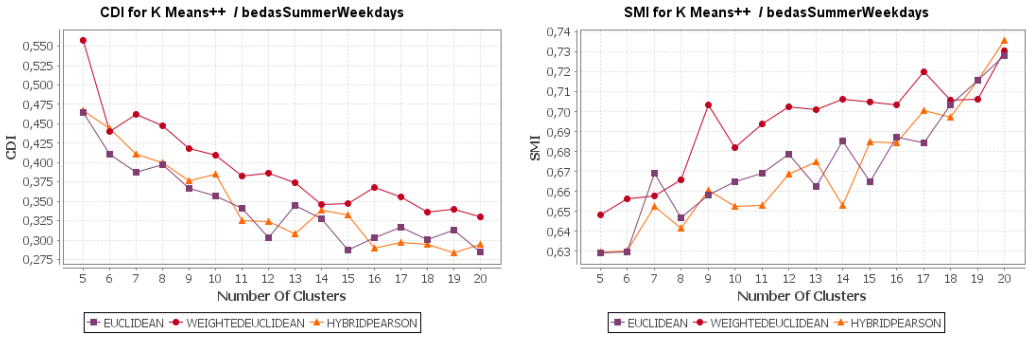

Figure 5.8 Evaluation of K-means++ for summer weekdays data. (cont'd)

DBI metric suggests that for six clusters greatest decrease in the chart is obtained and hybrid Pearson framework attains the lowest value followed by Euclidean and weighted Euclidean frameworks. MDI and SMI evaluations point out that for six clusters performance of Euclidean and hybrid Euclidean frameworks are almost of the same quality. Although CDI suggests that Euclidean framework yields slightly better results, clustering of K-means++ depicted below is presented in hybrid Pearson framework due to its achieving the greatest decrease for six clusters in DBI chart in (Fig. 5.8).

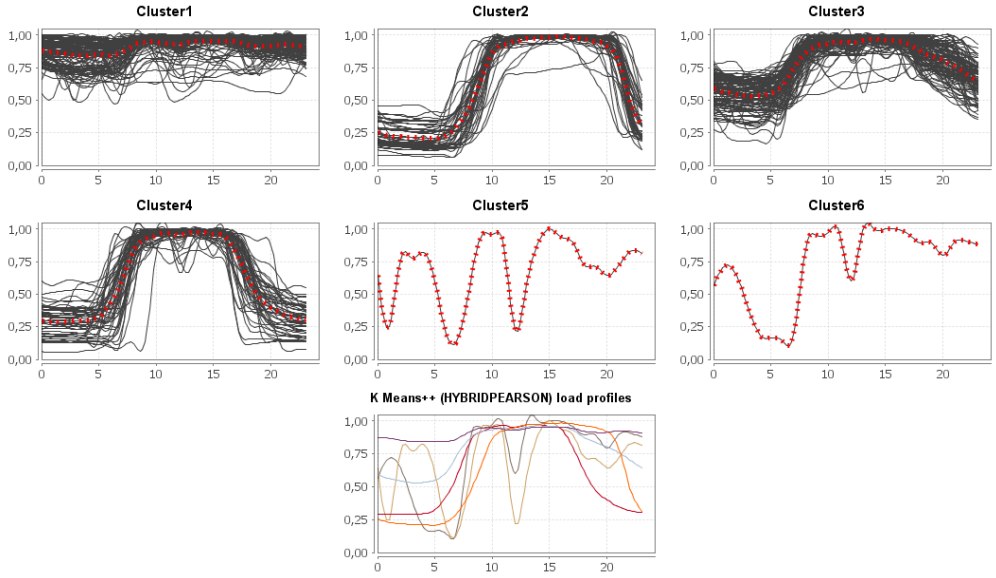

Figure 5.9 Clustering of summer weekdays by K-means++ in hybrid Pearson framework.

K-means++ clustering yields main clusters that can vary in size and generally includes less intra-cluster variation as in (Fig 5.9), compared to K-means clustering. The algorithm also groups outliers that tend to vary considerably from the rest of the load profiles in separate clusters. Visual inspection may come handy in cases which Euclidean and hybrid

Pearson distance measures have similar performance qualities, since the latter is expected to behave marginally better in outlier detection and grouping.

5.4.3 K-Medians Clustering Evaluation

Performance evaluation of K-medians with increasing number of clusters for winter weekdays data set is included subsequently.

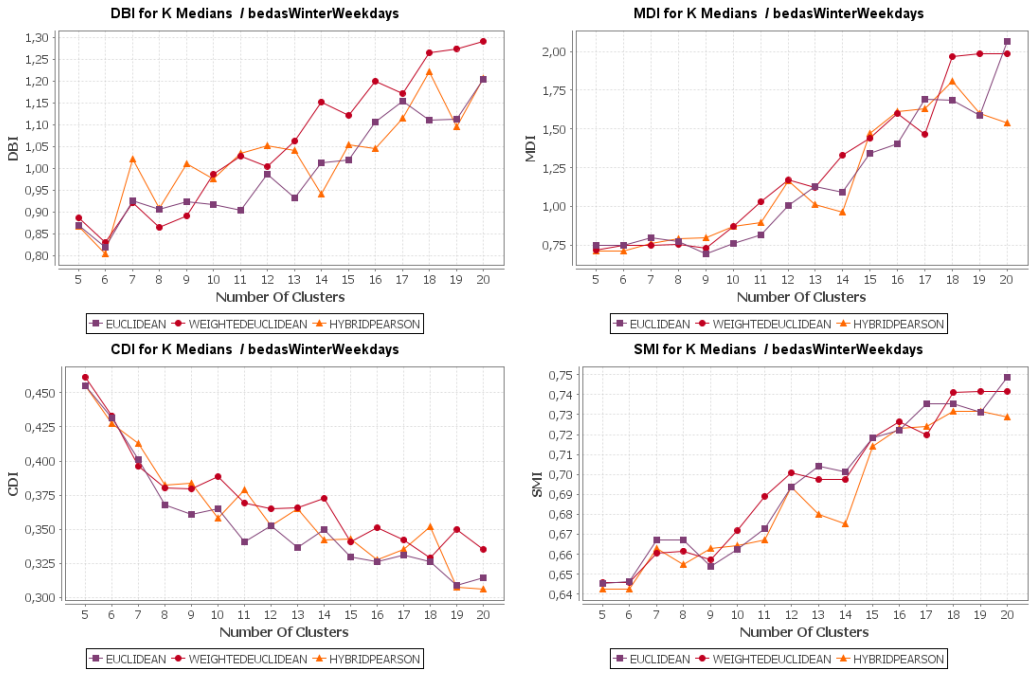

Figure 5.10 Evaluation of K-medians for winter weekdays data.

Performance of K-medians clustering in all distance frameworks is measured by the metrics in (Fig 5.10) to yield a similar condition as K-means and K-means++, in which weighted Euclidean framework is always outperformed by the latter frameworks. Although best performance according to DBI is for six clusters with hybrid Pearson distance measure, the greatest declines in both DBI and CDI indices correspond to eight clusters in hybrid Pearson framework, which is depicted in the following.

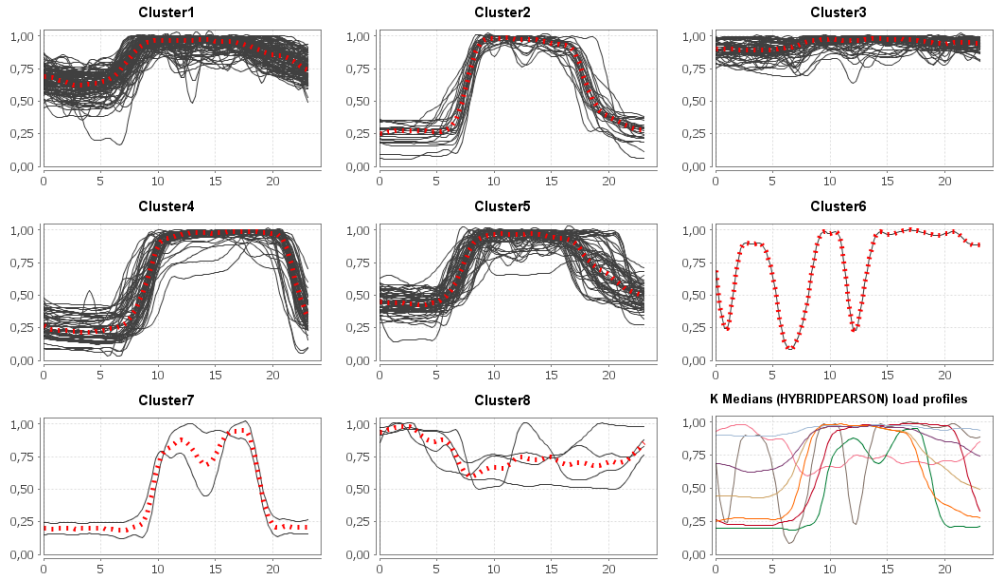

Figure 5.11 Clustering of winter weekdays by K-medians in hybrid Pearson framework.

K-medians algorithm effectively forms compact large clusters as well as small clusters holding customers with distinct electricity consumption patterns as illustrated in (Fig 5.11). Evaluation of summer weekdays data is included below for comparison.

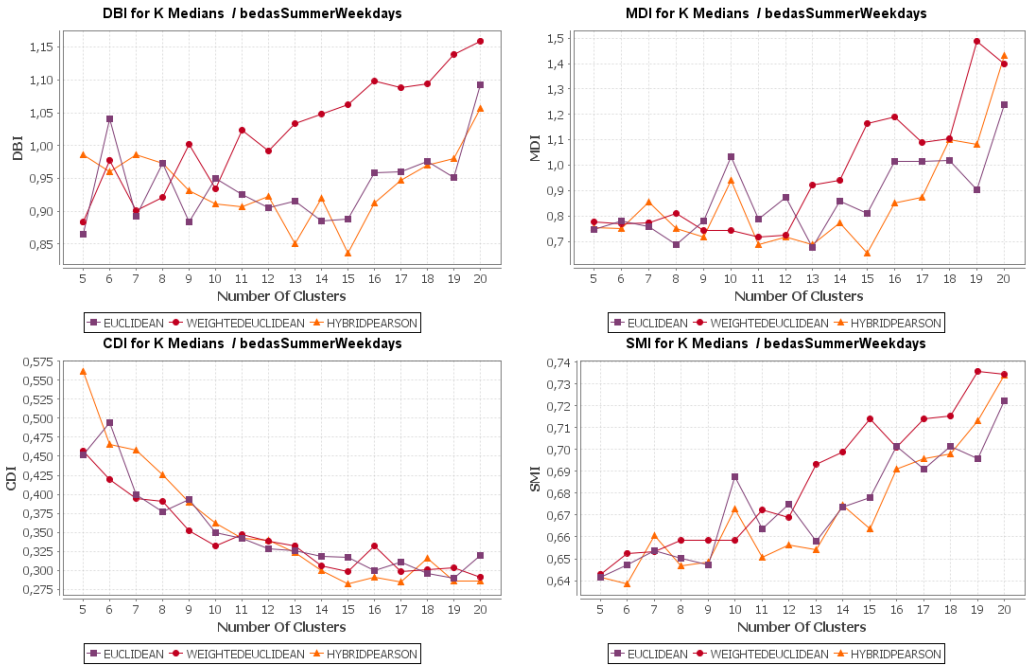

Figure 5.12 Evaluation of K-medians for summer weekdays data.

Knee point of CDI graph corresponds to seven clusters on Euclidean distance as included in (Fig. 5.12). However, second decline in DBI values corresponds to nine clusters, which is evaluated to be slightly more promising with respect to DBI, CDI, and SMI metrics. MDI value for nine clusters is more or less equal to the value for seven clusters. Looking for more diverse and robust grouping of customers, nine clusters in Euclidean framework are adopted as load profiles, as in the following graphs.

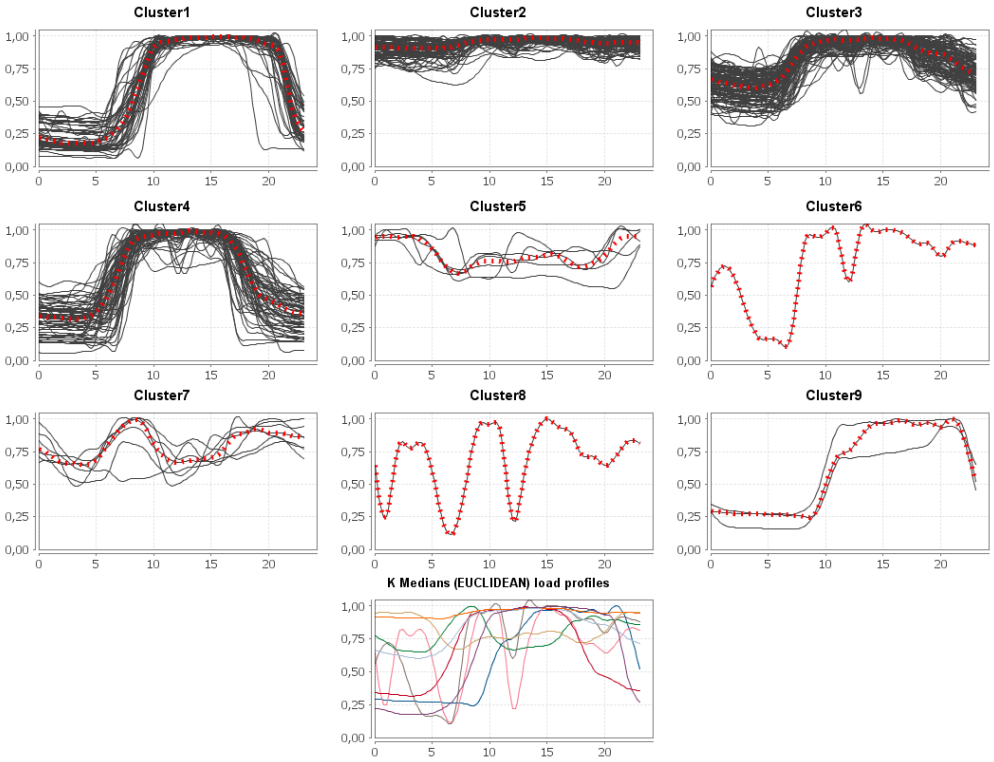

Figure 5.13 Clustering of summer weekdays data by K-medians in Euclidean framework.

Clustering included in (Fig. 5.13) shows that the algorithm ensures the formation of compact large clusters and small clusters with variable sizes, which hold the outlier consumption patterns. Utilizing median operator to determine cluster centers results in protecting clusters against noisy data points by filtering out the effect of such points in centroid computation.

5.4.4 WFA-K-Means Clustering Evaluation

Evaluation of performance of WFA-K-means clustering for winter weekdays data set is summarized in the following charts.

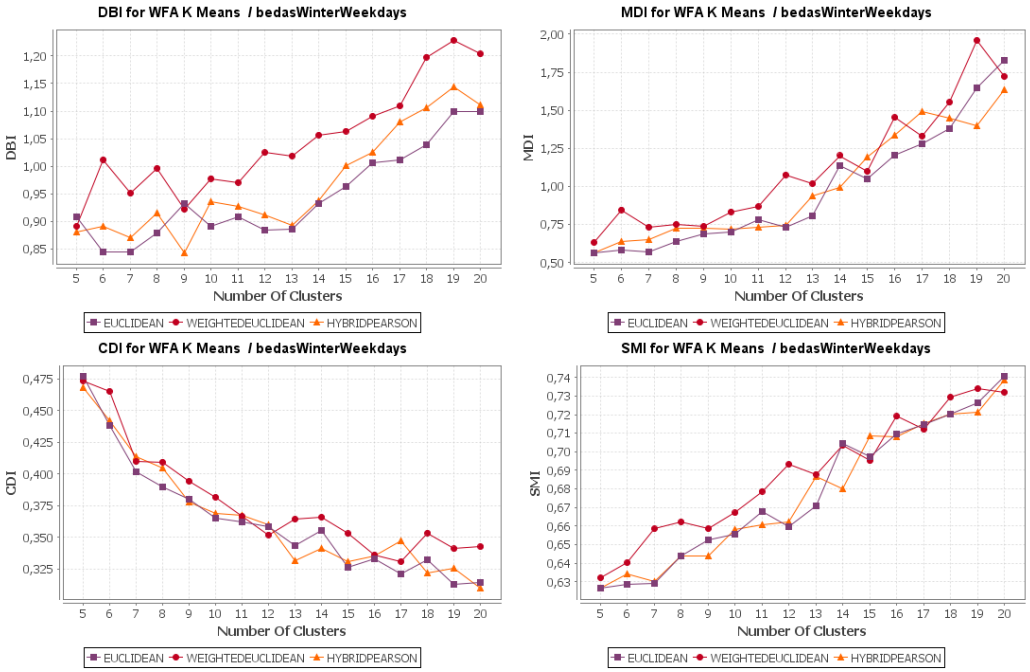

Figure 5.14 Evaluation of WFA-K-means for winter weekdays data.

Graphs in (Fig. 5.14) demonstrate that the lowest DBI value is for nine clusters in a hybrid Pearson distance framework. Similarly, the greatest decrease in CDI values is for nine clusters in this framework. MDI and SMI values for nine clusters are greater than prior number of clusters. However, for this number of clusters, DBI value is considerably high and MDI and SMI graphs tends to follow a steadily increasing curve for which the elbow point can be decided as nine clusters. Therefore, the winter weekdays data set is partitioned among nine clusters in hybrid Pearson distance framework as included in the following.

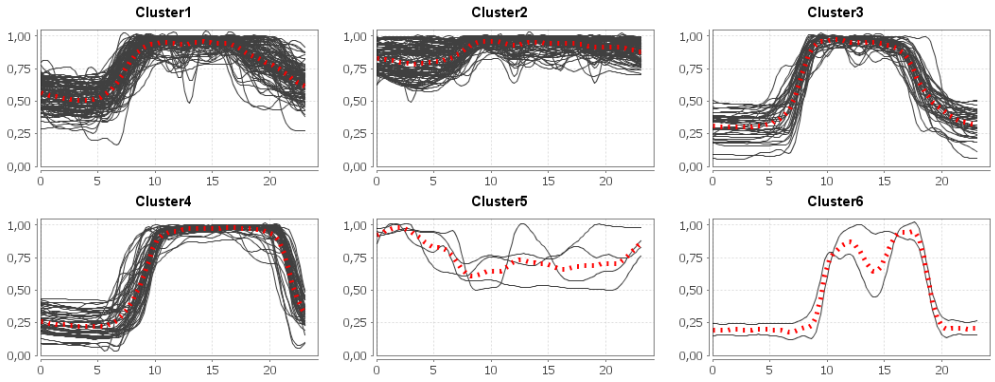

Figure 5.15 Clustering of winter weekdays data by WFA-K-means in hybrid Pearson framework.

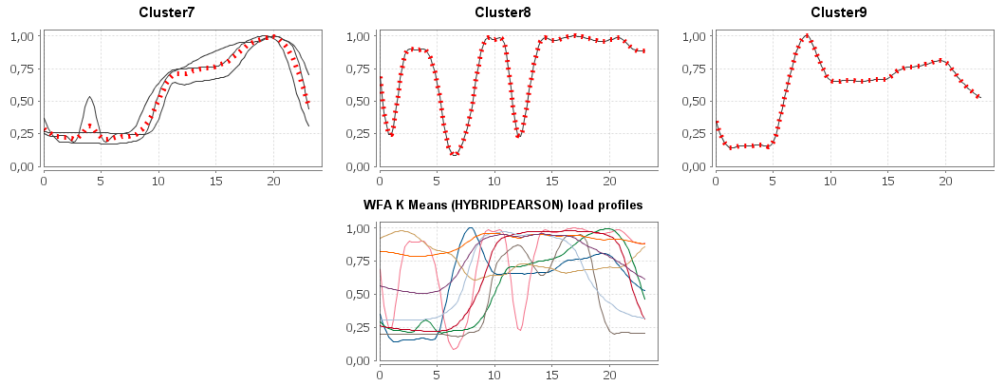

Figure 5.15 Clustering of winter weekdays data by WFA-K-means in hybrid Pearson framework. (cont'd)

WFA-K-means discovers newer outlier elements such as the ones grouped in *Cluster 7*, thus reducing the data diversity in large clusters, especially in *Cluster 4*, as shown in (Fig. 5.15).

Evaluation of summer weekdays data clustering is summarized in the charts below.

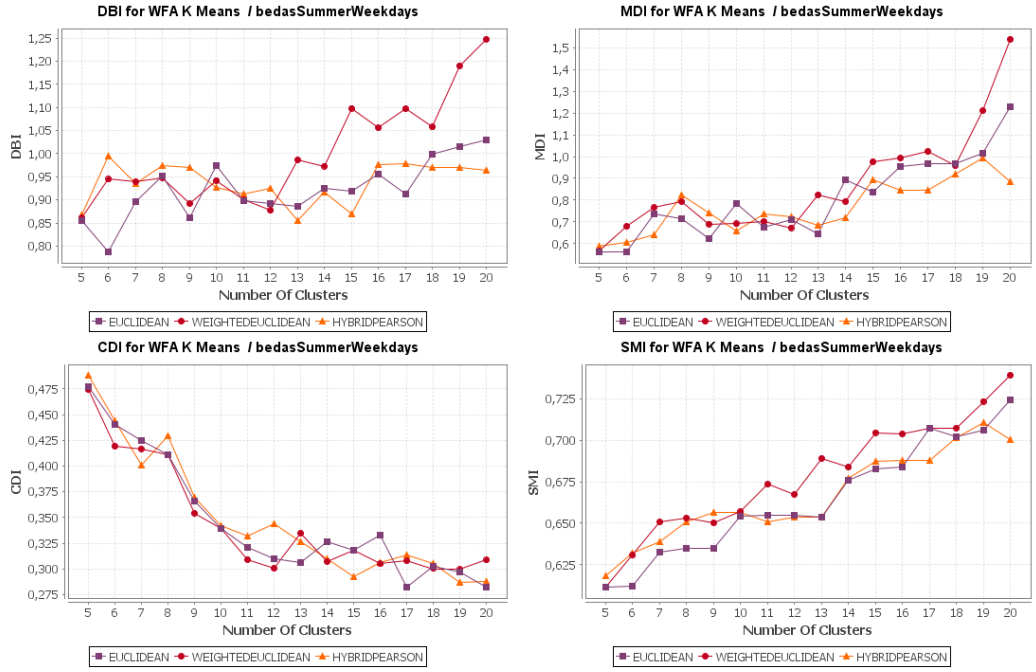

Figure 5.16 Evaluation of WFA-K-means for summer weekdays data.

DBI metric included in (Fig. 5.16) indicates that the best performance is for six clusters in a Euclidean distance framework. MDI and SMI evaluations confirm this and six clusters lie on a local knee point of CDI chart. Clustering of summer weekdays is shown in subsequent diagrams.

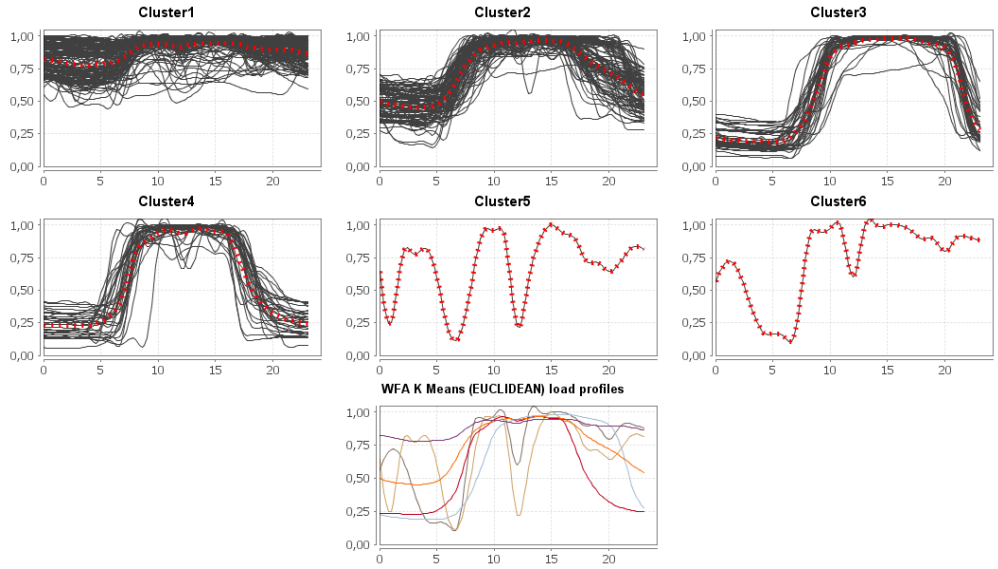

Figure 5.17 Clustering of summer weekdays by WFA-K-means in Euclidean framework.

Clustering in (Fig. 5.17) does not yield most of the outlier groupings that have been examined so far. However, it does capture the general shape of the consumption shapes and discriminates two data points as outliers. Nonetheless, large clusters contain certain amount of noise, especially *Cluster 1*.

5.4.5 Hopfield-K-Means Clustering Evaluation

Hopfield-K-means clustering evaluation for winter weekdays data set is summarized in the following charts.

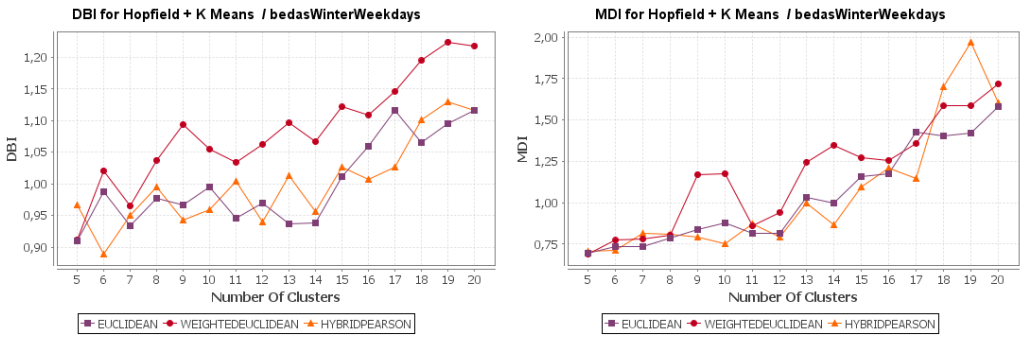

Figure 5.18 Evaluation of Hopfield-K-means for winter weekdays data.

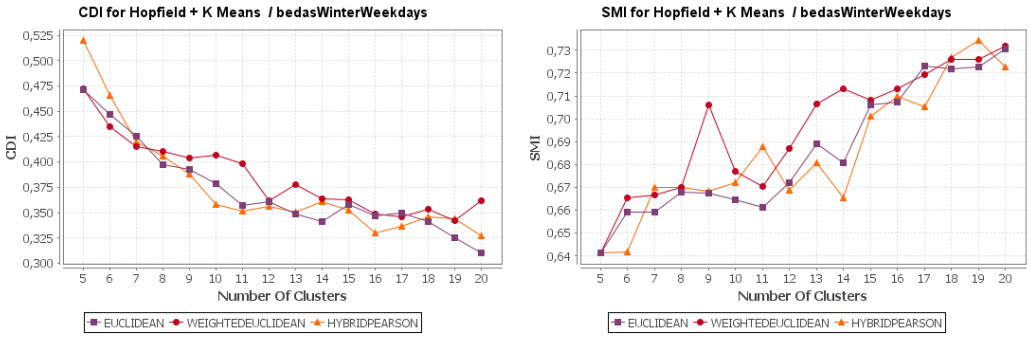

Figure 5.18 Evaluation of Hopfield-K-means for winter weekdays data. (cont'd)

14 clusters in Euclidean framework is a knee point for DBI, MDI and CDI metrics and a local minimum in SMI curve as in (Fig. 5.18). Therefore, 14 clusters are selected for winter weekdays as illustrated in the following diagrams together with the graph of energy function used in the training of the Hopfield network.

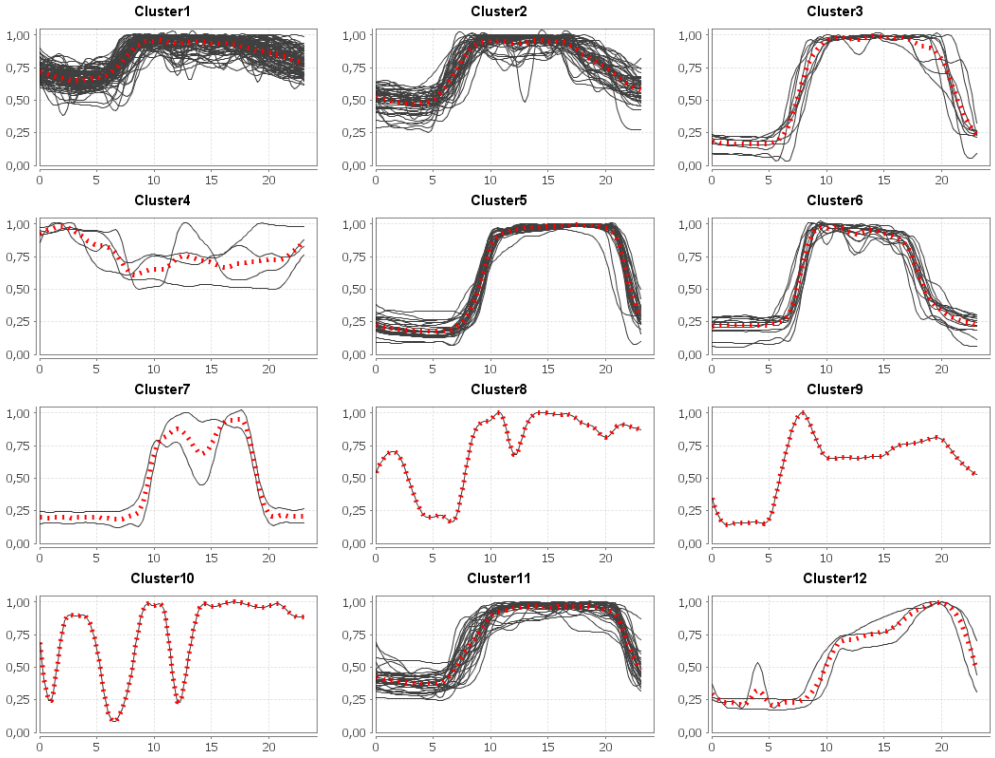

Figure 5.19 Clustering of winter weekdays data by Hopfield-K-means in Euclidean framework.

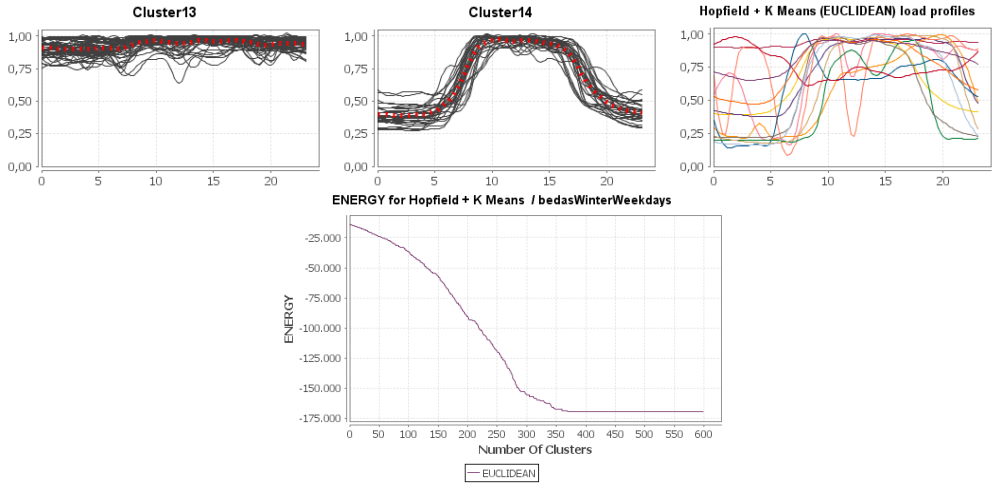

Figure 5.19 Clustering of winter weekdays data by Hopfield-K-means in Euclidean framework. (cont'd)

The algorithm forms large clusters of nearly equal sizes, medium clusters of similar sizes and also groups outliers in separate clusters, as shown in (Fig. 5.19). Note the presence of subtle noise in *Cluster 1* and *Cluster* 2.

Energy function computation over iterations during Hopfield network training is also included in clustering illustration. Energy values steadily decrease where particular knee points are occasionally encountered until convergence, which corresponds to the minimum energy configuration of states.

Summer weekdays data clustering evaluation results are summarized in the following graphs.

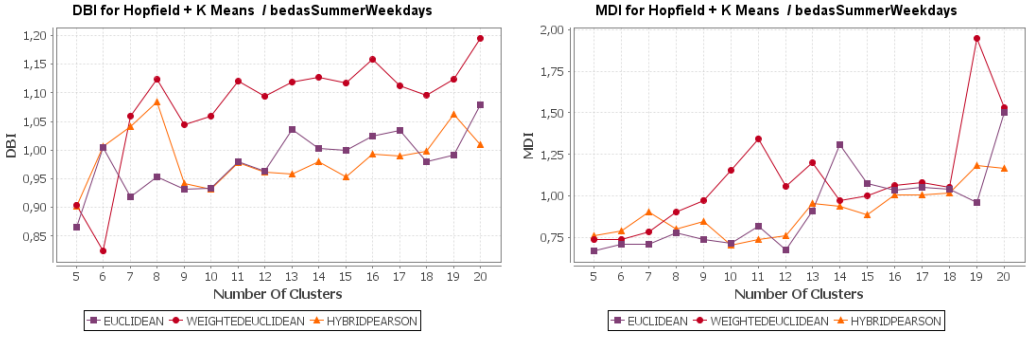

Figure 5.20 Evaluation of Hopfield-K-means for summer weekdays data.

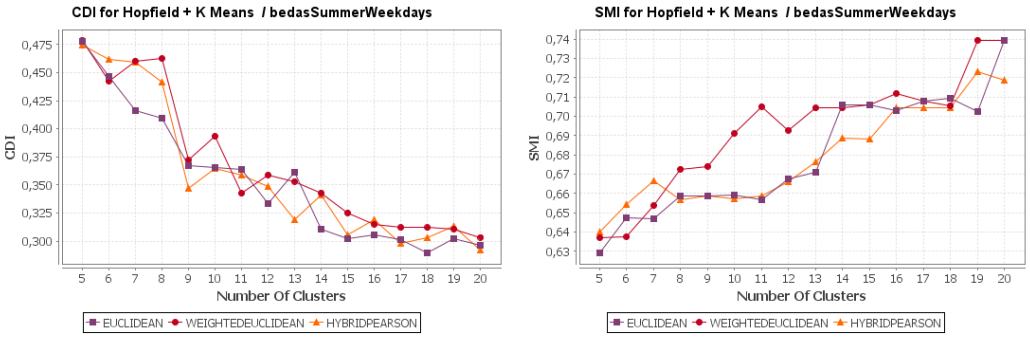

Figure 5.20 Evaluation of Hopfield-K-means for summer weekdays data. (cont'd)

10 clusters in Euclidean framework are slope-changing, moderately low values in DBI, MDI, and CDI curves and this value resides in a local plane structure (step) where surrounding points correspond to more or less the same metric values in the SMI curve, as shown in (Fig. 5.20). Diagrams of summer weekdays data partitioned into 10 clusters are included below.

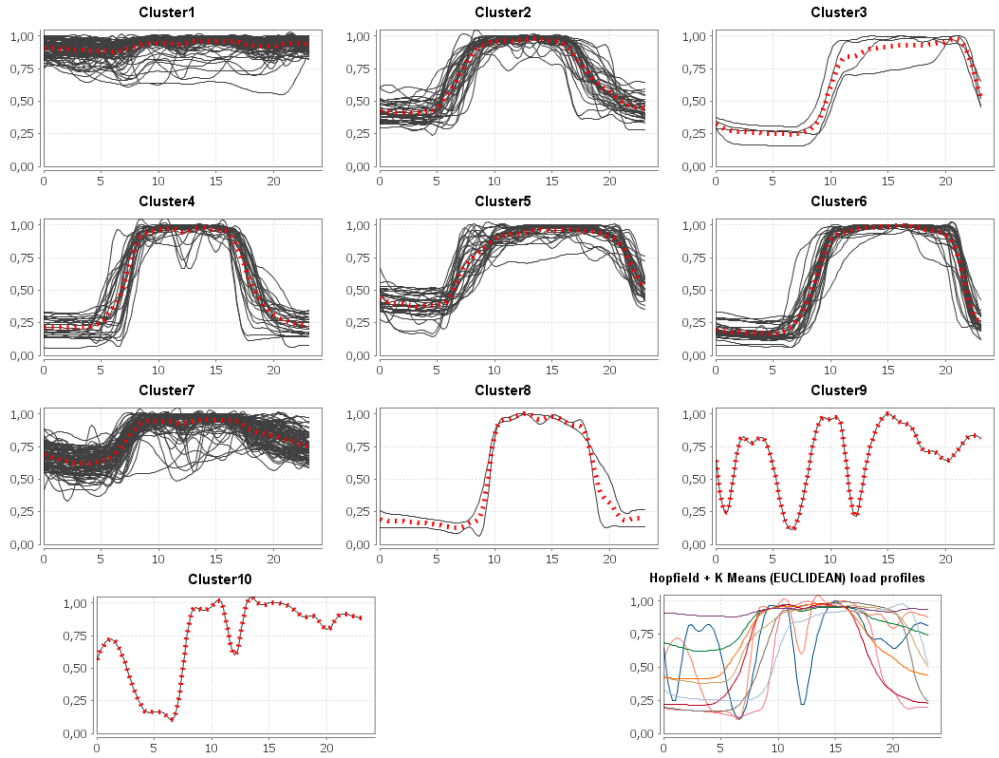

Figure 5.21 Clustering of summer weekdays data by Hopfield-K-means in hybrid Pearson framework.

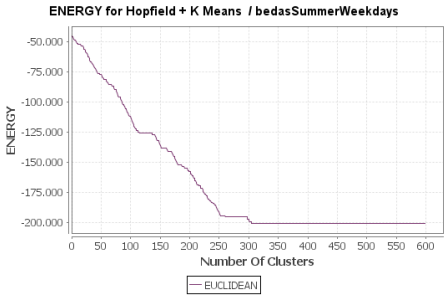

Figure 5.21 Clustering of summer weekdays data by Hopfield-K-means in hybrid Pearson framework. (cont'd)

Large clusters formed by the algorithm contain high-variance elements, possibly due to the incapacity of the algorithm to form the important outlier groupings *Cluster 5* and *Cluster 7* of the summer weekdays data created by K medians method, as shown in (Fig. 5.21).

Convergence of the energy function within maximum number of iterations can be observed from the graph included in (Fig 5.21), where certain shoulder points can be inspected, since some data points may not be assigned to a cluster other than the one it belongs to, if that operation would cause an increase in total energy value at that instant.

5.4.6 K-Medoids Clustering Evaluation

Evaluation of K medoids clustering for winter weekdays data set is summarized in subsequent charts.

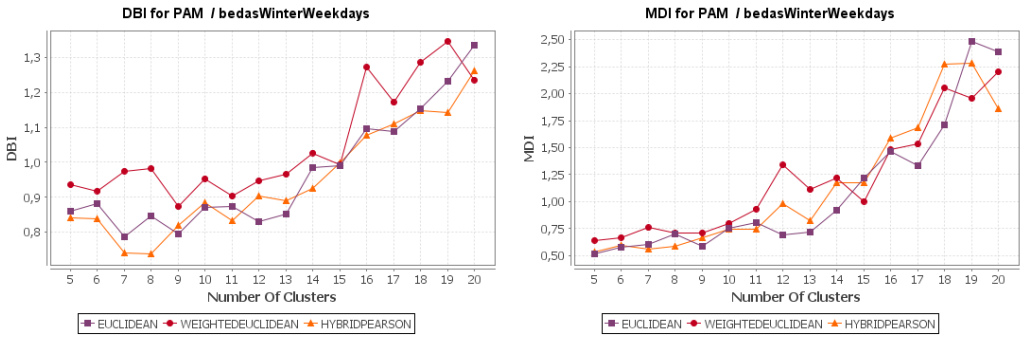

Figure 5.22 Evaluation of K-medoids for winter weekdays data.

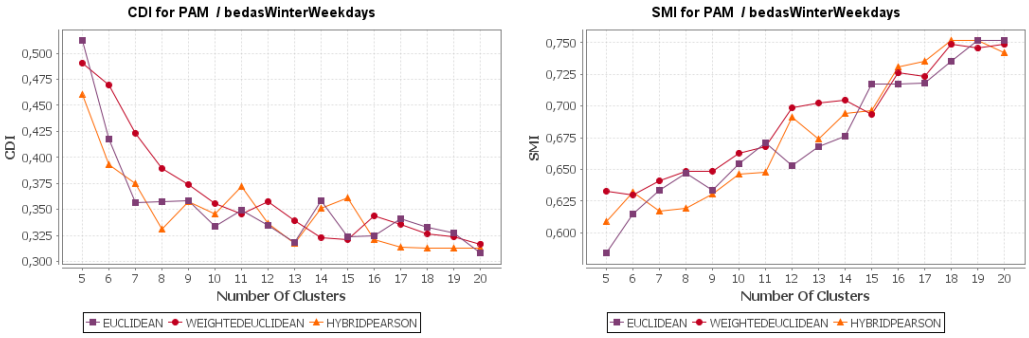

Figure 5.22 Evaluation of K-medoids for winter weekdays data. (cont'd)

DBI, MDI, and SMI metrics in (Fig. 5.22) have their lowers for seven and eight clusters in hybrid Pearson distance framework. Values of indices do not vary considerably between seven and eight clusters. However, CDI values indicate that performance of eight clusters is better than seven clusters. Therefore, winter weekdays data set is grouped into eight clusters using the algorithm in hybrid Pearson distance framework, as the following diagrams illustrate.

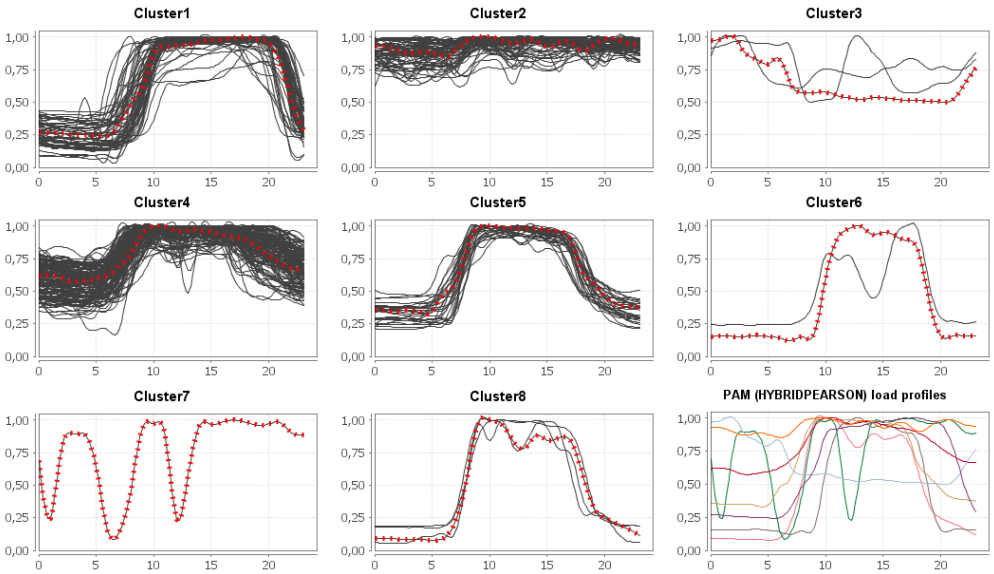

Figure 5.23 Clustering of winter weekdays by K-medoids in hybrid Pearson framework.

The algorithm forms main clusters with relatively small amounts of high-variance data and forms outlier clusters. Although there are no problems with representative powers of cluster medoids in singleton clusters or clusters that have few similar elements such as *Cluster 8* in (Fig. 5.23); medoids may not capture characteristics of each member of clusters consisting of two or three elements that resemble each other in shape, yet may be considered to be distant from one another, such as *Cluster 6* and *Cluster 3*. This is a

shortcoming of the algorithm enforcing one of the data points to be the center within clusters.

On the other hand, in large clusters, such as *Cluster 1*, *Cluster 2*, *Cluster 4*, and *Cluster 5* medoids are robust to data points that deviate considerably from other members of clusters, which turns out to be the greatest advantage of K medoids clustering algorithm.

Evaluation of summer weekdays data clustering is included in the following.

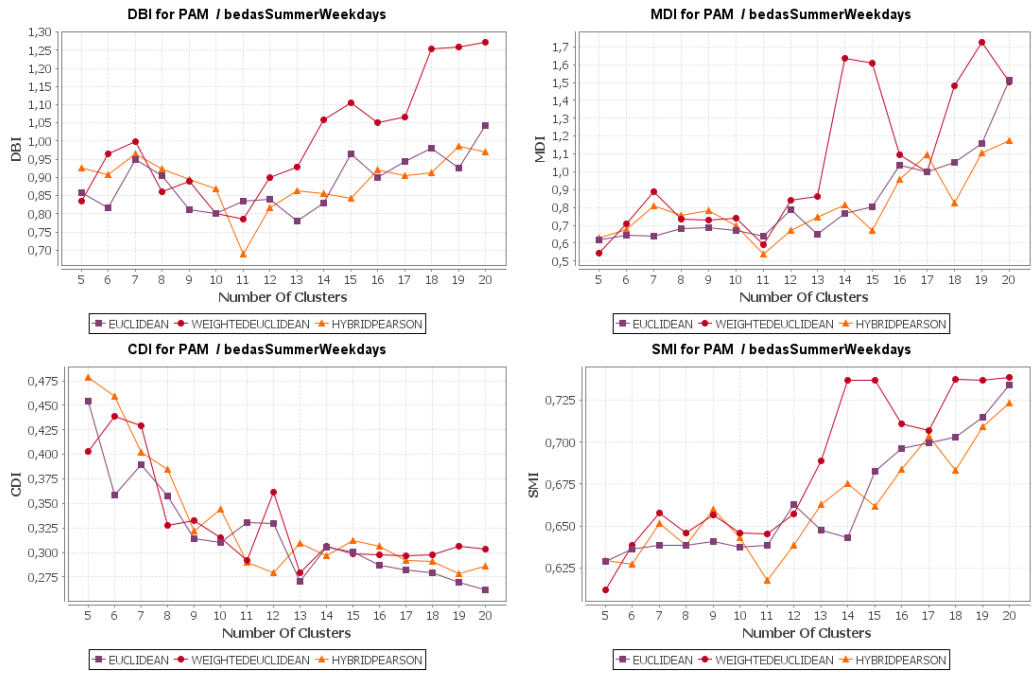

Figure 5.24 Evaluation of K-medoids for summer weekdays data.

DBI, MDI, and SMI metrics agree that 11 clusters have the lowest values in hybrid Pearson distance framework. 11 clusters correspond to the second lowest value in CDI graph as well, as depicted in (Fig. 5.24). Oscillations observed in MDI and SMI metric values after 13 clusters are due to medoid selection in outlier clusters that turns out to be not informative enough, i.e. outliers can be included in one of the large clusters instead.

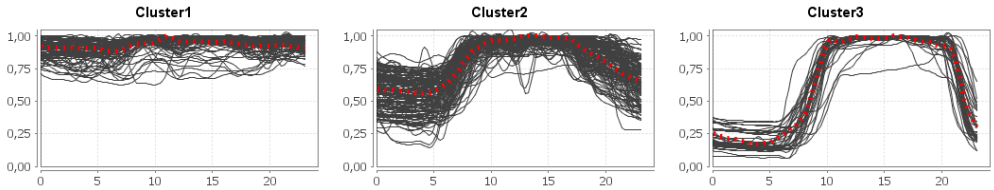

Figure 5.25 Clustering of summer weekdays data by K-medoids in hybrid Pearson framework.

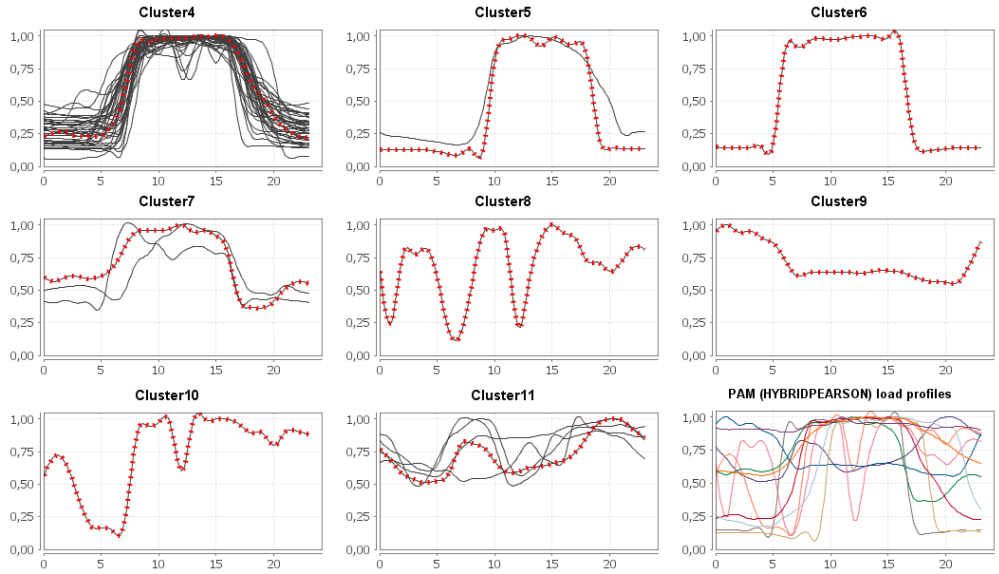

Figure 5.25 Clustering of summer weekdays data by K-medoids in hybrid Pearson framework. (cont'd)

As illustrated in (Fig. 5.25), K-medoids forms four main large clusters with certain amounts of noise that can be considered as moderate in case of *Cluster 1*, and forms outlier clusters, for most of which medoid selection is representative enough. *Cluster 9* out of outlier clusters does not differ greatly from *Cluster* 1, a not-so-compact large cluster. Nonetheless, generated clusters meet the performance standards of good quality clustering.

5.4.7 Similarity-Based K-Means Clustering Evaluation

Evaluation of the performance of the algorithm for clustering the winter weekdays data set is included in the following charts.

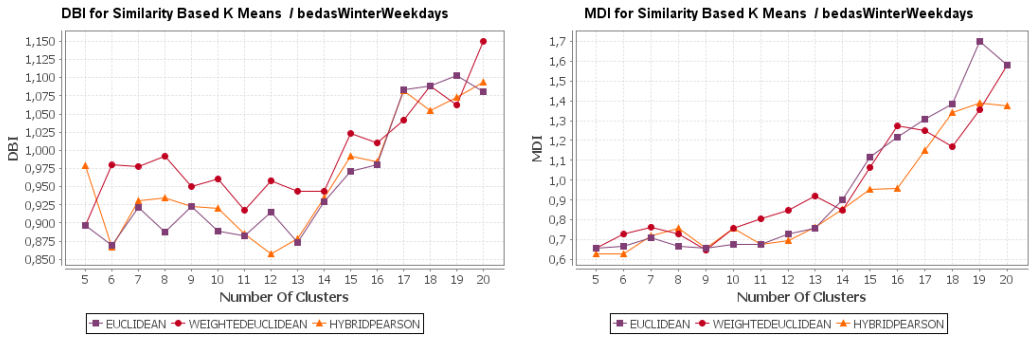

Figure 5.26 Evaluation of similarity-based K-means for winter weekdays data.

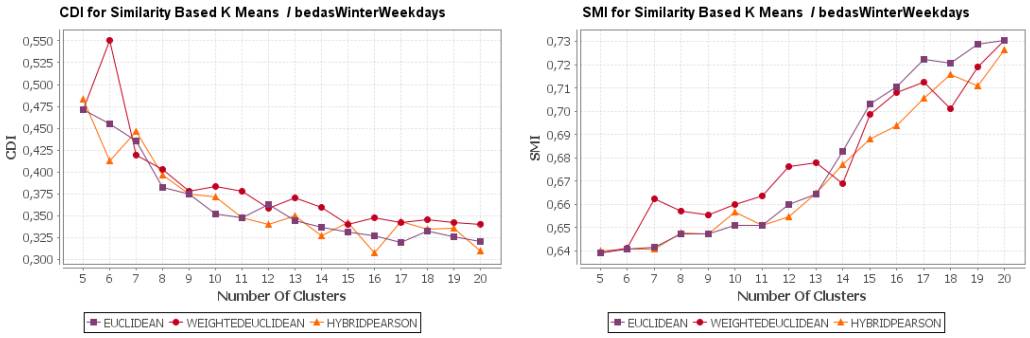

Figure 5.26 Evaluation of similarity-based K-means for winter weekdays data. (cont'd)

Similarity-based K-means algorithm produces high quality clustering until 14 clusters as demonstrated in (Fig. 5.26). DBI values versus number of clusters do not change steadily until 13 clusters in Euclidean framework. After 13 clusters, DBI values increase rapidly. Similarly, for MDI and SMI metrics, 13 clusters is a point after which slopes of the graphs change. Therefore, winter weekdays data is grouped into 13 clusters as in the following diagrams.

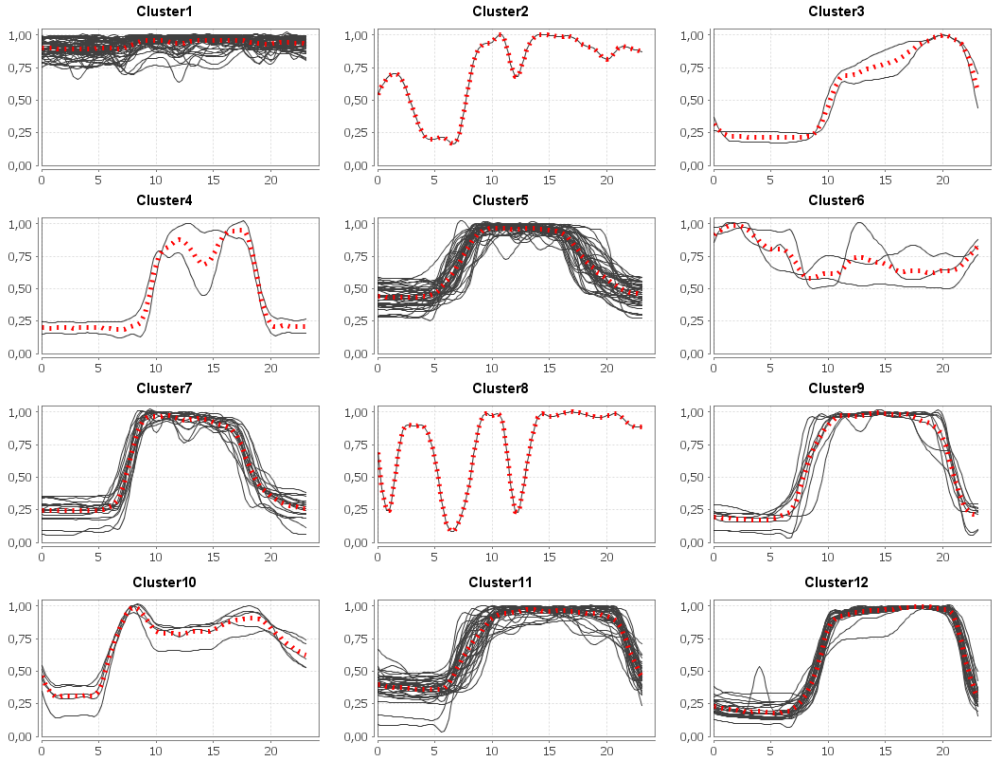

Figure 5.27 Clustering of winter weekdays data by similarity-based K-means in Euclidean framework.

Figure 5.27 Clustering of winter weekdays data by similarity-based K-means in Euclidean framework. (cont'd)

Clustering in (Fig. 5.27) yielded by the algorithm is highly robust to noise. It restricts the source of noise to be data points having similar shapes for most of the data features but distinct levels of consumption for few features such as data variation in *Cluster 11*. Between hours 0 and 5, customers have the same shape of consumption at varying amounts. Outlier clusters correspond to truly different shapes of consumption and therefore are informative.

Evaluation of summer weekdays data is summarized in the charts below.

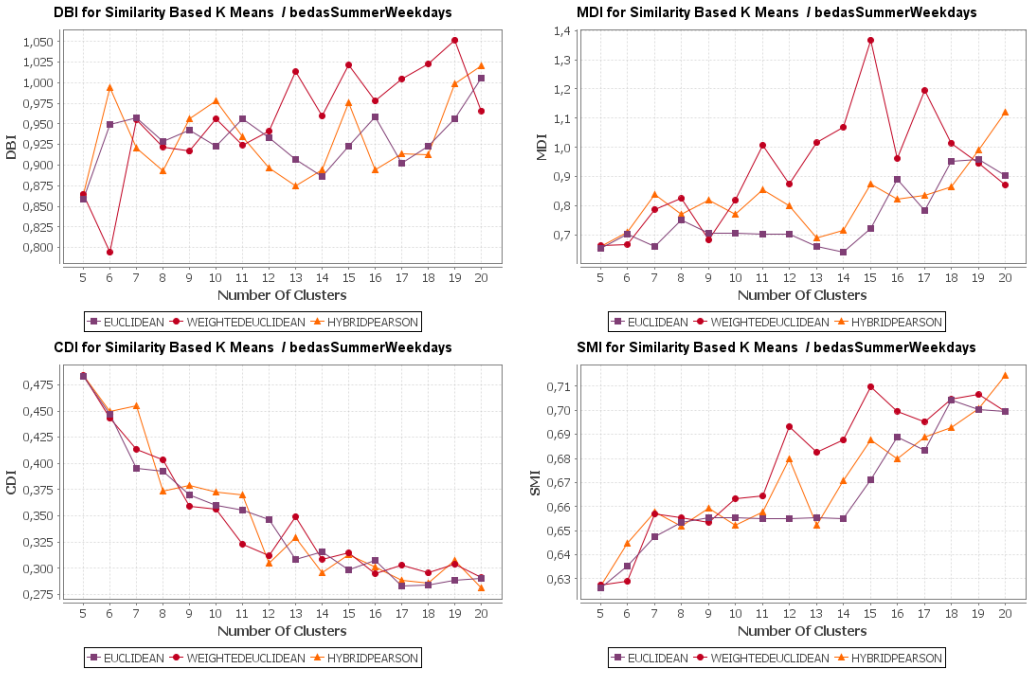

Figure 5.28 Evaluation of similarity-based K-means on summer weekdays data.

DBI, MDI, and SMI metric evaluations in (Fig. 5.28) indicate that 14 clusters in Euclidean framework correspond to locally minimum values. Although this tendency is not supported by the CDI metric, 14 clusters still reside within the acceptable limit according to this metric.

Surprisingly, six clusters in weighted Euclidean metric have the lowest DBI value. However, corresponding clustering results are not included, since six clusters are not informative enough to separate outliers from the rest of the data set. Partitioning summer weekdays into 14 clusters is illustrated as follows.

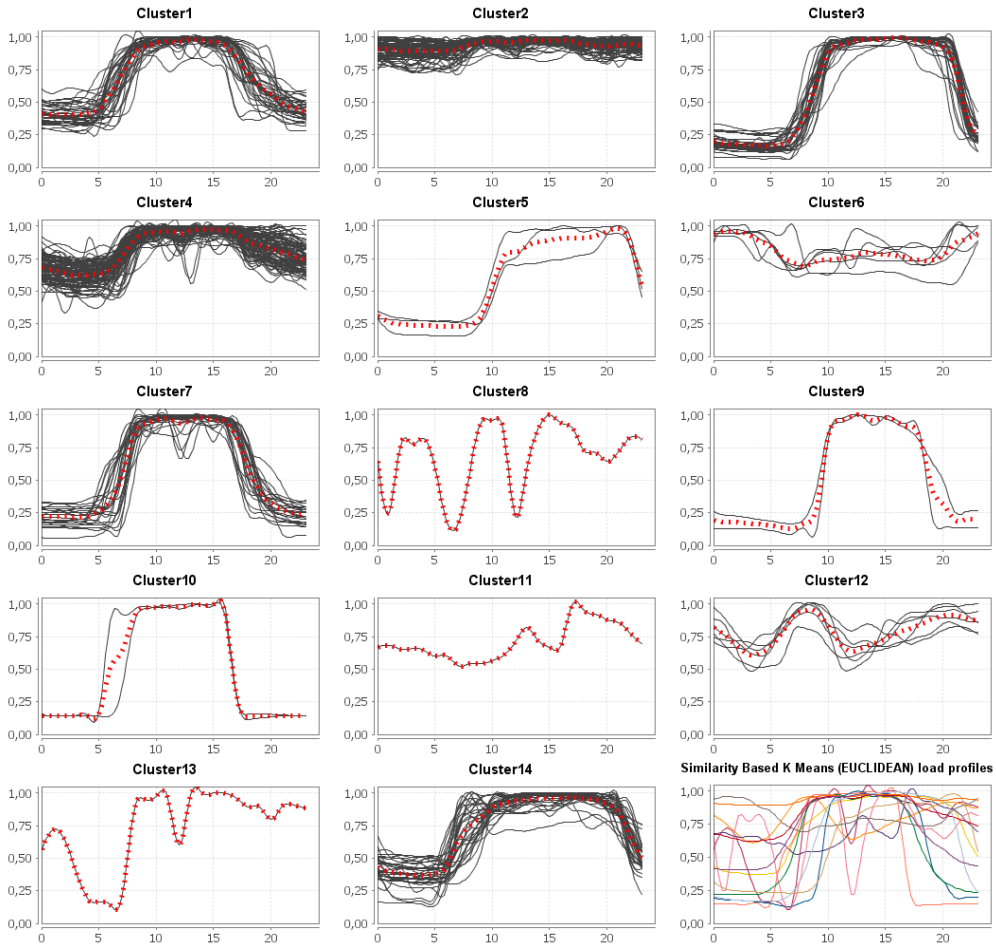

Figure 5.29 Clustering of summer weekdays data by similarity-based K-means in Euclidean framework.

Similarity-based K-means method works by grouping data points with similar shapes together both in large clusters and outlier clusters. Doing so, variation in cluster compositions reduces and performance becomes quite significant in case of large number of clusters such as 13 clusters of winter weekdays in (Fig 5.27) and 14 clusters of summer weekdays in (Fig 5.29). Similarity of data points within clusters can be observed in *Cluster 12*, *Cluster 10*, *Cluster 9*, *Cluster 6*, and *Cluster 5* as outlier clusters in (Fig. 5.29).

5.4.8 ISODATA Clustering Evaluation

Performance of the algorithm on winter weekdays data set is summarized in the following charts.

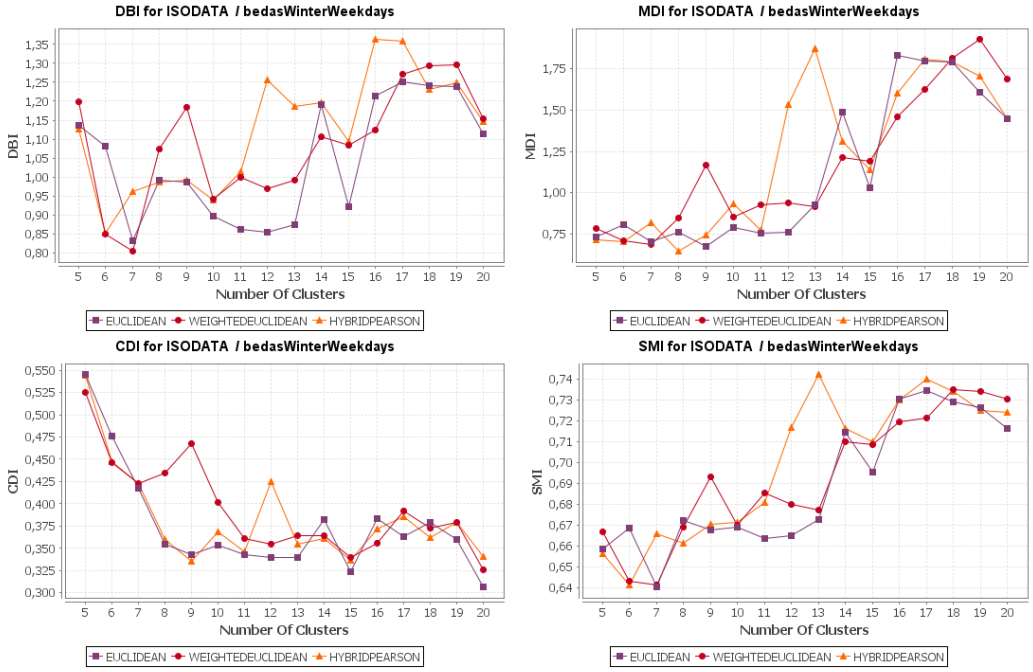

Figure 5.30 Evaluation of ISODATA clustering on winter weekdays data.

Unsteady behaviors observed in the charts of validity metrics in (Fig. 5.30) are due to the fact that final clusters are generated by either merging or splitting large or small number of initial clusters. Due to evaluation methodology, better-performing cluster formations are kept, which may have been created only by either the merge or the split operation.

Seven clusters in Euclidean distance framework are chosen to partition summer weekdays data set as shown in the following, because they correspond to generally low DBI, MDI and SMI values, and a moderate value of CDI metric.

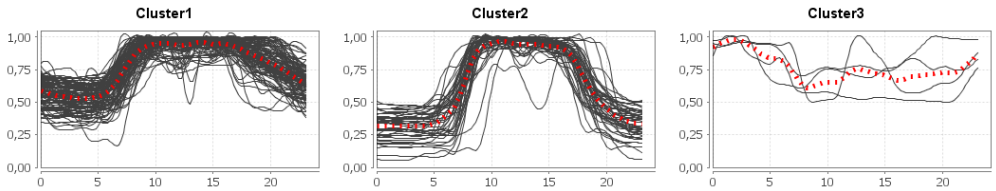

Figure 5.31 Clustering of winter weekdays data by ISODATA in Euclidean framework.

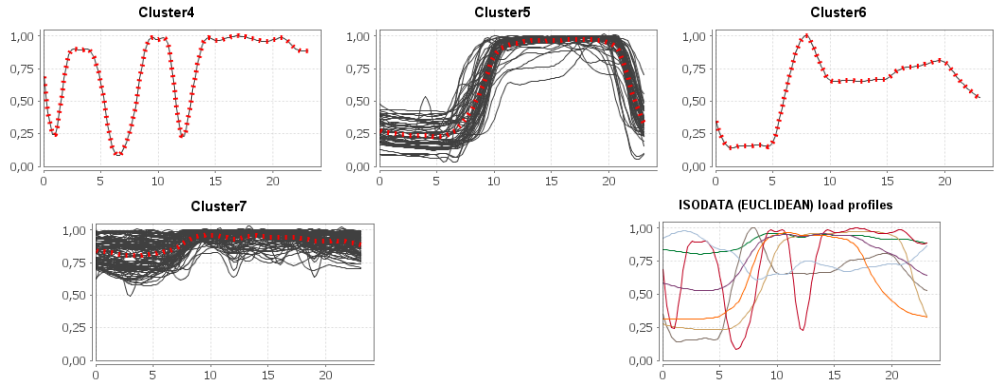

Figure 5.31 Clustering of winter weekdays data by ISODATA in Euclidean framework. (cont'd)

ISODATA forms densely populated main clusters of similar sizes including less within cluster scatter as illustrated in (Fig. 5.31). Outlier clusters yielded by ISODATA is considered to be informative, preventing the amount of noise from increasing within large clusters if outlier elements were to be added to them.

Summer weekdays data set clustering evaluation is presented in subsequent charts.

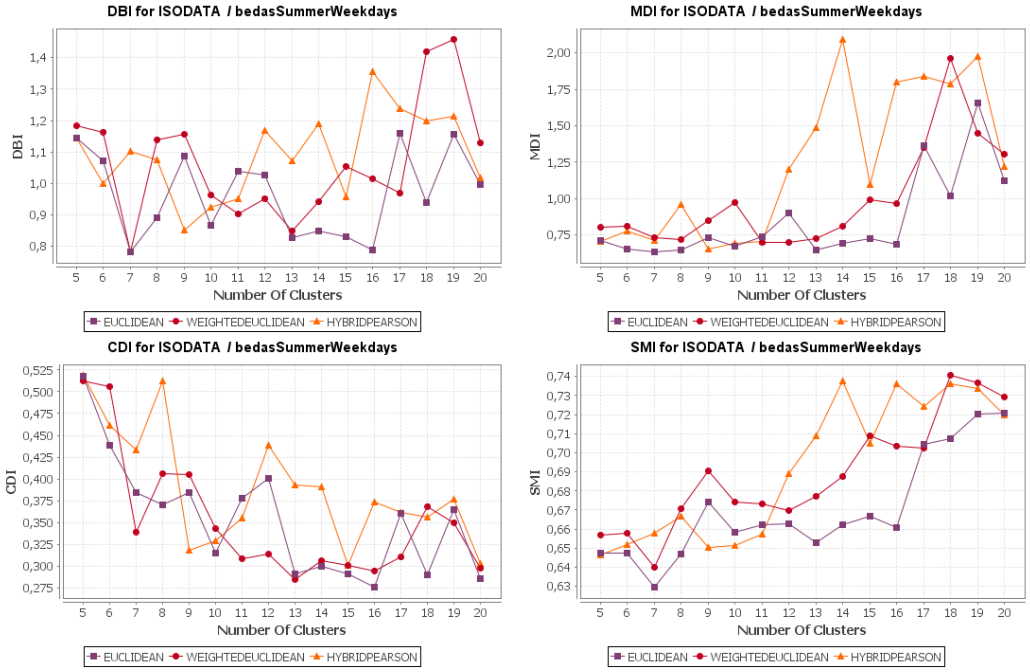

Figure 5.32 Evaluation of ISODATA on summer weekdays data.

Seven clusters in Euclidean distance metric correspond to local minimum values for DBI, MDI, and SMI curves in (Fig. 5.32). The value of seven clusters is also a knee point for

CDI metric. Therefore, seven clusters are selected to represent partitioning of summer weekdays data.

It should be noted that performance of weighted Euclidean metric generally supersedes that of hybrid Pearson, and in some cases the performance of Euclidean metric. However, it tends to show substantial unsteady behavior as indicated by DBI and CDI graphs.

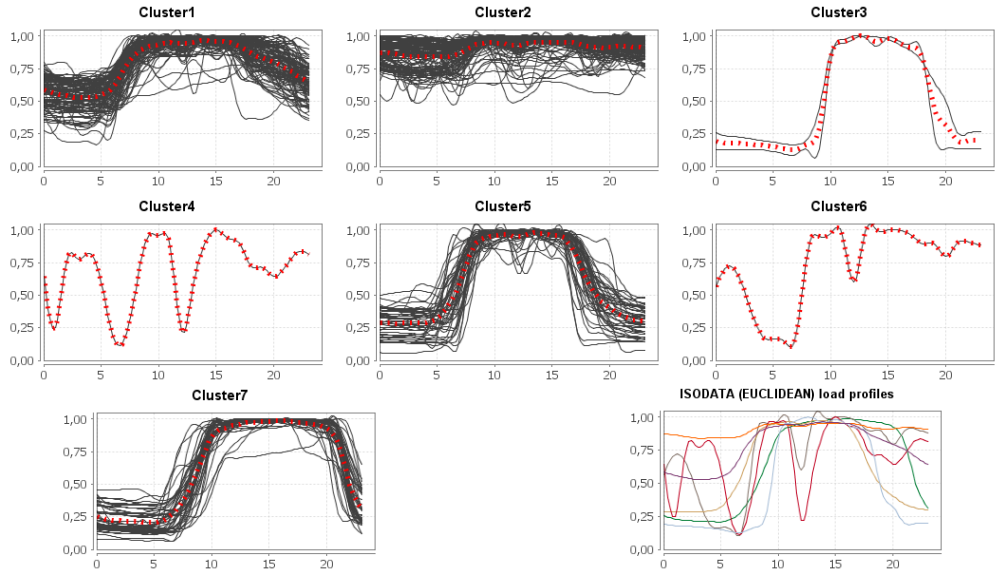

Figure 5.33 Clustering of summer weekdays data by ISODATA in Euclidean framework.

Main large clusters, *Cluster1* and *Cluster* 2 in (Fig 5.33) contain considerable amount of noise, in other words data points within these clusters are notably scattered. Outlier clusters yielded by the algorithm are informative in sense that their presence in any other of the large clusters would decrease the overall compactness of clusters.

5.4.9 Fuzzy K-Means Clustering Evaluation

Winter weekdays clustering by Fuzzy K-means with a fuzziness factor of 2 is included subsequently.

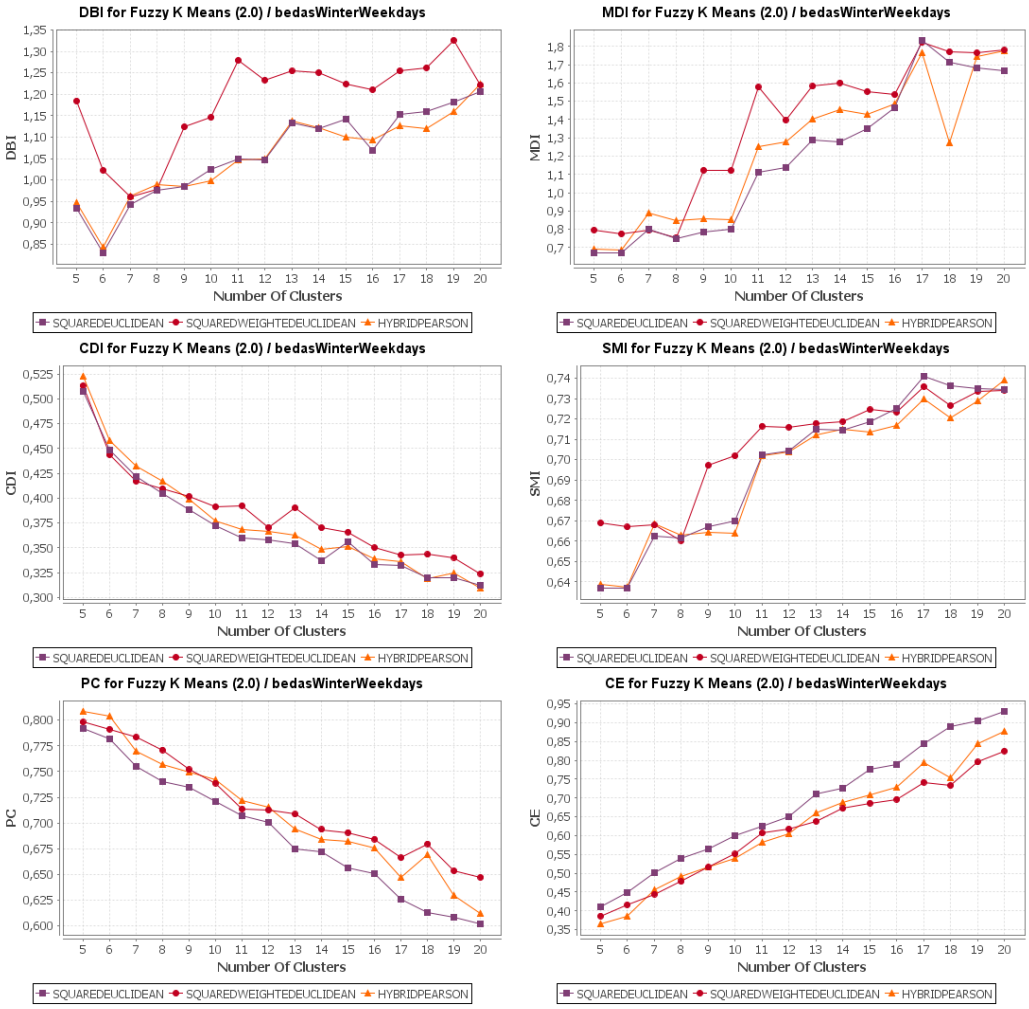

Figure 5.34 Evaluation of Fuzzy K-means on winter weekdays data.

Six clusters for winter weekdays in squared Euclidean framework are awarded with lower values of DBI, MDI, and SMI and the greatest decline in CDI metric value, as the curves in (Fig. 5.34) indicate. Moreover, since higher values of PC and lower values of CE are desirable so that belonging of a data point to its assigned cluster is not confronted with doubts, selecting six clusters appears to be reasonable.

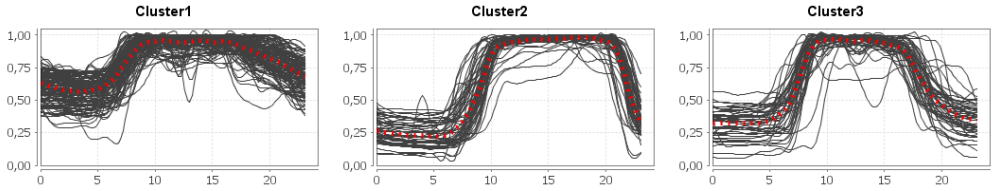

Figure 5.35 Clustering of winter weekdays data by Fuzzy K-means in squared Euclidean framework.

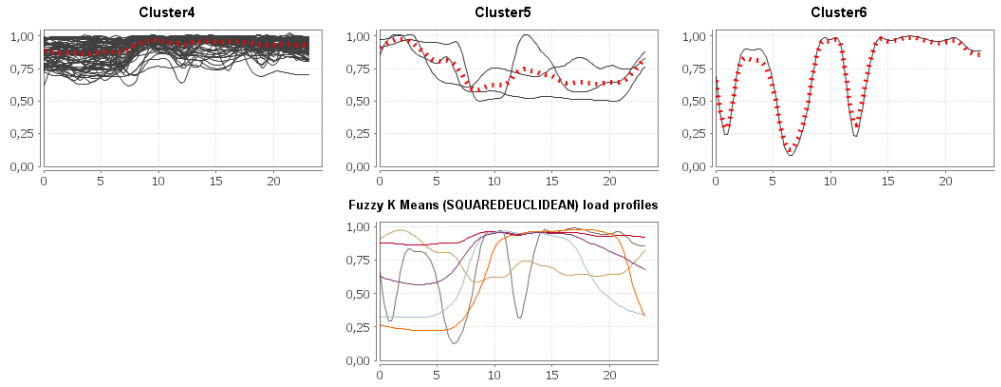

Figure 5.35 Clustering of winter weekdays data by Fuzzy K-means in squared Euclidean framework. (cont'd)

Fuzzy K-means clustering illustrated in (Fig. 5.35) yields large clusters some of which contain moderate amounts of noise such as in *Cluster 3*. Note that although *Cluster 6* contains only one element, its centroid diverges slightly from its sole element due to the fact that centroid computation involves all data points in the set.

Summer weekdays clustering by the algorithm is evaluated as follows.

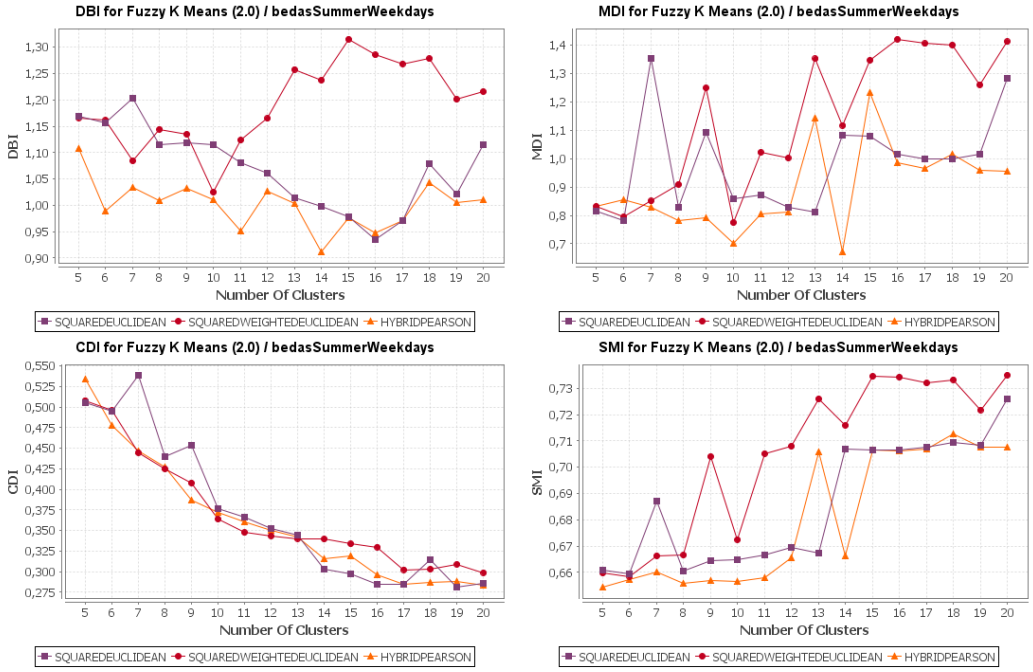

Figure 5.36 Evaluation of Fuzzy K-means on summer weekdays data.

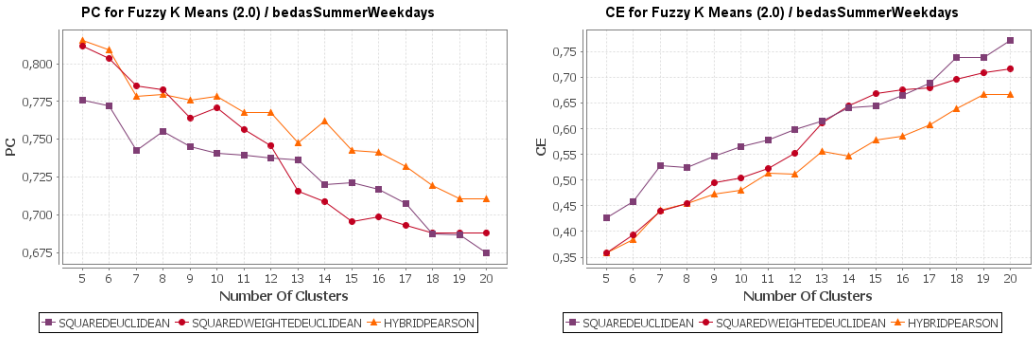

Figure 5.36 Evaluation of Fuzzy K-means on summer weekdays data. (cont'd)

Partition of summer weekdays data set into 14 clusters in a hybrid Pearson framework achieves lowest values of DBI and MDI metrics as shown in (Fig. 5.36). 14 clusters also attain a local minimum value of SMI and constitute a slope-changing point in the chart of CDI with a low value of the metric. This number of clusters is further promoted by relatively high values of PC and low values of CE measures. Hence, corresponding clustering is included in the following diagrams.

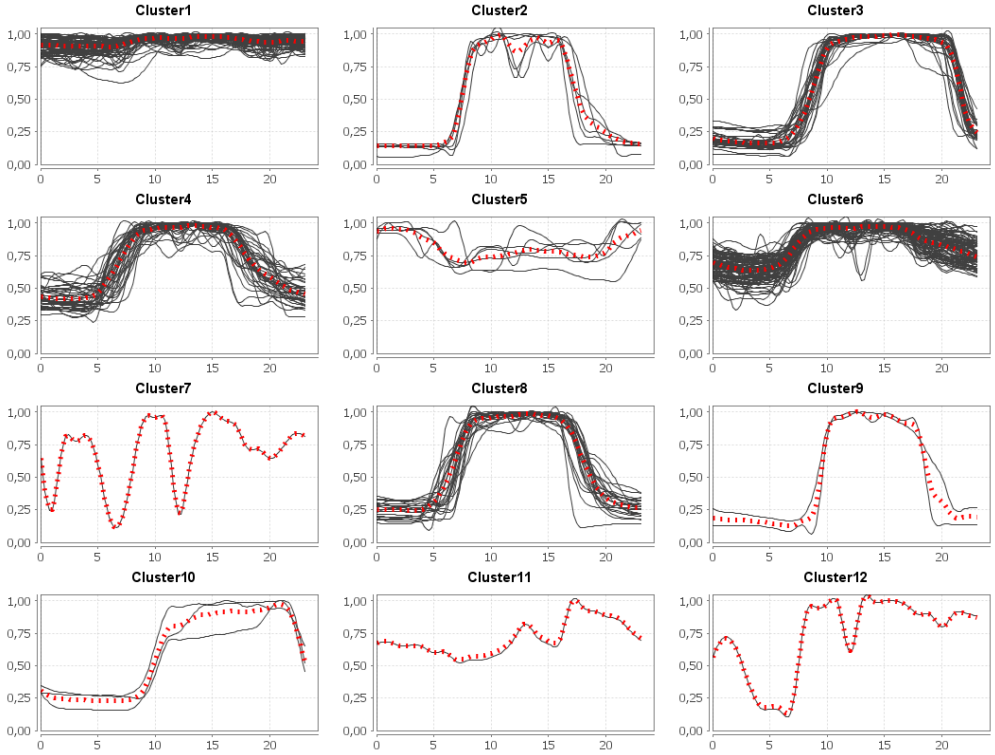

Figure 5.37 Clustering of summer weekdays data by Fuzzy K-means in hybrid Pearson framework.

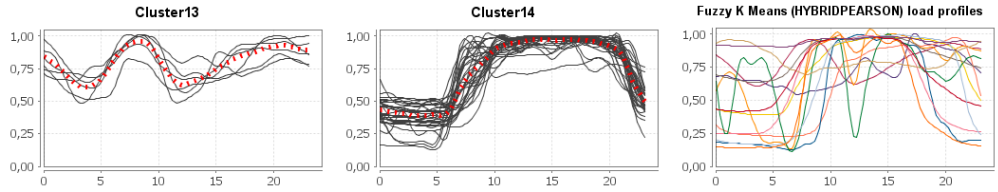

Figure 5.37 Clustering of summer weekdays data by Fuzzy K-means in hybrid Pearson framework. (cont'd)

Clustering of summer weekdays data as depicted in (Fig. 5.37) reveals well-formed outlier clusters, most of which are non-singleton and have successfully gathered similar elements. Data scatter within cluster is somewhat high in *Cluster 6*, yet majority of the large clusters are quite compact.

5.4.10 Follow-The-Leader Clustering Evaluation

Winter weekdays data set clustering by the algorithm is evaluated in the following charts.

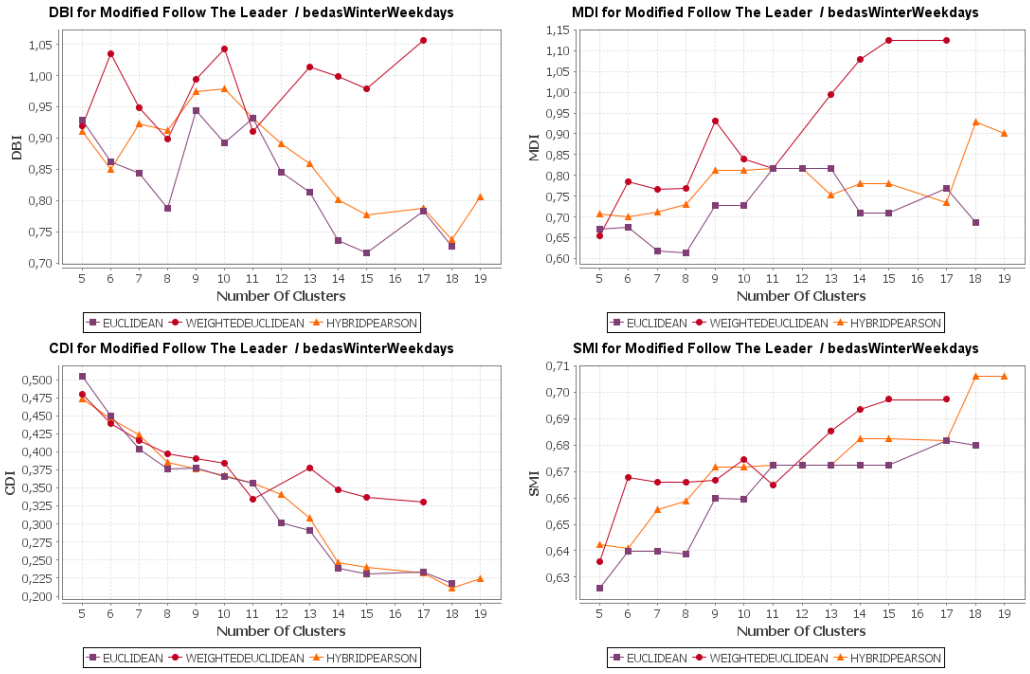

Figure 5.38 Evaluation of Follow-the-Leader clustering on winter weekdays data.

Follow the leader clustering of winter weekdays data set into 5-to-20 clusters achieves the smallest interval for DBI and MDI metrics compared to the clustering algorithms that have been examined so far, according to the graphs in (Fig. 5.38). Almost every clustering carried out by the algorithm corresponds to good performance results.

Note that charts do not include values for some number of clusters. This is due to distance threshold utilized by the algorithm. Although evaluation methodology for Follow-the-Leader method searches narrow threshold intervals for finding different number of clusters, particular number of clusters could not be yielded due to data set characteristics.

Follow-the-Leader algorithm originally comes with weighted Euclidean distance metric in the name of *modified follow-the-leader* as examined in the literature review of load profile classification in Chapter 2. However, on winter weekdays data set, Euclidean metric outperforms the latter. Eight clusters in a Euclidean distance framework is adopted for clustering demonstration, since this value corresponds to a local minimum in DBI and MDI curves and slope-changing points in CDI and SMI graphs. Corresponding clustering is included as follows.

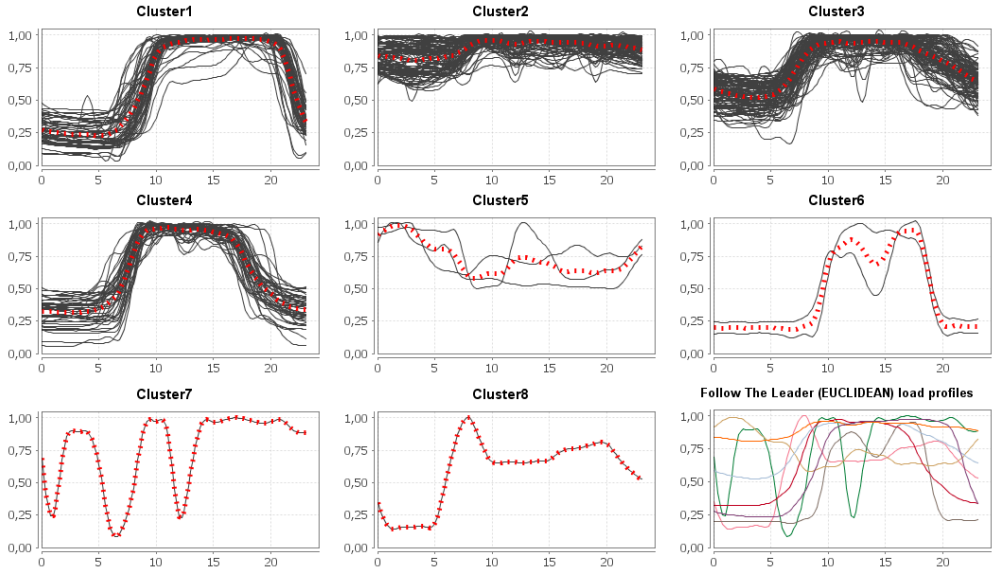

Figure 5.39 Clustering of winter weekdays data by Follow-the–Leader method in Euclidean framework.

The algorithm organizes data points into main large clusters and outlier clusters as illustrated in (Fig. 5.39). Follow the leader method works by iterating over data points. In this case, small outlier clusters follow the large clusters due to the order of neighborhoods of the data points in the set with respect to the distance threshold. Although some data points may have changed clusters in succeeding few iterations of the algorithm, this reflects a tendency of the winter weekdays data set.

Summer weekdays data clustering evaluation is included in the following charts.

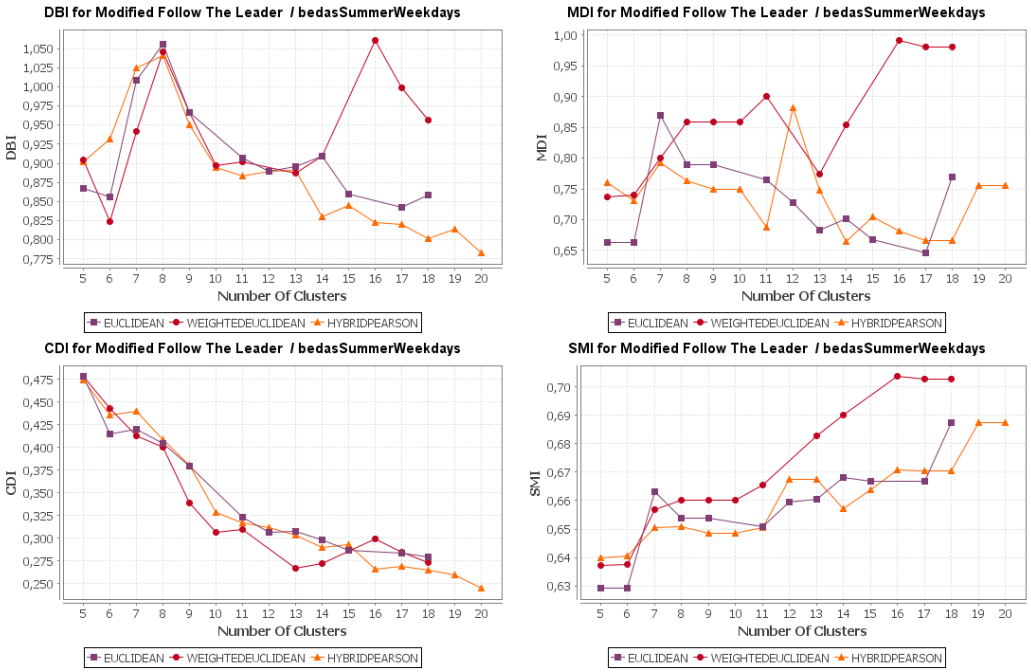

Figure 5.40 Evaluation of Follow-the-Leader clustering on summer weekdays data.

14 clusters in hybrid Pearson framework are selected for exemplifying Follow-the-Leader clustering, because this value corresponds to local minima of DBI and MDI charts and is within the range of low values of CDI and SMI metrics, whose graphs are included in (Fig. 5.40). Corresponding clustering is included subsequently.

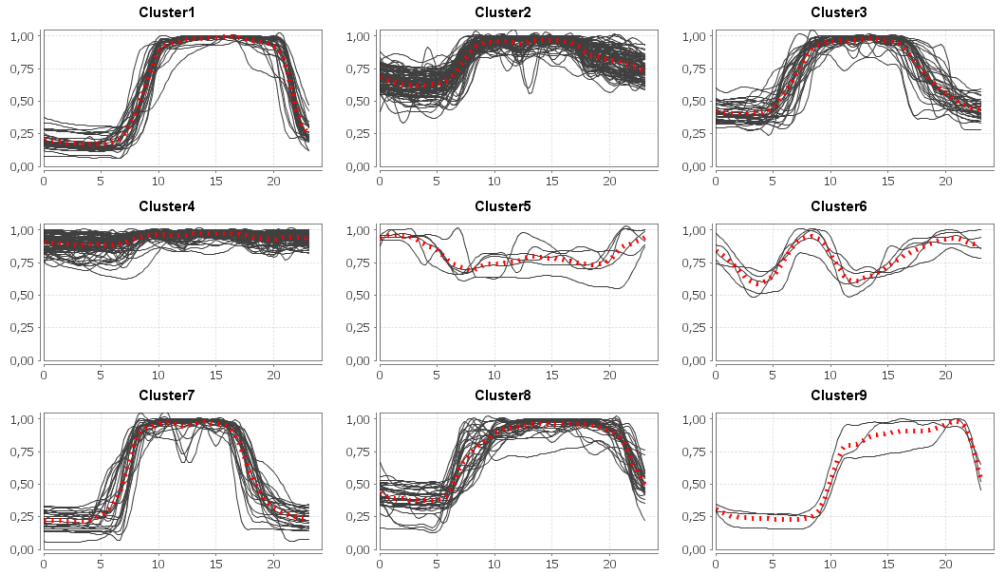

Figure 5.41 Clustering of summer weekdays data by Follow-the-Leader algorithm in hybrid Pearson framework.

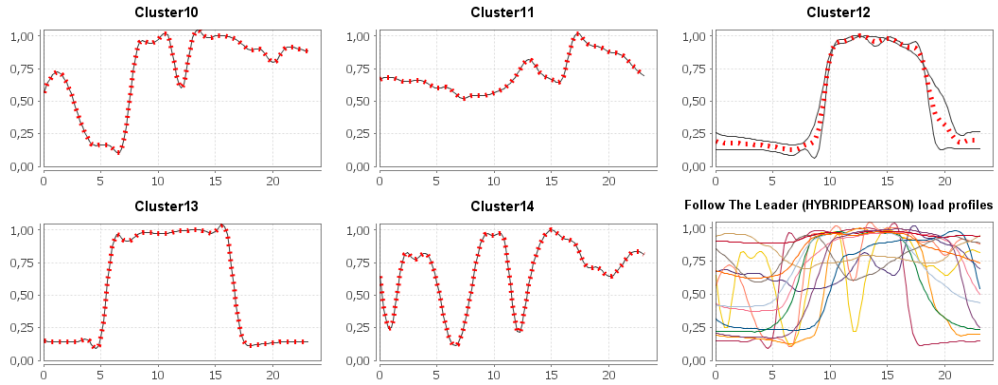

Figure 5.41 Clustering of summer weekdays data by Follow-the-Leader algorithm in hybrid Pearson framework. (cont'd)

Clustering of summer weekdays data set in (Fig. 5.41) reveals outliers together with large and medium-sized clusters. Small clusters tend to follow the latter, although there is no particular order between large and medium-sized clusters.

5.4.11 Hierarchical Clustering Evaluation

Hierarchical clustering offers a variety of linkage criteria. In this section, all linkage criteria but UPGMA and flexible linkage are tested solemnly on winter weekdays data set, so as to reduce the volume of this chapter. UPGMA and flexible linkage criteria are evaluated on both data sets.

5.4.11.1 Single Linkage Clustering Evaluation

Performance of Hierarchical clustering with single linkage for winter weekdays data set is evaluated as follows.

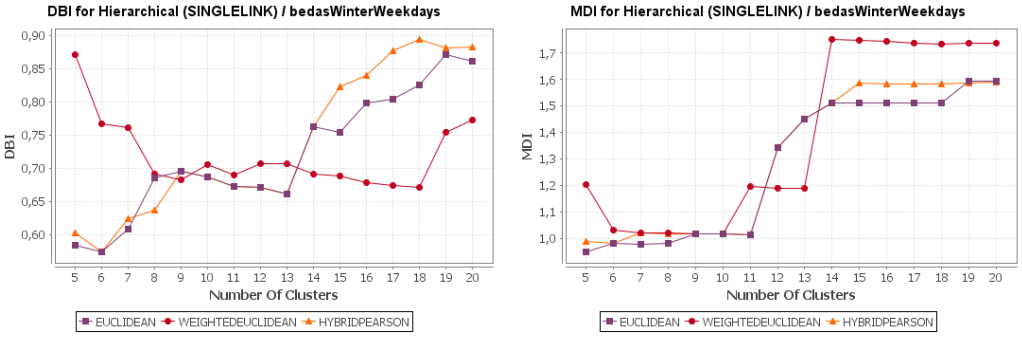

Figure 5.42 Evaluation of Hierarchical clustering with single linkage on winter weekdays data.

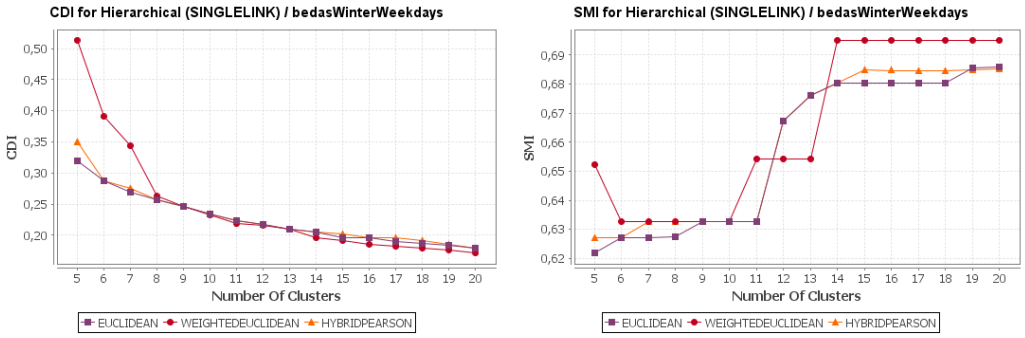

Figure 5.42 Evaluation of Hierarchical clustering with single linkage on winter weekdays data. (cont'd)

Hierarchical clustering is a deterministic algorithm, consequently reducing the intrinsic errors of evaluation methodology. All distance frameworks come to rendezvous point at nine clusters on curves of DBI, MDI, and SMI metrics in (Fig. 5.42). The values of the CDI metric for all distance measures are nearly the same following the rendezvous point. Since weighted Euclidean measure only very slightly yields better performance results, winter weekdays data set grouping into nine clusters is presented in this framework in the following.

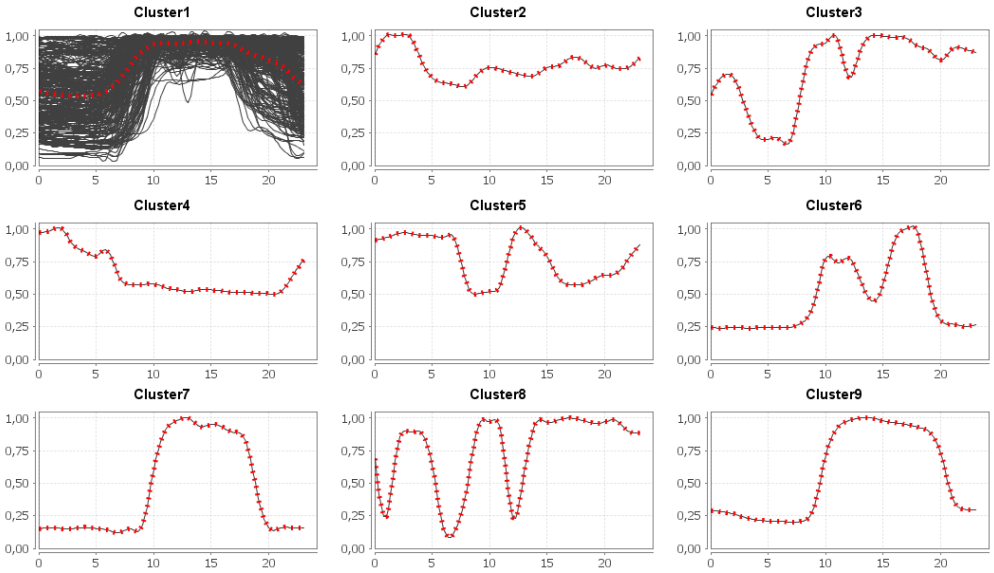

Figure 5.43 Clustering of winter weekdays data by Hierarchical clustering with single linkage in weighted Euclidean framework.

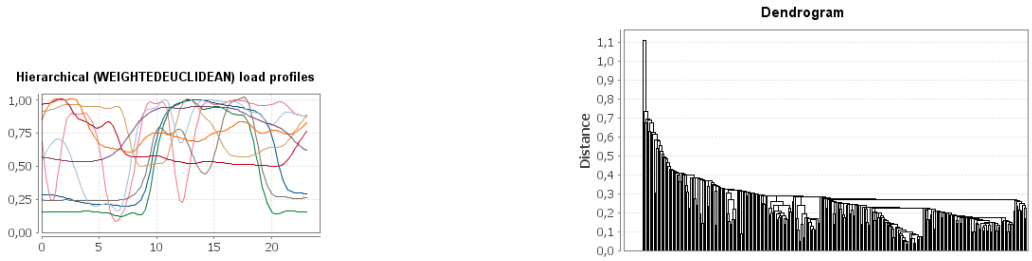

Figure 5.43 Clustering of winter weekdays data by Hierarchical clustering with single linkage in weighted Euclidean framework. (cont'd)

Single link clustering combines nearest cluster pairs as it progresses. At cutoff level, an overly large cluster accompanied by singleton clusters is obtained. Dendrogram included in (Fig. 5.43) illustrates the progress of merge operations, pointing out how quickly clusters combine to form one exceptionally large cluster.

Most of the singleton clusters have their members categorized as outliers in prior algorithms. However, *Cluster7* and *Cluster 9* had never appeared as singleton outlier clusters. Inheriting knowledge from the performance of other algorithms, it cannot be said that single linkage criterion effectively groups outliers in suitable clusters. On the contrary, the algorithm combined with this linkage criterion only tends to separate singleton clusters from the rest of the data set that is grouped in a massive cluster.

5.4.11.2 Complete Linkage Clustering Evaluation

Winter weekdays data set clustered by the algorithm employing complete linkage is evaluated as follows.

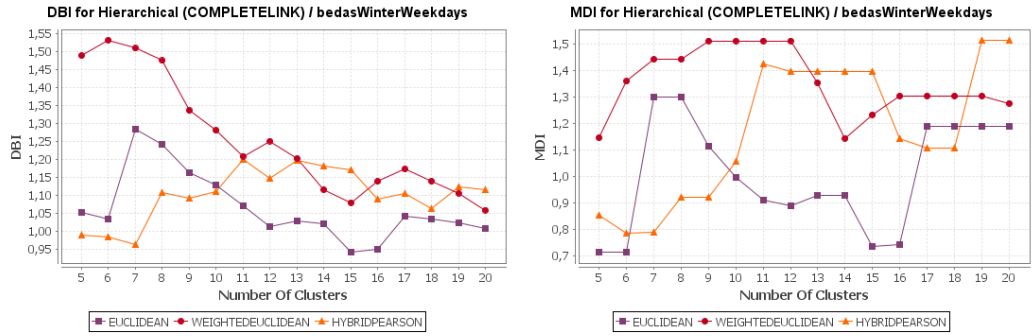

Figure 5.44 Evaluation of Hierarchical clustering with complete linkage on winter weekdays data.

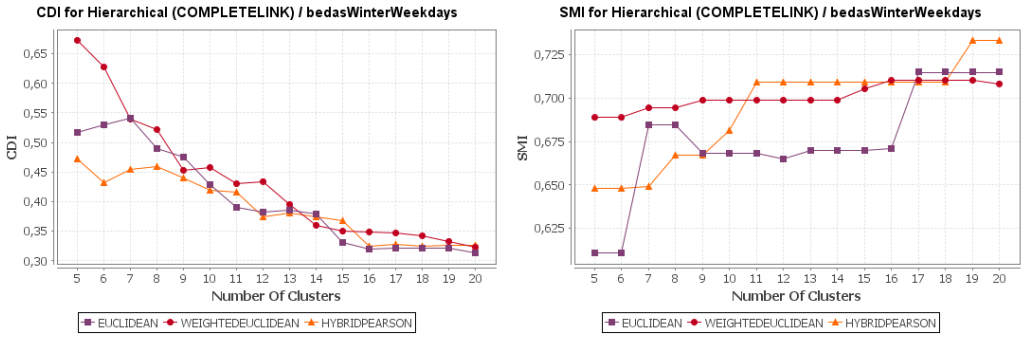

Figure 5.44 Evaluation of Hierarchical clustering with complete linkage on winter weekdays data. (cont'd)

12 clusters in a Euclidean framework are local minima in DBI, MDI, and CDI curves included in (Fig. 5.44). This value also corresponds to a minimum value in the second floor of SMI curve. Hence, 12 clusters are adopted to represent winter weekdays as illustrated subsequently.

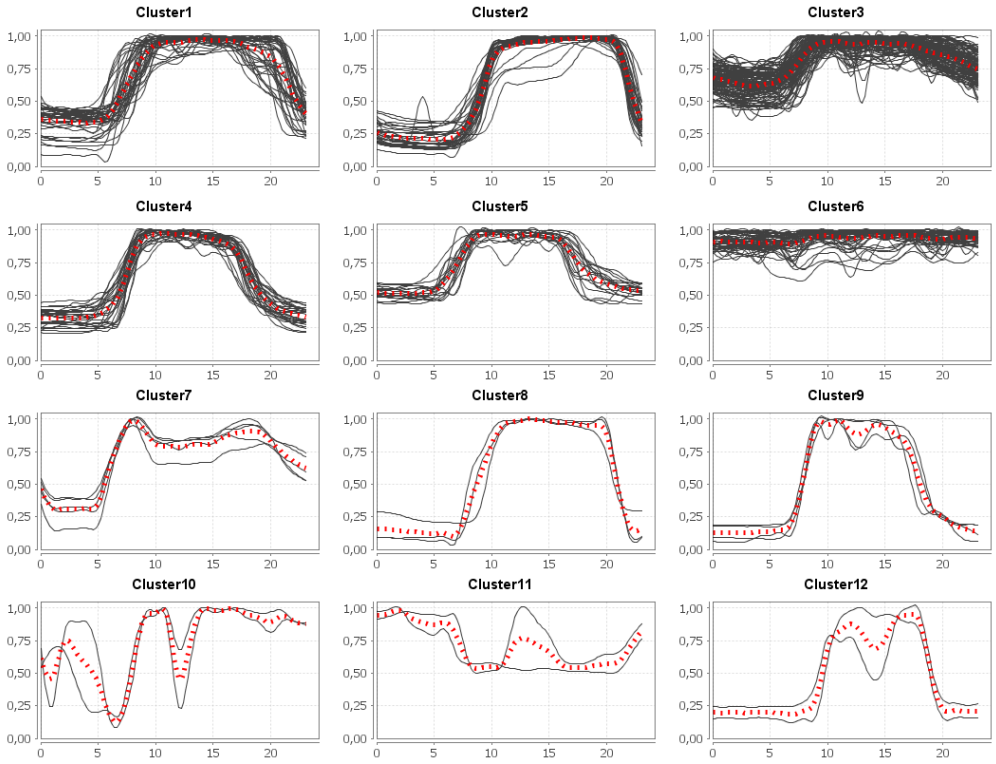

Figure 5.45 Clustering of winter weekdays data by Hierarchical clustering with complete linkage in weighted Euclidean framework.

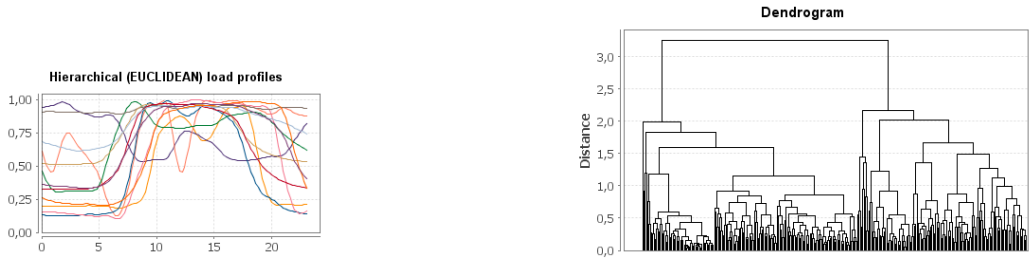

Figure 5.45 Clustering of winter weekdays data by Hierarchical clustering with complete linkage in weighted Euclidean framework. (cont'd)

Unlike single linkage, complete linkage tends to avoid singleton outlier clusters. As the formation process summarized in the dendrogram in (Fig. 5.45) demonstrates, complete linkage criterion forms large clusters of similar sizes, while leaving outliers with few elements. This may result in existence of certain proportions of scattered data in large clusters, as in case of *Cluster 3* and *Cluster 6*.

Once assigned, a data point cannot change its cluster within successive cycles of Hierarchical clustering. Only the cluster it belongs to may expand. This tendency explains the similarity of *Cluster 2* and *Cluster 8*, while their sizes clearly differ.

Since complete linkage does not favor singleton clusters, it can actually misplace data points by merging their clusters at some level, which is irreversible. This is a probable source of scatter in large clusters.

5.4.11.3 WPGMA Linkage Clustering Evaluation

Hierarchical clustering with WPGMA linkage has its performance evaluated on winter weekdays data as shown in the following validity index curves.

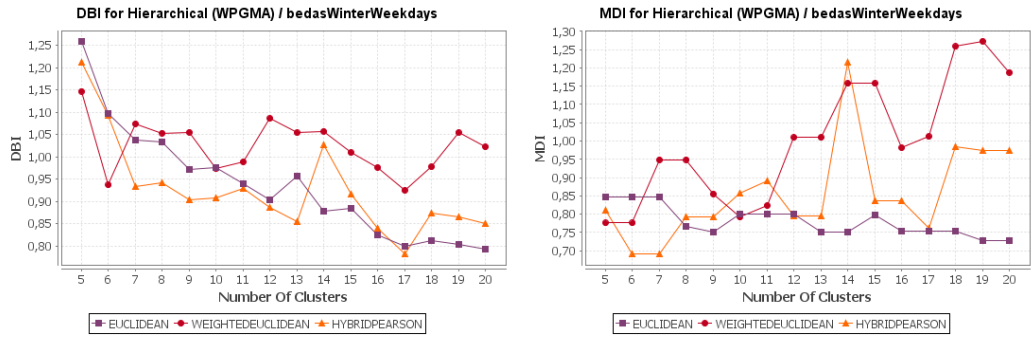

Figure 5.46 Evaluation of Hierarchical clustering with WPGMA linkage on winter weekdays data.

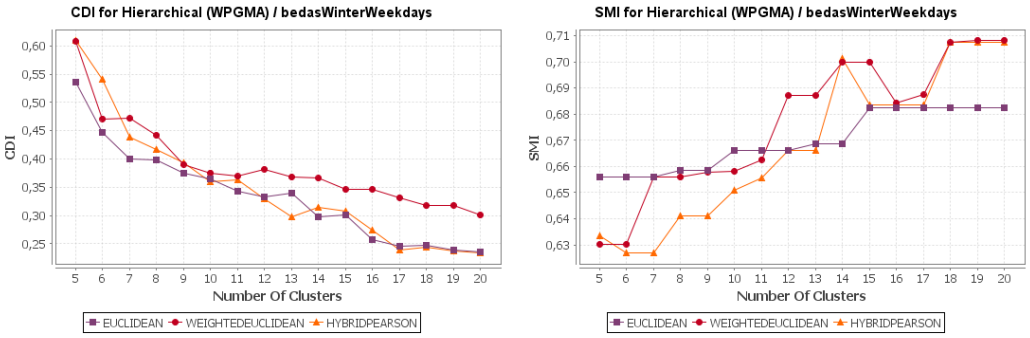

Figure 5.46 Evaluation of Hierarchical clustering with WPGMA linkage on winter weekdays data. (cont'd)

Seven clusters in hybrid Pearson metric are slope-changing points for all indices, and a minimum value for MDI and SMI indices in (Fig. 5.46). Hence, winter weekdays are grouped into seven clusters by the algorithm using WPGMA linkage as follows.

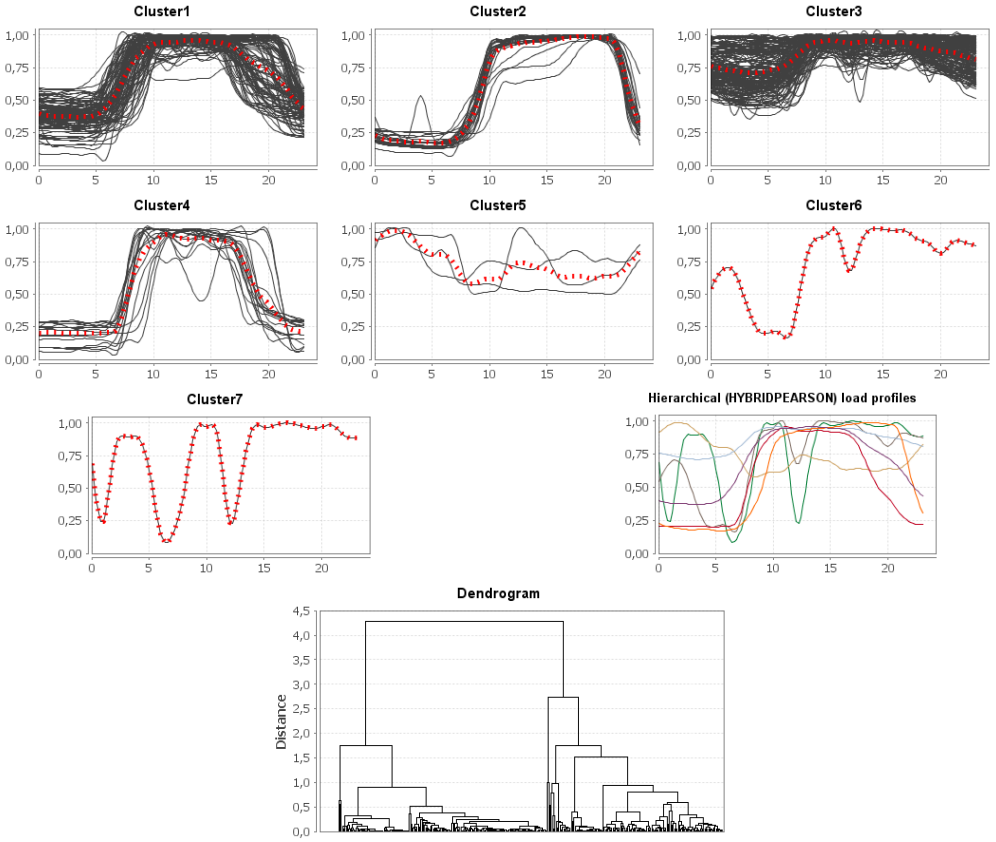

Figure 5.47 Clustering of winter weekdays data by Hierarchical clustering with WPGMA linkage in hybrid Pearson framework.

WPGMA linkage creates main clusters that are either over-populated or under-populated, together with some outlier clusters as in (Fig. 5.47). The history of formation of large

clusters can be tracked via the dendrogram by focusing on parts of the tree around leaves, where smaller clusters are combined at earlier iterations of the algorithm. One can also deduce that smaller main clusters and outlier clusters are formed either at the time of large cluster formation or later.

Large clusters such as *Cluster 3* may contain many data points that do not resemble each other possibly due to their history of linkage in pre-stages of the algorithm never to be broken later. Also relatively large clusters like *Cluster 4* seem to have combined elements with slightly different shapes causing its noisy appearance.

5.4.11.4 UPGMA Linkage Clustering Evaluation

Evaluation of winter weekdays data clustering by UPGMA linkage is included in subsequent graphs.

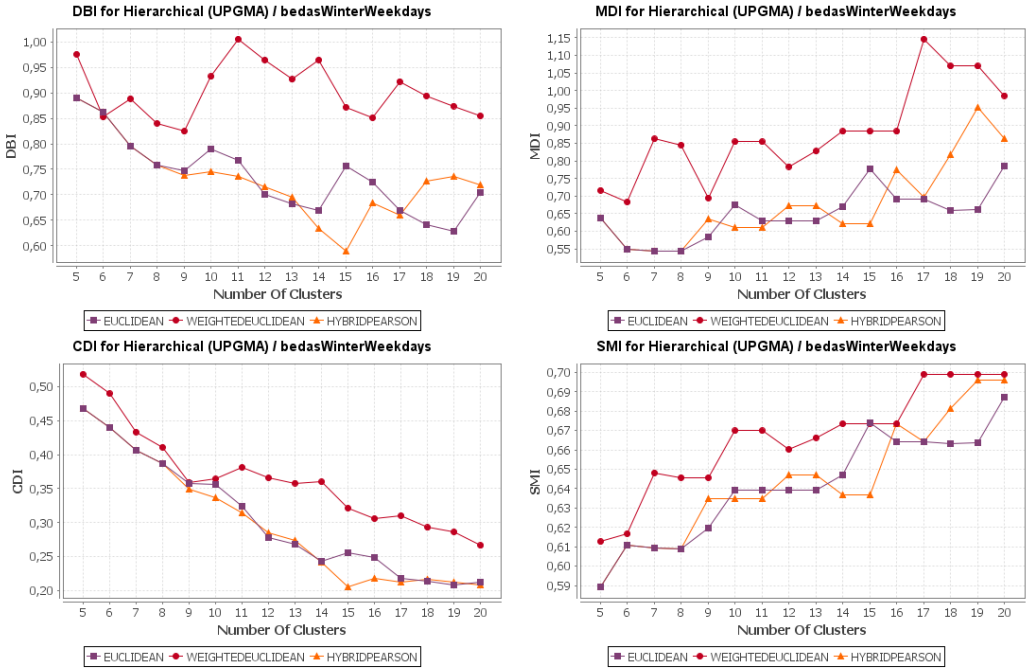

Figure 5.48 Evaluation of Hierarchical clustering with UPGMA linkage on winter weekdays data.

Nine clusters in Euclidean framework are selected for illustration in the following diagrams, because this value corresponds to elbow or knee points in DBI, MDI, CDI, and SMI charts in (Fig. 5.48).

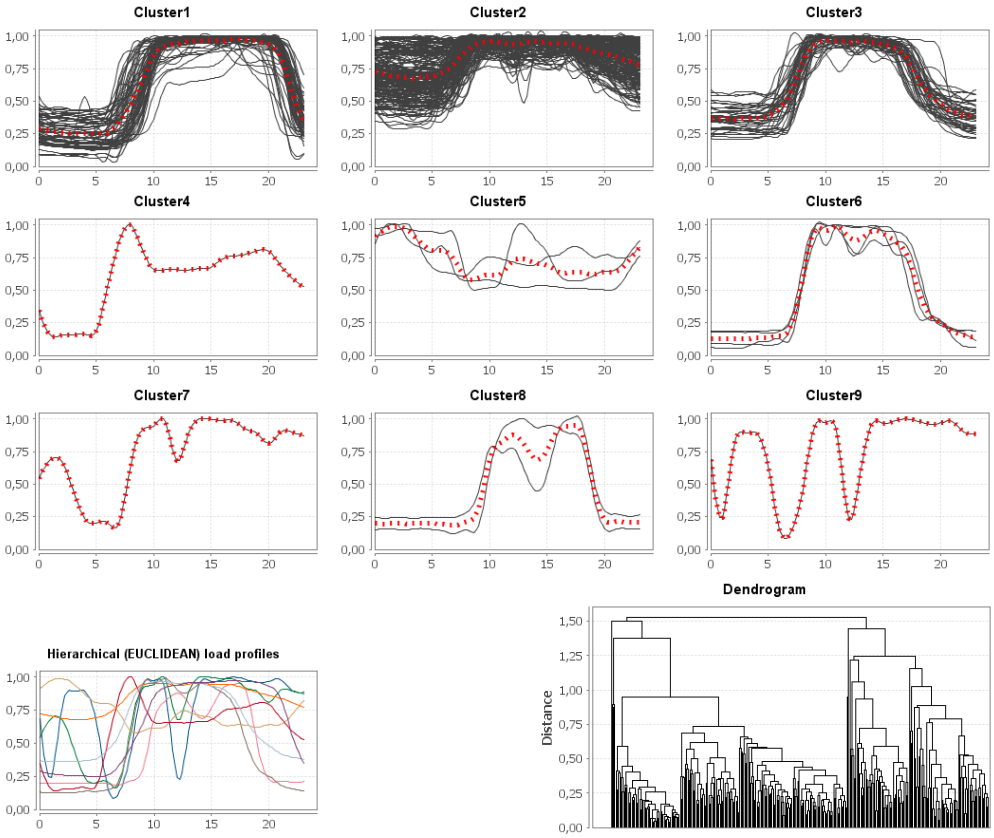

Figure 5.49 Clustering of winter weekdays data by Hierarchical clustering with UPGMA linkage in Euclidean framework.

Clustering with UPGMA linkage in (Fig. 5.49) tends to form less number of main dense clusters such as *Cluster 2*, which may hold distant points together. Outlier clusters generated by the algorithm are informative, behaving as barriers against noise in large clusters.

UPGMA behaves similarly as WPGMA, yet can be assessed as superior in terms of performance quality. UPGMA favors even larger clusters due to the fact that cluster sizes are weight factors in the computation of average distance by the algorithmic progress, thus attracting elements having smaller inter-distance to gather up, while repelling outliers from these relatively homogeneous large clusters. Main clusters contain less scattered elements compared to WPGMA.

Summer weekdays clustering by the algorithm with UPGMA linkage is evaluated as subsequently.

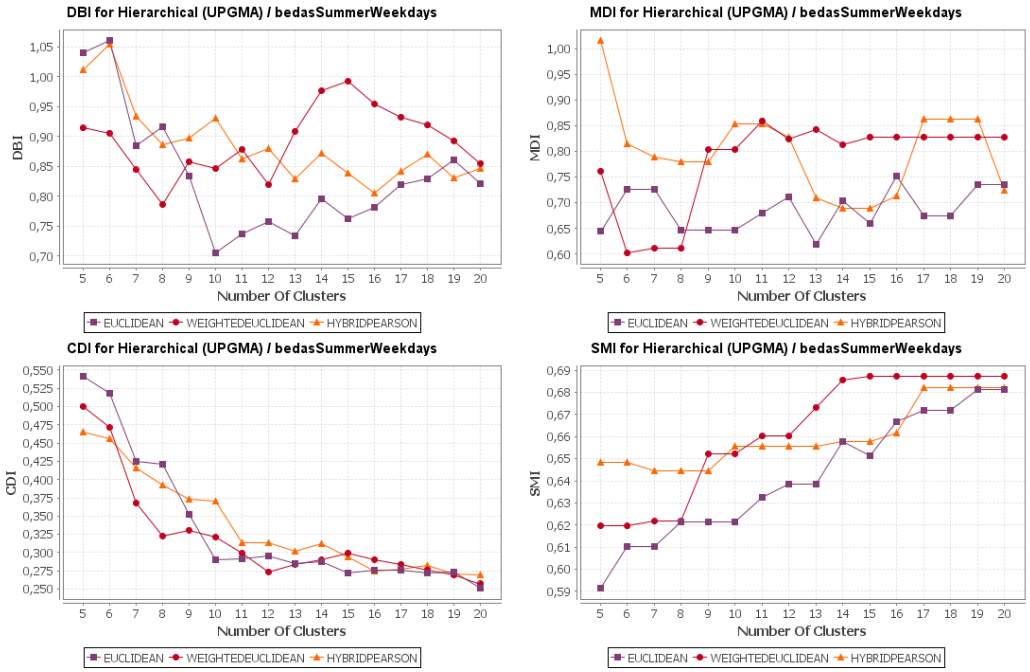

Figure 5.50 Evaluation of Hierarchical clustering with UPGMA linkage on summer weekdays data.

It is evident from DBI and CDI curves in (Fig. 5.50) that 10 clusters in Euclidean framework constitute turning points regarding the slope with low value readings of these metrics. MDI and SMI charts also include 10 clusters as one of their knee points. For grouping summer weekdays data, 10 clusters are selected to be illustrated in the following.

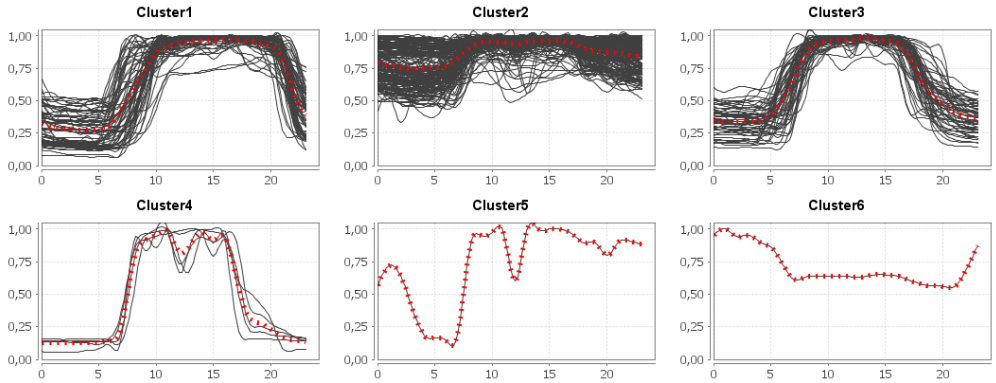

Figure 5.51 Clustering of summer weekdays data by Hierarchical clustering with UPGMA linkage in Euclidean framework.

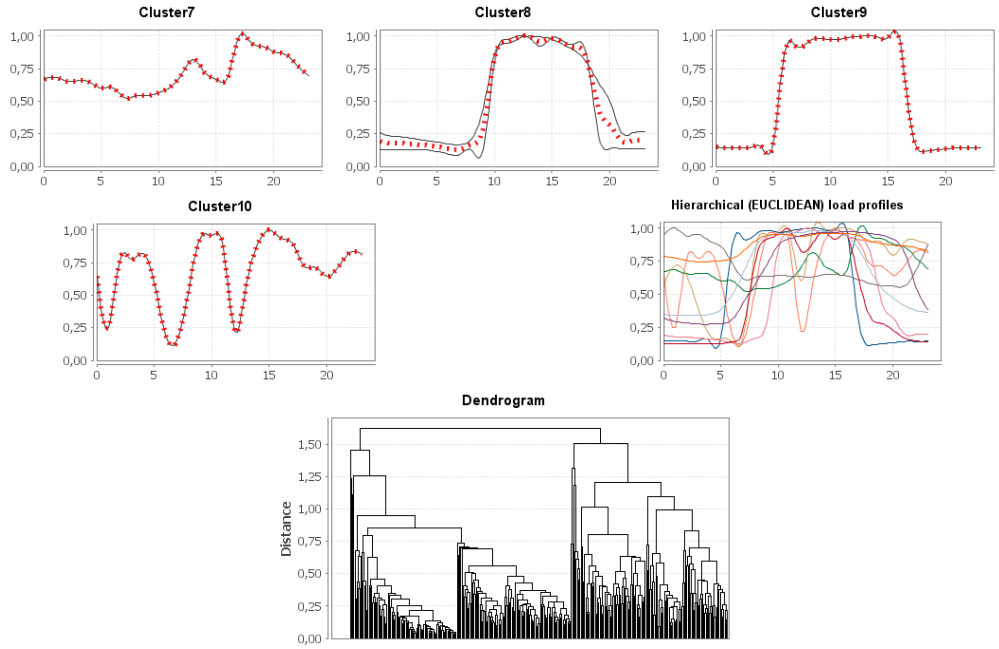

Figure 5.51 Clustering of summer weekdays data by Hierarchical clustering with UPGMA linkage in Euclidean framework. (cont'd)

UPGMA linkage forms abundant number of outlier clusters whose presence is considered to be informative, helping to detect distinct shapes of consumption illustrated in (Fig. 5.51). However, some large clusters such as *Cluster 2* include elements with high variance.

5.4.11.5 Ward's Linkage Clustering Evaluation

Winter weekdays data clustering evaluation by the algorithm adopting Ward's linkage is included in the following.

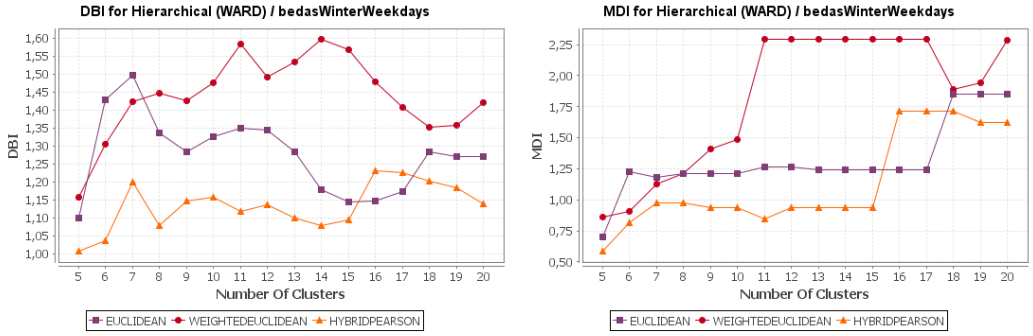

Figure 5.52 Evaluation of Hierarchical clustering with Ward's linkage on winter weekdays data.

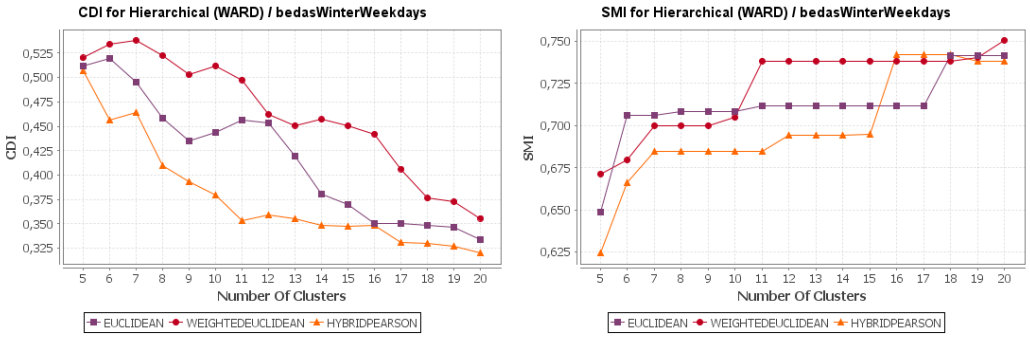

Figure 5.52 Evaluation of Hierarchical clustering with Ward's linkage on winter weekdays data. (cont'd)

Hybrid Pearson framework outperforms the remaining distance frameworks whenever Ward's linkage is employed. Curves of the graphs have local minima in 11 clusters in hybrid Pearson framework for DBI and MDI metrics. This value is also a knee point in CDI and SMI curves, as shown in (Fig. 5.52). Therefore, winter weekdays data is grouped into 11 clusters as follows.

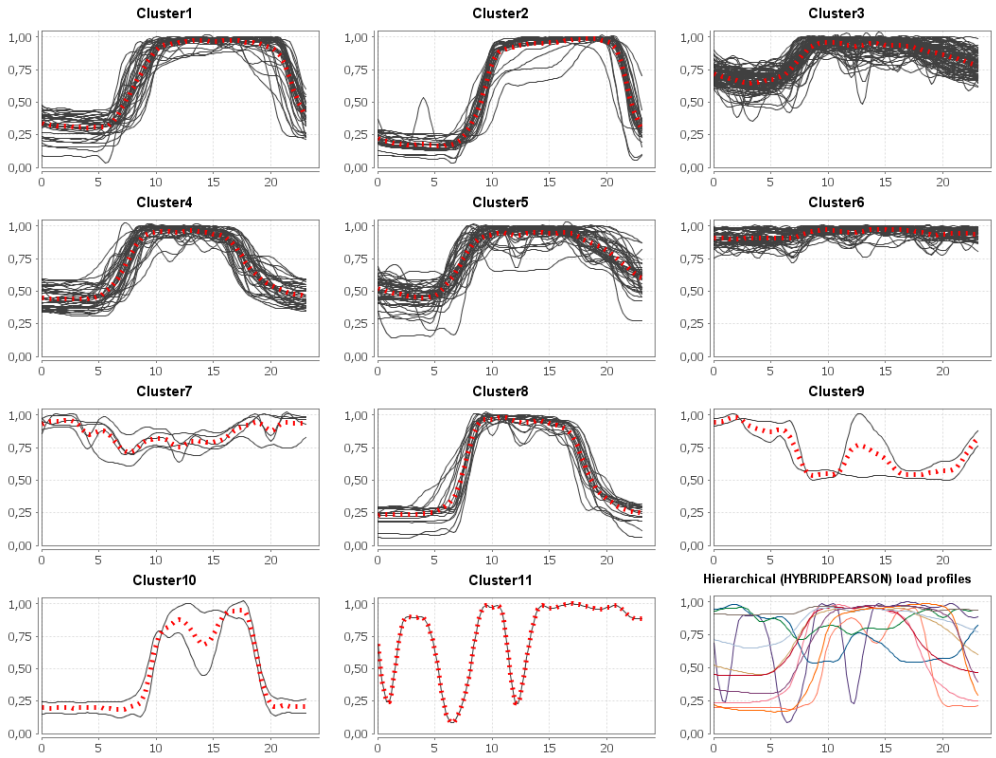

Figure 5.53 Clustering of winter weekdays data by Hierarchical clustering with Ward's linkage in hybrid Pearson framework.

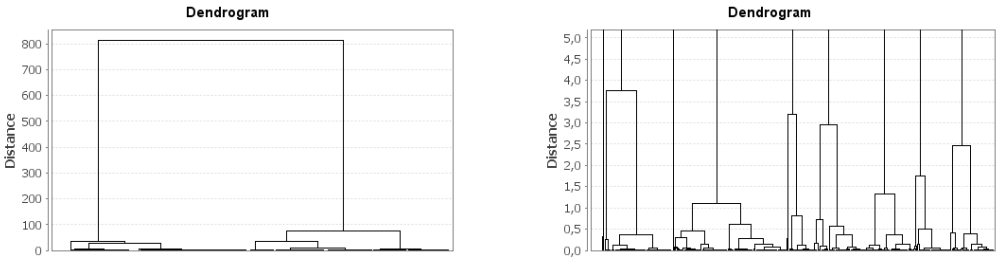

Figure 5.53 Clustering of winter weekdays data by Hierarchical clustering with Ward's linkage in hybrid Pearson framework. (cont'd)

Ward's linkage forms large clusters of similar sizes together with few outlier clusters, as shown in (Fig. 5.53). Due to lack of several distinct consumption shapes in the data set, some main clusters resemble each other, differing only in details. Certain noisy groupings are present in *Cluster 5* and *Cluster 2*. Within cluster scatter combined with general resemblance of large clusters accumulate error readings by validity indices. However, Ward's linkage presents suitable groupings that can be useful in applications whose goals may favor its clustering.

As distance levels increase exponentially in Ward's linkage, it is hard to visualize the dendrogram that shows clustering history. Hence, a close-up picture of the dendrogram is also included in (Fig. 5.53). This view verifies that Ward's linkage tends to form clusters of similar sizes.

5.4.11.6 Flexible Linkage Clustering Evaluation

Hierarchical clustering can be manipulated to yield clustering that fits with user demands through flexible linkage. In the rest of this section, power of flexible linkage in minimizing MDI value for all number of clusters will be analyzed. DBI minimization is not sought since minimum values of DBI metric correspond to a single linkage type of clustering that gathers most of the elements in a substantially large cluster while the rest is assigned to singleton clusters.

Flexible linkage to minimize MDI evaluation values for winter weekdays clustering is evaluated subsequently.

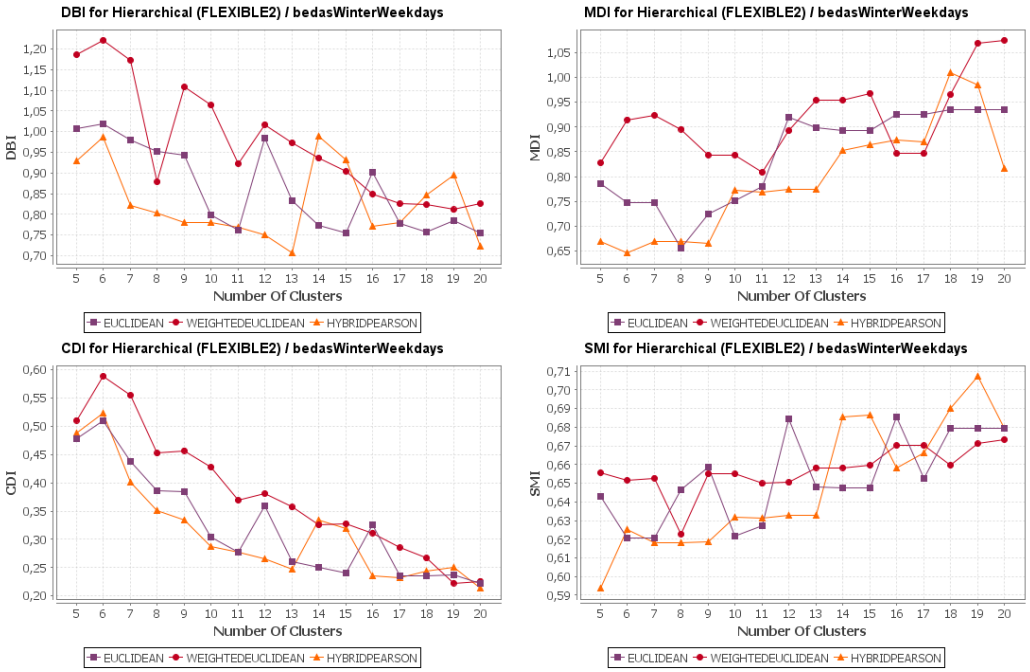

Figure 5.54 Evaluation of Hierarchical clustering with flexible linkage on winter weekdays data.

Nine clusters in hybrid Pearson framework are selected to represent grouping of winter weekdays data, because this value appears as local minima in DBI, MDI, SMI curves and constitutes a slope-changing point of CDI curve in (Fig. 5.54).

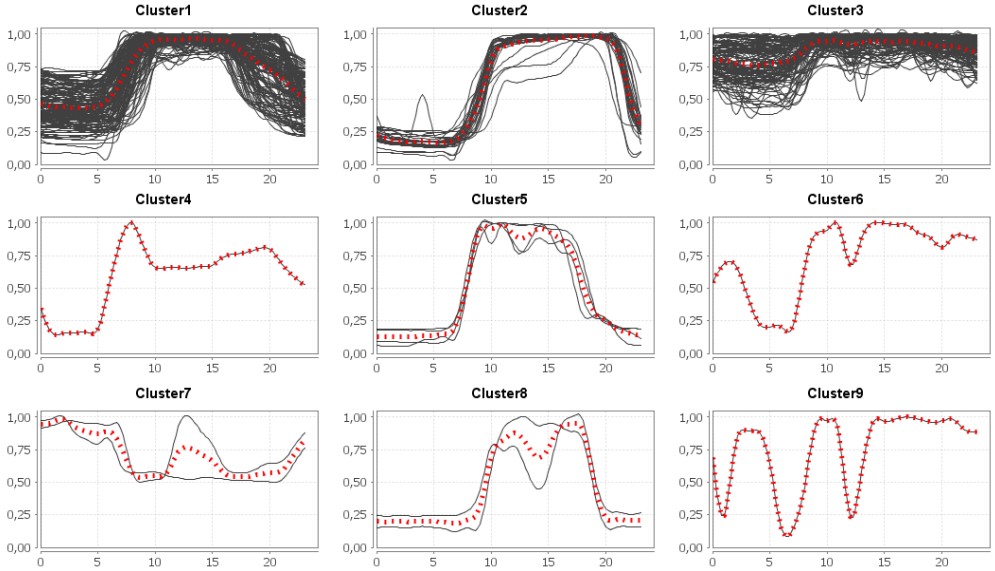

Figure 5.55 Clustering of winter weekdays data by Hierarchical clustering with flexible linkage in hybrid Pearson framework.

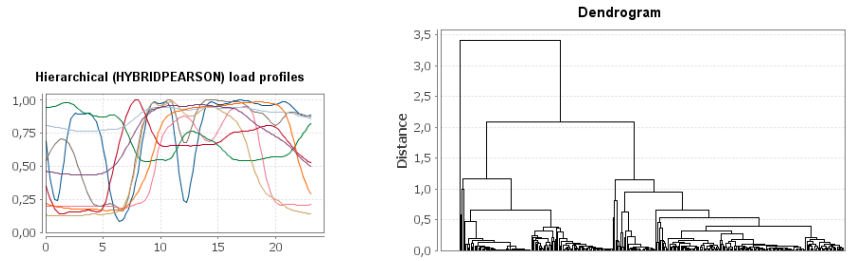

Figure 5.55 Clustering of winter weekdays data by Hierarchical clustering with flexible linkage in hybrid Pearson framework. (cont'd)

Flexible clustering yields main clusters of variable sizes, and carries out outlier detection and separation of informative quality as demonstrated in (Fig. 5.55). Although some of the main clusters include elements with higher variance, general quality of flexible linkage clustering turns out to be promising.

Summer weekdays data clustering by flexible linkage is evaluated as follows.

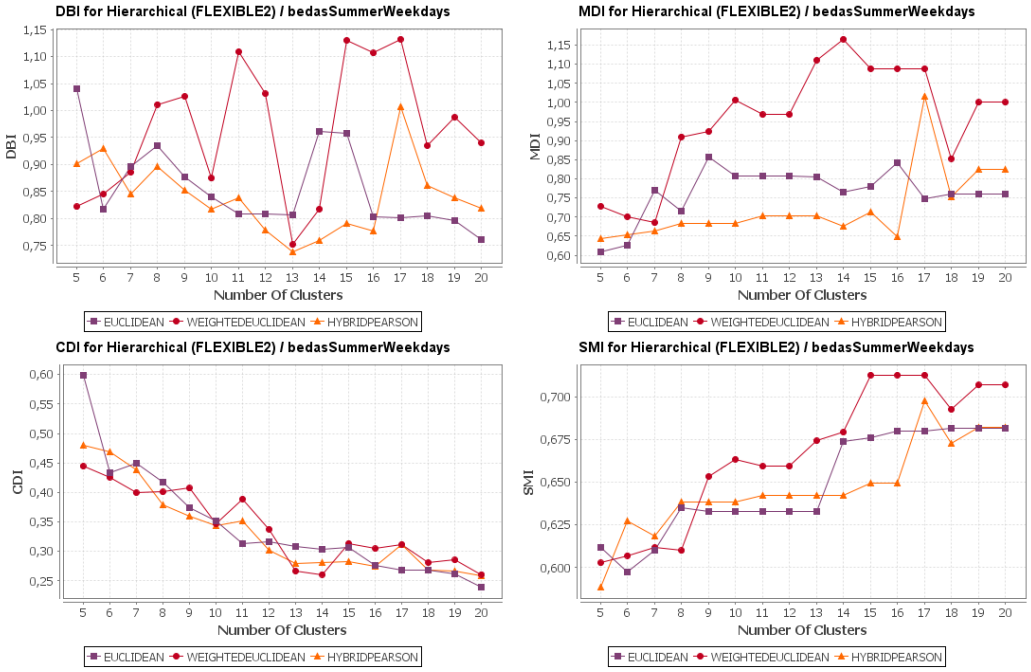

Figure 5.56 Evaluation of Hierarchical clustering with flexible linkage on summer weekdays data.

13 clusters in hybrid Pearson framework are selected to represent groupings in summer weekdays data as this value corresponds to the lowest DBI value, a slope-changing point of CDI curve, and low MDI and SMI values, as graphs in (Fig. 5.56) suggest. Corresponding clustering is included subsequently.

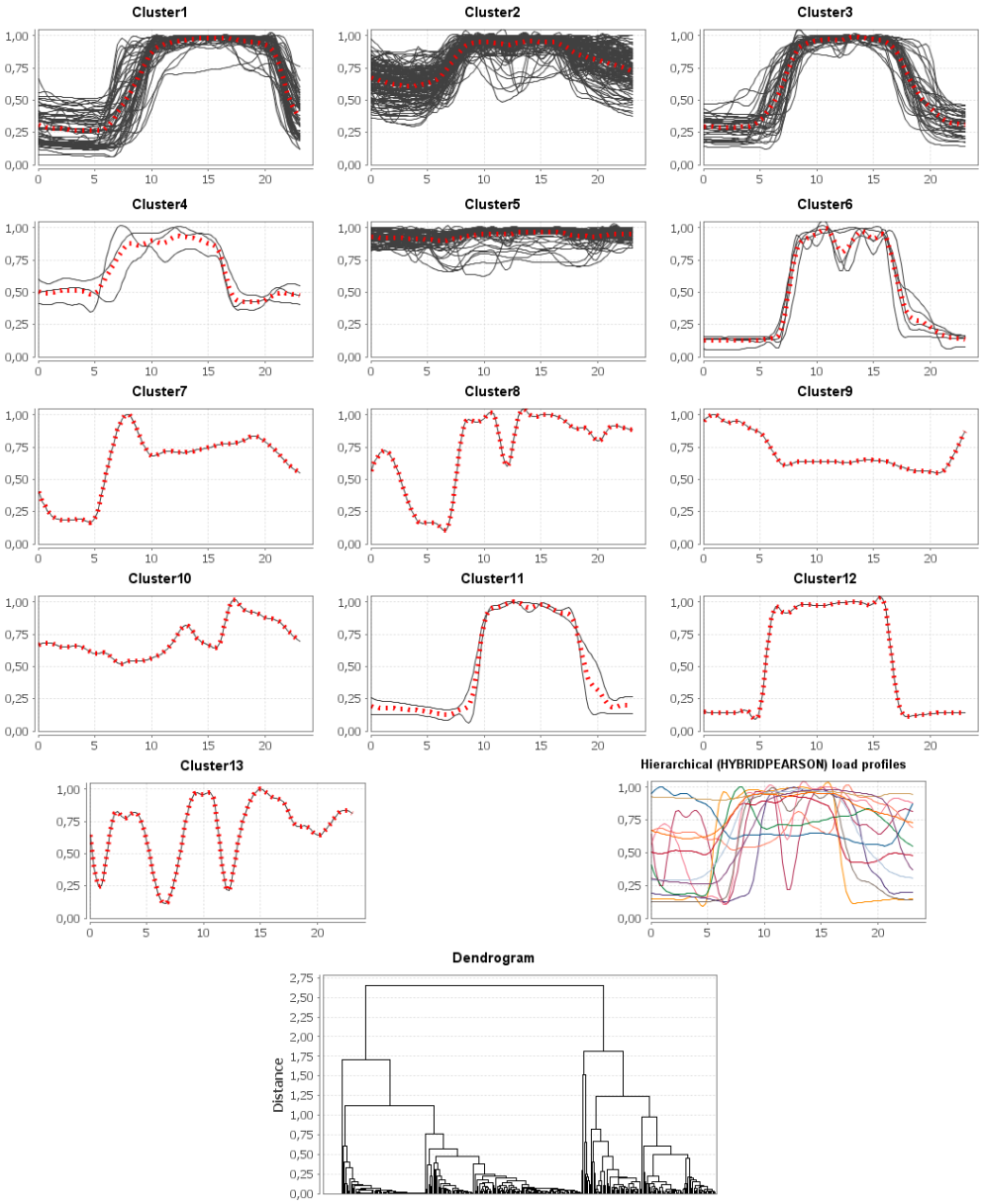

Figure 5.57 Clustering of summer weekdays data by Hierarchical clustering with flexible linkage in hybrid Pearson framework.

Looking at the dendrogram included in clustering results in (Fig. 5.57), one could observe characteristics of single linkage and complete linkage taking action in different regions of the graph. Flexible clustering produces large clusters some of which include moderate amounts of within-cluster scatter, and several outlier clusters. Note that in both winter and summer weekdays data clustering examples, value of β is set to 10^{-1}.

5.4.12. SOM Clustering Evaluation

SOM clustering evaluation is carried out using two distinct setups for batch and sequential training of the map. Results are presented for each of them in the following subsections.

5.4.12.1 Sequentially Trained SOM Clustering Evaluation

Evaluation of clustering winter weekdays data by sequentially trained SOM of size 6 by 15 is included as follows.

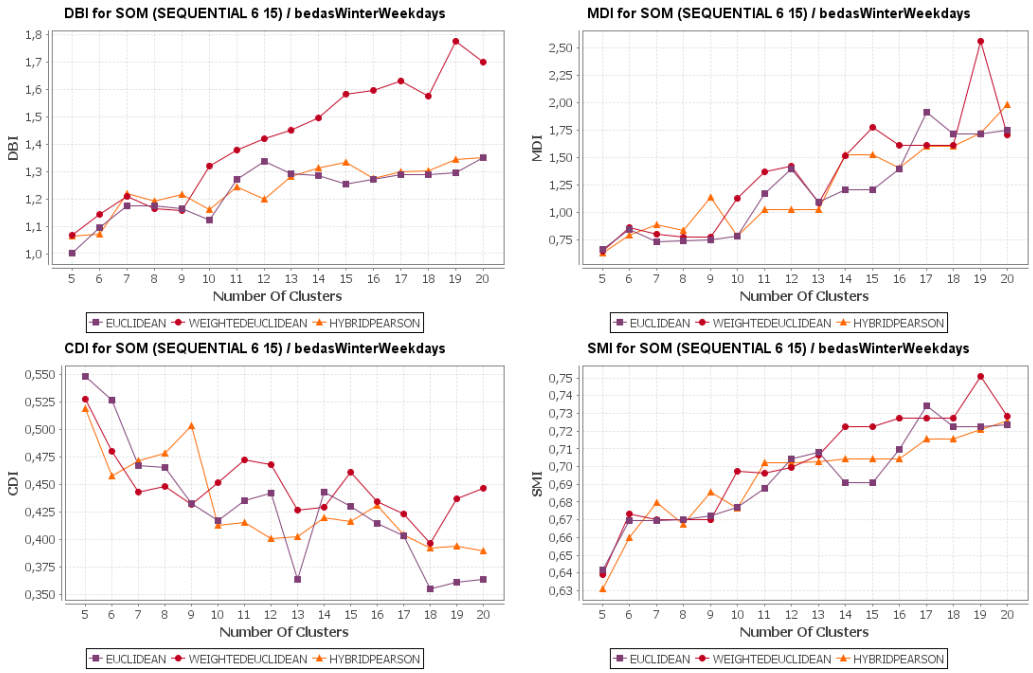

Figure 5.58 Evaluation of SOM (Seq. 6 15) clustering on winter weekdays data.

10 clusters to group winter weekdays data set are selected, since this value corresponds to local minima in DBI, MDI and CDI indices in (Fig. 5.58). Corresponding clustering is included in the following.

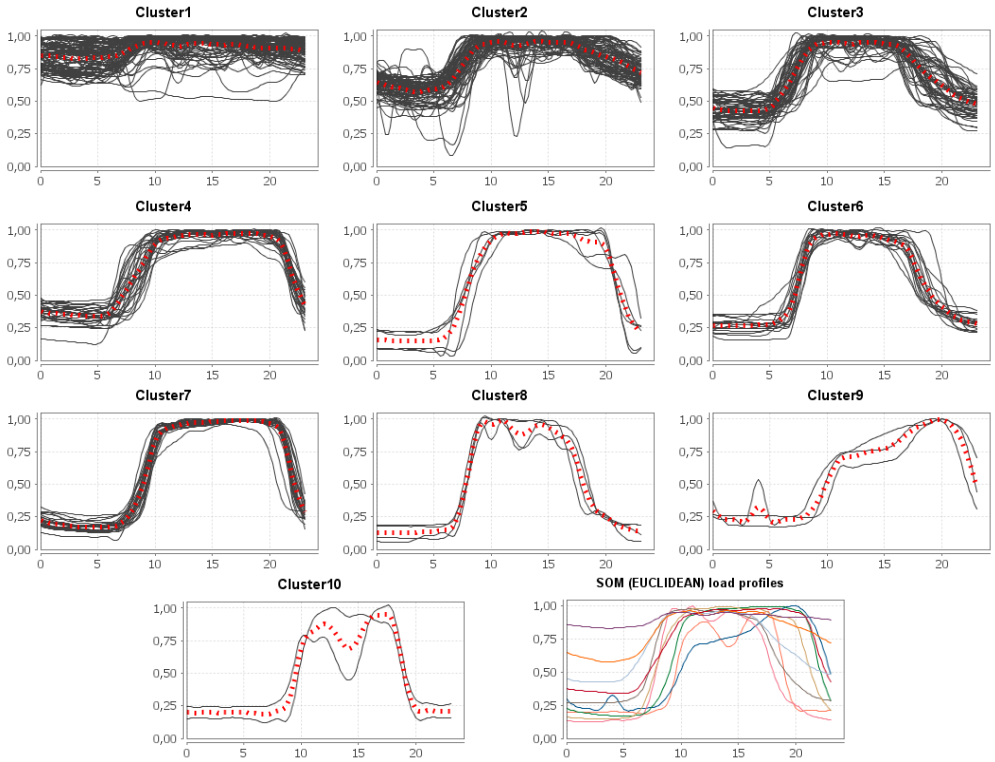

Figure 5.59 Clustering of winter weekdays by SOM (Seq. 6 15) in Euclidean framework.

SOM clustering illustrated in (Fig. 5.59) tends to form clusters that have relatively close centroids, as load profiles diagram illustrates. Main clusters come with varying number of elements. Some clusters may contain substantial amount of scatter such as in *Cluster 2*. Some outlier clusters such as *Cluster 9* and *Cluster 10* reveal distinct consumption styles.

The map obtained at the end of training, following the post-clustering operation is included subsequently. Nodes belonging to the same clusters are illustrated in same colors and names of the data points are marked to display node contents.

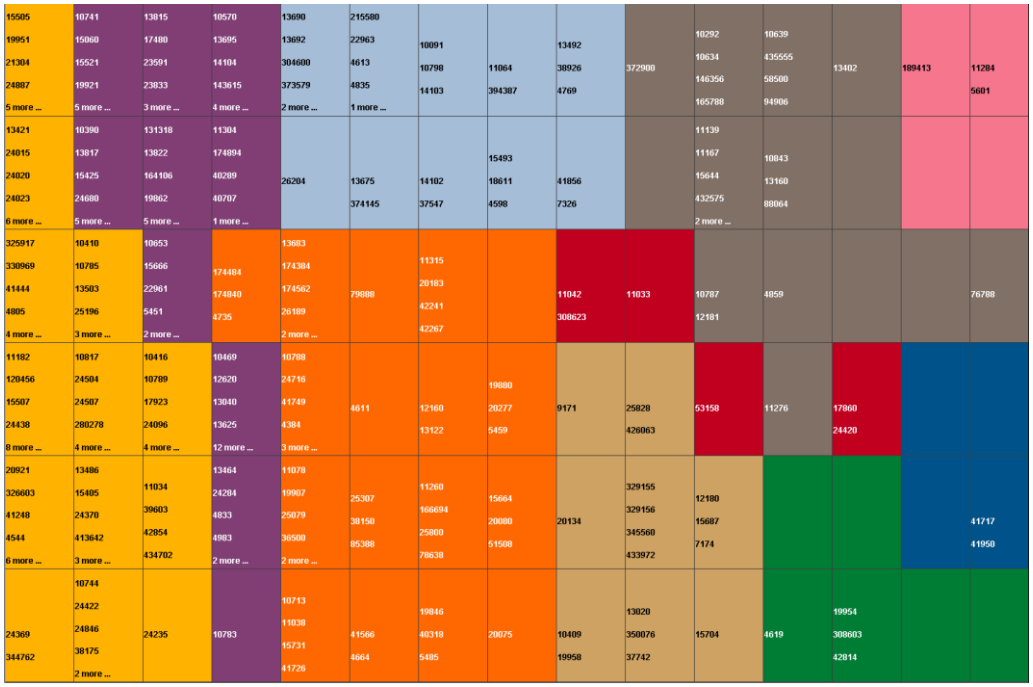

Figure 5.60 Final SOM (Seq 6 15) of winter weekdays data set grouped into 10 clusters in Euclidean framework, Map 1.

SOM provides means of visualization for the whole data set as in (Fig. 5.60). Most of the time, data points having same clusters lie in the same neighborhood in 2D space except *Cluster 5* which is illustrated by red on the map and has one of its members not adjacent to the rest of the data points pertaining to it.

5.4.12.2 Batch Trained SOM Clustering Evaluation

Evaluation of clustering of winter weekdays data set by batch-trained SOM of size 6 by 15 is included in the following.

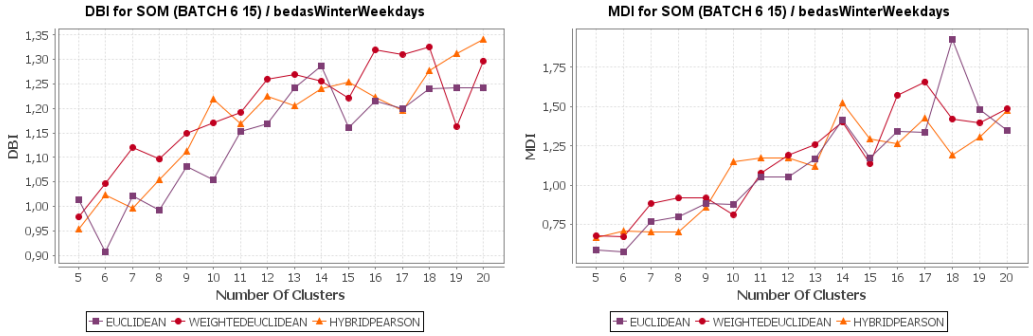

Figure 5.61 Evaluation of SOM (Batch 6 15) clustering on winter weekdays data.

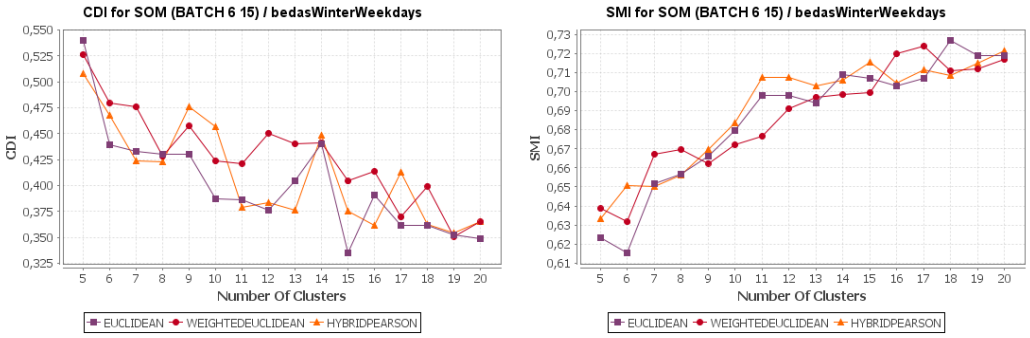

Figure 5.61 Evaluation of SOM (Batch 6 15) clustering on winter weekdays data. (cont'd)

Lowest DBI, MDI, and SMI values in (Fig. 5.61) indicate that performance of the algorithm at six clusters in Euclidean distance framework is superior to any other number of clusters. This value attains the greatest decline in CDI curve, as well. Therefore, six clusters are selected to group winter weekdays data, as illustrated below.

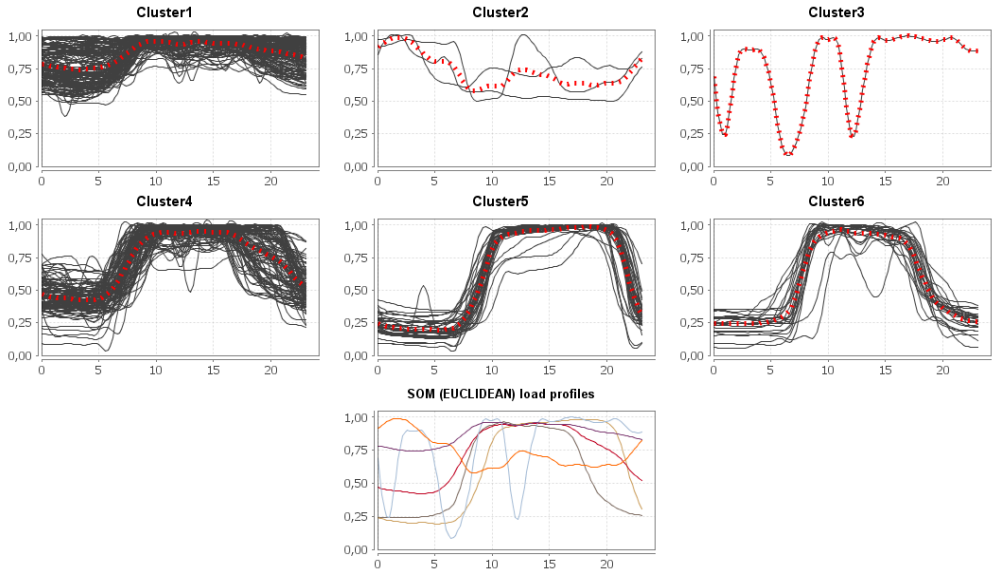

Figure 5.62 Clustering of winter weekdays data by SOM (Batch 6 15) in Euclidean framework.

Winter weekdays data clustering yields main clusters of varying sizes that contain scattered elements, and two outlier clusters. Main clusters are dense and clusters having less amounts of scatter such as *Cluster 1* in (Fig. 5.62) appear compact.

Trained and clustered map is included subsequently.

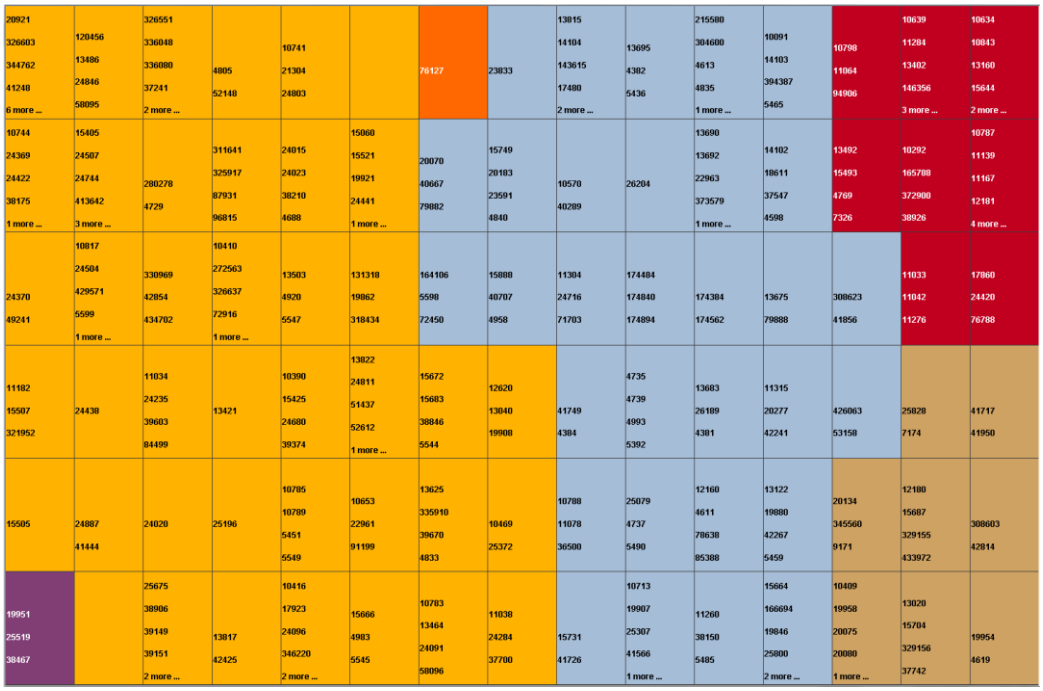

Figure 5.63 Final SOM (Batch 6 15) of winter weekdays data set grouped into 6 clusters in Euclidean framework, Map 2.

Note that all data points residing in *Cluster 2* have the same best matching unit in the map marked by purple color, as shown in (Fig. 5.63). This phenomenon explains the source of scatter present in main clusters: As distinct data points will eventually share same best matching units, hence the same locations in the map, a source for higher within-cluster variance naturally arises, because total number of neurons in a 6-by-15 map is less than one third of the data points in the winter weekdays set.

Quantization and topographic errors (Pölzlbauer, 2004) of the two maps in (Fig. 5.60) and (Fig. 5.63) are included in the following.

Table 5.1 Quantization and topographic errors comparison for the two maps.

	q_e	t_e
Map 1	0.2491	0.0817
Map 2	0.2231	0.0033

According to the results presented in Table 5.1, SOM under batch training has more prominent topology-preserving properties and generates less quantization error, meaning

that weight vectors in SOM lattice have closer values to actual data points, compared to SOM under sequential training.

Comparing first and second maps reveals that the second one preserves the topology of the multidimensional data set better than the latter, which is an expected result since the first map is reported to violate adjacency of data points within a cluster.

5.5 Discussion on Evaluation of Clustering Algorithms

In this section, algorithms are evaluated only on winter weekdays data set in Euclidean and hybrid Pearson frameworks, in order to demonstrate their performance with increasing number of clusters.

Subsequent subsections cover the evaluation of K-means family, Hierarchical clustering family and a comparison of all clustering algorithms, in which Hierarchical clustering and K-means families are represented by only one member algorithm that is analyzed to have acquired better performance quality in terms of validity indices.

5.5.1 Comparison of K-Means Clustering Family

K-means and algorithms with similar mechanisms that group data items into hard, i.e. non-fuzzy clusters are evaluated in this section in order to analyze their performances. Included algorithms are shown in these colors throughout this section: K-means in purple, K-means++ in red, K-medians in blue, WFA-K-means in orange, K-medoids (PAM) in brown, Hopfield-K-means in grey, and Similarity-Based K-means in green.

Evaluation of K-means family on winter weekdays data set in Euclidean distance framework is included in the following.

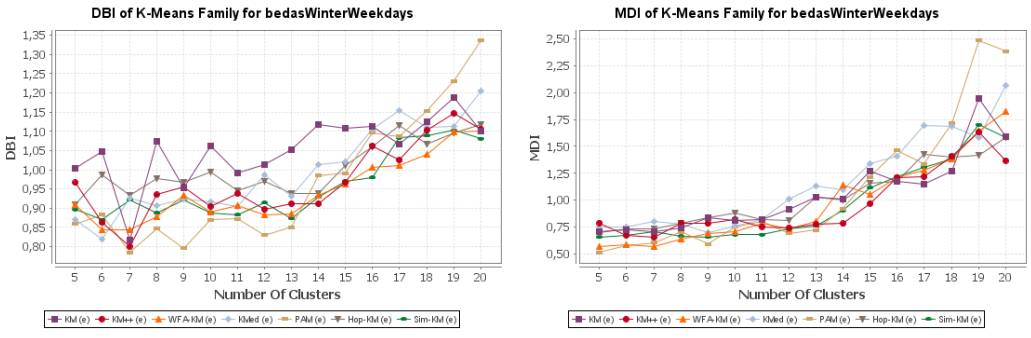

Figure 5.64 Evaluation of K-means family algorithms on winter weekdays data in Euclidean distance framework.

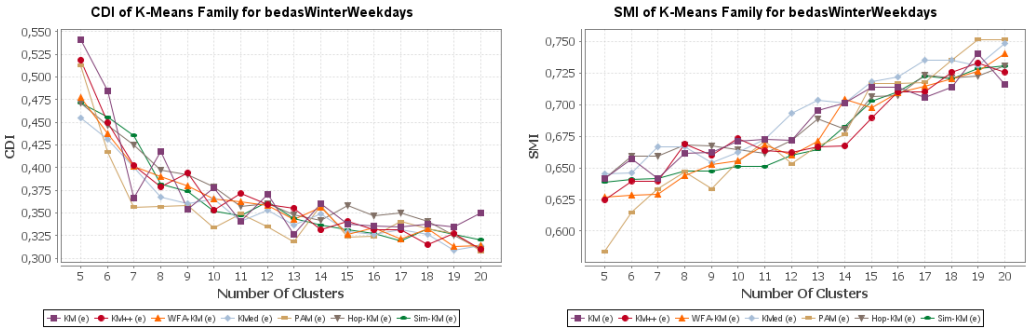

Figure 5.64 Evaluation of K-means family algorithms on winter weekdays data in Euclidean distance framework. (cont'd)

K-means and similar algorithms compete with each other in terms of performance quality. However, DBI, MDI, and SMI metrics evaluation in (Fig. 5.64) cover a wider range with respect to other clustering algorithm families. Especially according to MDI metric performance degrades considerably as number of clusters increase. DBI and SMI metric concur with this observation, though not at the same scale. CDI metric also suggests that K-means clustering algorithms perform significantly better for lower number of clusters as CDI curve exhibits greater declines in value for up to nine clusters. Afterwards, the curve follows rather a linear decrease with occasional oscillations to yield more or less the same evaluation results for higher number of clusters.

For lower number of clusters, K-medoids, WFA-K-means and similarity-based K-means algorithms tend to lead other family members. However, for higher number of clusters performance of K-medoids rapidly decays, possibly due to the contribution of medoid selection operation to total error values, since medoids correspond to actual data points, not to some hypothetical centers computed by going over all data points pertaining to clusters.

Zigzags in evaluation curves of K-means and K-means++ algorithms are notable. On the contrary, WFA-K-means and Similarity-Based K-means methods follow relatively smooth curves. Hopfield-K-means also exhibits fewer oscillations and its evaluation curves are rather continuous, yet it is evaluated to perform slightly worse than most of the family members. However, neither WFA-K-means nor Similarity-Based K-means can be said to outperform other member algorithms, as curves of evaluation metrics demonstrate several trade-offs. In other words, situational analysis is obligatory to arrive at a decision concerning which family member to use for some desired number of clusters.

Partitioning winter weekdays data into clusters by K-means family algorithms in hybrid Pearson distance framework is evaluated subsequently.

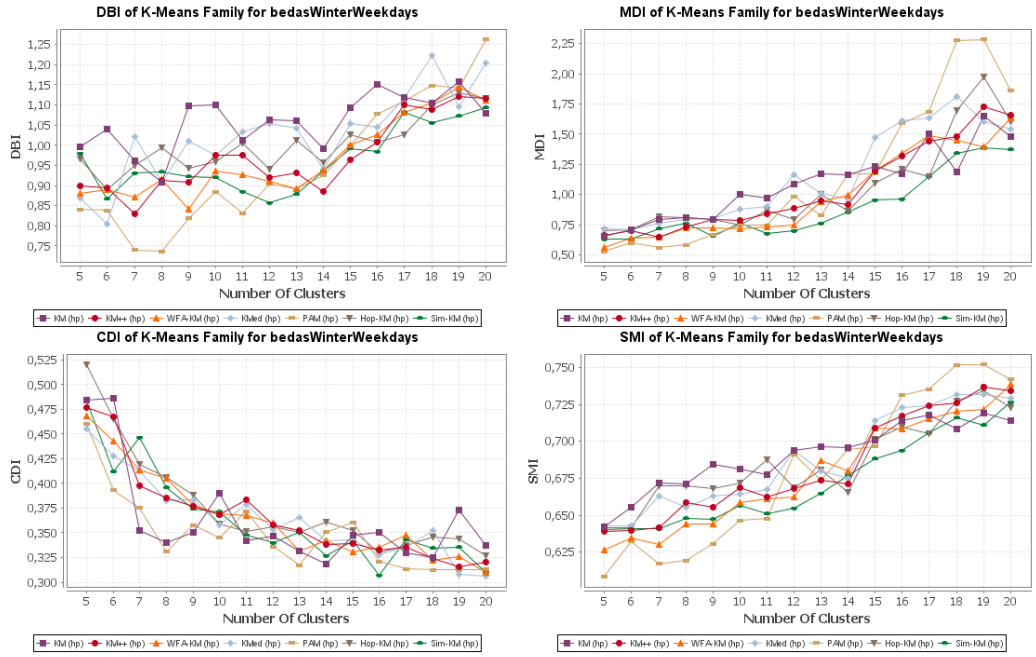

Figure 5.65 Evaluation of K-means family algorithms on winter weekdays data in hybrid Pearson distance framework.

In hybrid Pearson framework, K-medoids clustering is evaluated to have the best performance for lower number of clusters, as the results in (Fig 5.65) shows. However, the same algorithm attains the worst performance among family members as number of clusters increase. It is quite hard to make a strict ordering between algorithms based on evaluation metric readings. Nonetheless, robust performances of K-medoids, WFA-K-means, and similarity-based K-means turn out to be stimulating.

For both distance frameworks, performance of WFA-K-means is considered to be the most promising, because the algorithm attains relatively low validity index values and less amounts of change in these values exist between successive numbers of clusters in curves of evaluation metrics. Sudden great deteriorations of performance in adjacent cluster numbers such as in case of classical K-means clustering, or tendency of achieving only isolated discontinuous better performance results, as observed in case of K-medoids clustering do not occur in WFA-K-means clustering. Therefore, WFA-K-means algorithm is selected to represent K-means family for further comparisons among all algorithms.

5.5.2 Comparisons of Hierarchical Clustering Family

Performance of Hierarchical clustering algorithms on winter weekdays data set in Euclidean framework is evaluated in the following.

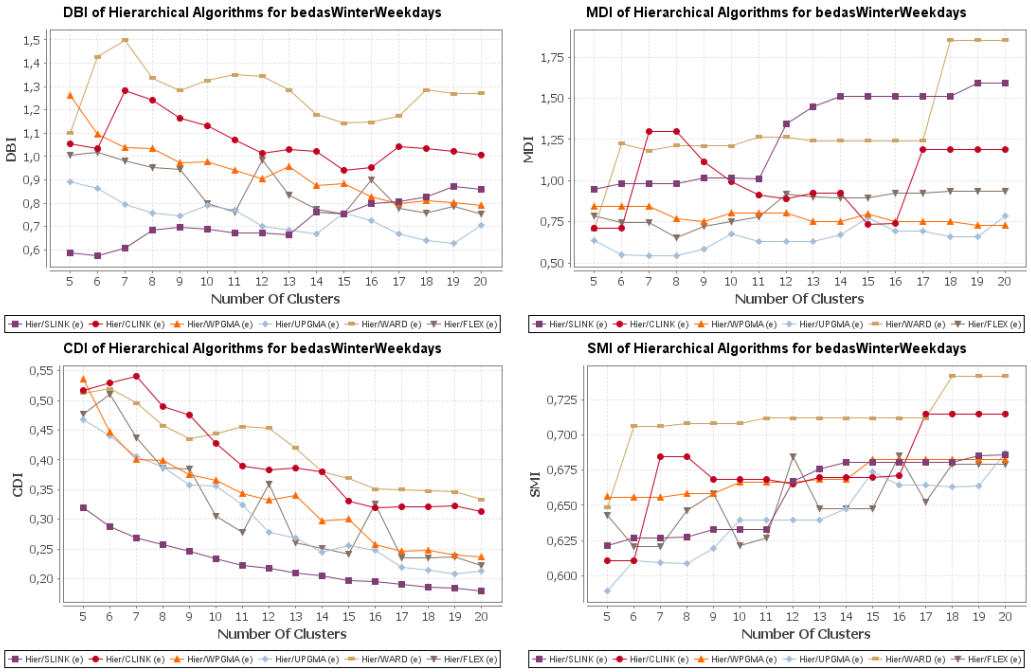

Figure 5.66 Evaluation of Hierarchical clustering with distinct linkage criteria clustering winter weekdays data in Euclidean distance framework.

Concerning evaluation results included in (Fig. 5.66) and in subsequent curves included in this section, purple curve represents single linkage, while the blue represents UPGMA, the gray represents flexible linkage, the orange represents WPGMA, the brown represents Ward's linkage, and finally the red represents the complete linkage, as included in the legend.

DBI evaluation depicted in (Fig. 5.66) indicates that for 5 to 14 clusters, the most adequate performance is met by Hierarchical clustering with single linkage. However, DBI is biased towards singleton clusters, and single linkage is analyzed to yield only one over-populated cluster accompanied by several singleton clusters within the range of 5 to 20 clusters. CDI metric is also overwhelmed by this phenomenon, so single linkage clearly outperforms all other linkage criteria according to this index as well.

MDI and SMI metrics do not exhibit as biased behaviors praising singleton clusters as DBI and CDI. Graphs included in (Fig. 5.66) indicate that UPGMA linkage criteria turns out to be more promising than the latter by attaining lower evaluation values in almost each case. Leaving aside single linkage, DBI and CDI curves also concur with this observation.

If one has to make only one choice of linkage criteria only by looking at the graphs of validity indices before running Hierarchical clustering on winter weekdays data in Euclidean distance framework, the highest quality of performance is most likely to be obtained by selecting UPGMA linkage. However, as all evaluation curves suggest, there

are many trade-offs concerning the adopted linkage criterion. Instead of making blindfold choices, a data analyst should deeply analyze and visually inspect clustering results; as done in previous sections of this chapter.

Winter weekdays data clustering by Hierarchical algorithms in hybrid Pearson framework is evaluated as follows.

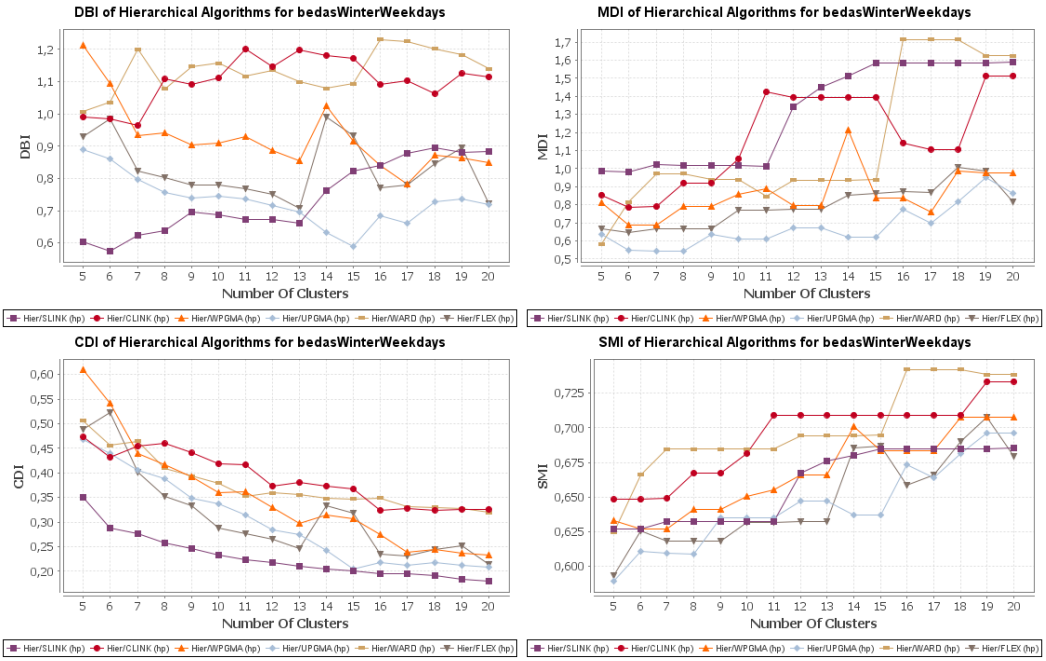

Figure 5.67 Evaluation of Hierarchical clustering with distinct linkage criteria clustering winter weekdays data in hybrid Pearson distance framework.

Evaluation of the same data set in hybrid Pearson framework in (Fig. 5.67) verifies that performance of UPGMA linkage is favorable. Under hybrid Pearson distance, the gap between single linkage and UPGMA increases at each side of the crossover in DBI curve, covering a wider interval than Euclidean framework.

Flexible linkage follows less regular curves in metric graphs, since for each number of clusters it may set algorithmic parameters utilized in distance computation differently. However, in certain cases flexible linkage outperforms UPGMA, as CDI and SMI metrics evaluation in (Fig. 5.67) put forward. Therefore, flexible clustering shall be considered as an alternative to UPGMA in particular cases, though the data analyst should be aware that flexible linkage has no intrinsic characteristics or definite predefined behavior.

Although cases have been shown in which UPGMA is overthrown by other linkage criteria, it is selected to represent Hierarchical clustering in future comparisons with other clustering algorithms. The reason for this choice is that UPGMA exhibits a consistent behavior and even in cases validity indices indicate that it may not be the most promising

criterion, difference in metric value between UPGMA and the winning criterion never gets too significant.

5.5.3 Comparisons of All Clustering Algorithm Families

Performance comparisons among main clustering techniques; WFA-K-means representing K-means family, Hierarchical Clustering with UPGMA linkage representing Hierarchical clustering family, Follow-the-Leader clustering, Fuzzy K-means, and SOM clustering; are likely to reveal the grounds for adopting a particular algorithm. ISODATA clustering is not included in this section due to deficiencies of the evaluation methodology in assessing the performance of this algorithm, as discussed previously.

Evaluation of algorithms is carried out on both winter and summer weekdays data sets in Euclidean distance framework. Winter weekdays data clustering by all algorithm families in Euclidean distance framework is evaluated as follows.

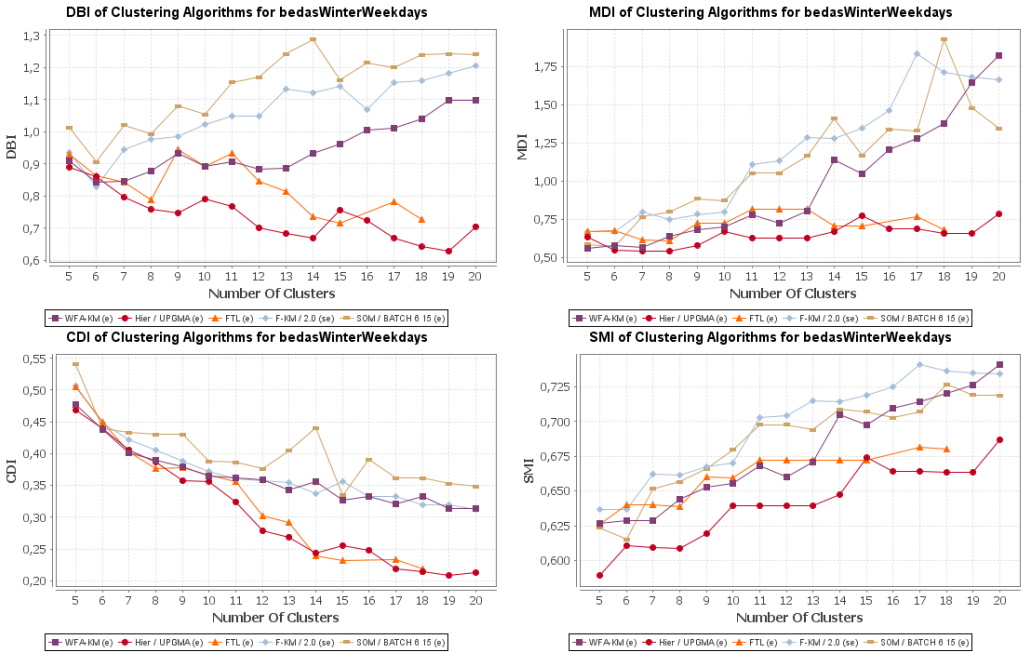

Figure 5.68 Evaluation of all families of clustering algorithms on winter weekdays data in Euclidean distance framework.

Validity index curves included in (Fig. 5.68) and in other figures covered in this section display WFA-K-means in purple, Hierarchical clustering with UPGMA linkage in red, Follow-the-Leader clustering in orange, Fuzzy K-means in blue and SOM clustering with batch training in brown.

Evaluation of algorithms summarized in (Fig. 5.68) indicates that Hierarchical clustering with UPGMA linkage and Follow-the-Leader algorithm turn out to be the most promising

clustering techniques that exhibit similar behaviors. On the other hand, WFA-K-means, Fuzzy K-means and SOM clustering methods follow the same pattern in metric curves, among which WFA-K-means clearly outperforms the latter.

Summer weekdays data clustering by algorithms is evaluated in the following charts.

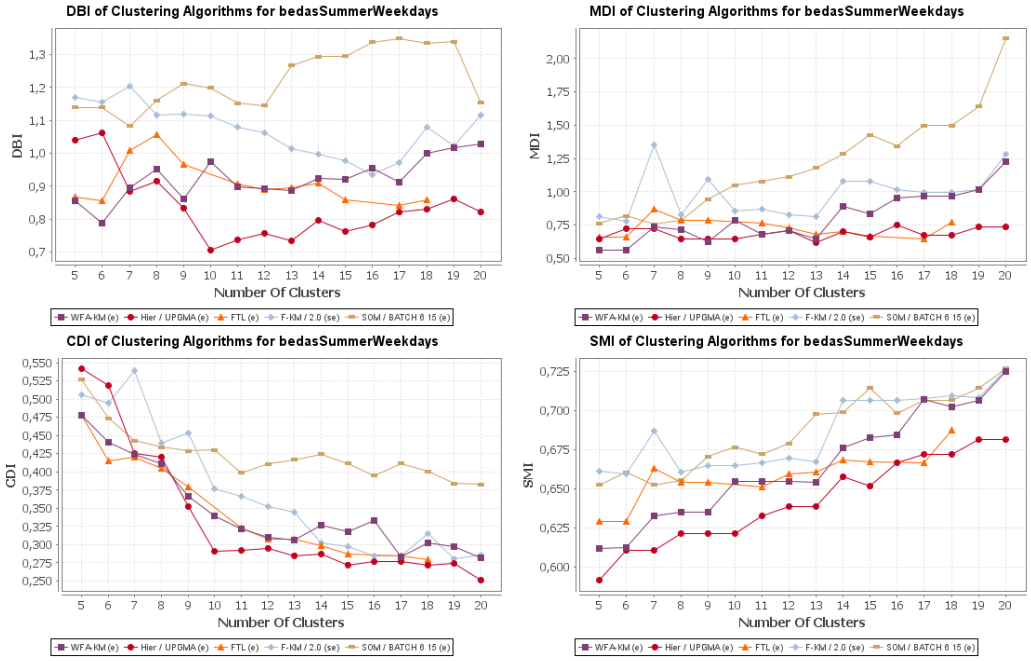

Figure 5.69 Evaluation of all families of clustering algorithms on summer weekdays data in Euclidean distance framework.

In case of summer weekdays data, Hierarchical clustering with UPGMA linkage outperforms the rest of the algorithms, while WFA-K-means surpasses Follow-the-Leader clustering for most number of clusters shown in (Fig. 5.69). This observation suggests that characteristics of a data set constitutes a crucial criterion in deciding a particular algorithm to use and all candidate methods must be evaluated on all available data sets prior to decision. However, both evaluation results indicate certain tendencies of performance qualities of clustering algorithms such as relatively poor performance of SOM clustering and outstanding performance of UPGMA linkage as verified by evaluation of clustering of both data sets.

5.6 Summary

Evaluation metrics do not necessarily explain every aspect of performance quality of algorithms in comparing clusterings generated by two distinct methods. Throughout the evaluation chapter, clusters that are evaluated to attain either the lowest or relatively low values of validity indices are visualized for all algorithms so that data analyst who can

arrive at a final concrete decision on a particular algorithm to use can visually inspect the generated clusters.

Furthermore, discovering natural clusters in a data set is most of the time an intermediate step of a large project aiming at developing an application. In such cases, subjective goals that are not necessarily captured by evaluation metrics come into question to complicate the problem fundamentally. Therefore, proposing a variety of clustering algorithms that are demonstrated to effectively cluster a data set associated to a particular problem domain and modifying mechanisms of algorithms in search for increased quality of performance will be wiser than trying to prove the superiority of a certain algorithm, which can easily be falsified by a hypothetical data set.

Running time complexities and scalability of algorithms to increasing amounts of data sizes are analyzed in the proceeding chapter to elaborate the evaluation of clustering algorithms.

CHAPTER 6

RUNNING-TIME EVALUATION OF CLUSTERING ALGORITHMS

6.1 Introduction

Running time complexities of clustering algorithms come to be crucial issues when large data sets are involved in the problem. In this chapter, performances of all algorithms are evaluated based on the time they spend to generate clusters on winter and summer weekdays data sets.

Running times of algorithms are also measured on synthetic data sets of increasing sizes generated by using models based on winter and summer weekdays data sets so that *scalability* of algorithms to larger data is revealed. Fundamental algorithmic solutions to decrease running times and to improve scalability whenever possible are also discussed in course of the chapter.

Running time evaluation is conducted on a computer with Intel® Core™ i7-3829QM CPU with 2.70 GHz. Maximum number of processing units is 8, attained by Turbo Boost Technology.

6.2 Synthetic Data Generation

Members of prominent clusters as well as outliers produced by algorithms can be utilized as models to generate synthetic data points that are similar in shape to their models, yet differ in values and in the amount of noise present.

11 distinct model sets composed of particular data points belonging to major or outlier clusters are formed. Number of cases generated using main and outlier models is kept proportional to actual number of data points assigned to these two types of clusters.

Multivariate normal random distribution is used to synthesize data points based on models. MATLAB *mvnrnd* function is adopted for synthetic data generation in the thesis (Theodoridis et al., 2010). Multivariate normal random numbers function of MATLAB operates on each of the models whose feature-based mean values and feature-by-feature covariance matrix together with the desired number of instances of synthetic data are used as parameters of the *mvnrnd* function.

Data points are normalized in the process, as done in pre-clustering stages. Instances having some of their features with negative values may be generated by *mvnrnd* function. These instances are excluded from synthesized data sets.

Three data sets with increasing number of instances are formed such that the first data set, *Synthetic Data 1*, consists of 883 data points, the second data set, *Synthetic Data 2*, consists of 1613 data points, and the third data set, *Synthetic Data 3*, has 3222 data points. Note that sizes of sets grow twice as much while traversing from the first to the second and from the second to the third data set.

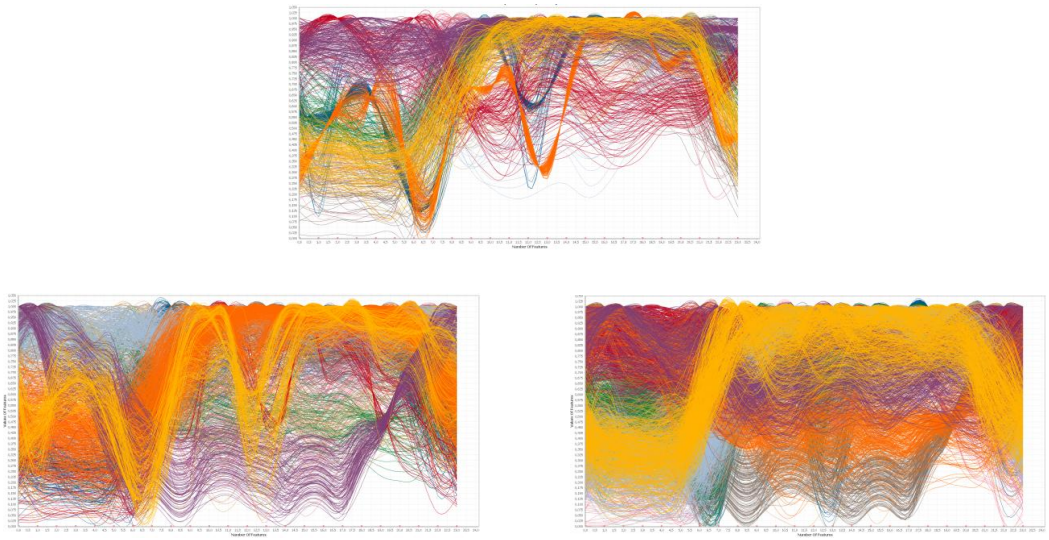

Figure 6.1 All points of the first, second, and the third synthetic data sets partitioned into 10 clusters.

Instances of synthetic data sets, as partitioned into 10 clusters by WFA K-means algorithm, are illustrated in (Fig. 6.1), in which horizontal axis stands for hours and vertical axis stands for normalized consumption values. Data points belonging to the same cluster are shown in the same color. The first synthetic data set is displayed in the upper middle, the second is in the lower left, and the third is in the lower right positions of the illustration in (Fig. 6.1).

Although every cluster is not visible due to overlapping features of data points, data set sizes can be compared by observing the illustration.

6.3 Running Times of Algorithms on Real Consumption Data

Running times and number of iterations to generate 10 clusters by all algorithms in Euclidean, hybrid Pearson, and weighted Euclidean frameworks are summarized in the following table.

Table 6.1 Running times and number of iterations of all algorithms running on winter and summer weekdays data sets to generate 10 clusters.

		Euclidean		Hybrid Pearson		Weighted Euclidean	
		Run-time (ms)	Iteration #	Run-time (ms)	Iteration #	Run-time (ms)	Iteration #
K-means	Winter	47	12	94	27	44	17
	Summer	36	15	64	16	36	15
K-means++	Winter	52	19	66	15	44	6
	Summer	43	16	63	6	46	7
K-medians	Winter	73	12	54	11	37	7
	Summer	48	9	89	25	32	6
WFA-K-means	Winter	43	9	41	4	61	14
	Summer	89	15	64	10	80	13
Hopfield-K-means	Winter	$29+T_t$	17	$40+T_t$	13	$26+T_t$	12
	Summer	$28+T_t$	15	$21+T_t$	5	$20+T_t$	10
K-medoids	Winter	53	2	97	3	59	2
	Summer	58	2	184	3	64	2
Similarity K-means	Winter	7290	39	2978	22	2353	14
	Summer	1918	13	1762	11	3855	27
ISODATA	Winter	248	21	293	15	160	18
	Summer	147	12	243	10	146	12
Fuzzy K-means	Winter	347	8	885	12	3081	84
	Summer	552	13	2058	29	384	7
Follow-the-Leader	Winter	57	7	79	6	110	14
	Summer	31	4	49	4	53	4
Hierarchical (UPGMA)	Winter	157	295	176	295	151	295
	Summer	123	307	136	307	137	307
SOM (Seq. 6 15)	Winter	6098	(90, 80)	10892	(90, 80)	27677	(90, 80)
	Summer	6621	(90, 80)	11357	(90, 80)	27804	(90, 80)
SOM (Batch 6 15)	Winter	7257	(90, 80)	11997	(90, 80)	29756	(90, 80)
	Summer	7630	(90, 80)	14334	(90, 80)	30924	(90, 80)

A quick look in Table 6.1 to compare running times of algorithms under distinct distance frameworks elicits that running time complexity associated with hybrid Pearson measure is higher than other metrics, while similar results are yielded regarding Euclidean and weighted Euclidean metrics. However, number of iterations until convergence differs among distance metrics and algorithms. Since majority of the algorithms included in Table 6.1 contains mechanisms working in a random fashion and the evaluation methodology aims to present the clustering with the lowest metric values, a direct comparison of running time complexities regarding distance metrics by taking the majority vote of all algorithms may be error prone.

In order to understand better running time complexities related to adopting a particular distance framework, comparisons can be made among algorithms having deterministic nature. A candidate for such an algorithm is Follow-the-Leader clustering, whose evaluation indicates that Euclidean distance metric requires less computation time than the latter, and most of the time running time complexity of hybrid Pearson measure is lower than weighted Euclidean distance.

Most of the algorithms do not compute distance between clusters and data points at every iteration, unless a cluster is updated at the assignment step and running time values included in Table 6.1 includes time spent outside the main loop of the algorithms, as well. Therefore, the ratio of total running time to iterations until convergence is not a strictly informative measure of running time comparison.

However, SOM clustering computes distance between every node and every input vector at all iterations of the training step without any special update rules. Although random initialization of the map and time spent to perform the Hierarchical clustering in the post-clustering step contribute to measured running time, SOM clustering on every data set converges more quickly for Euclidean framework than the latter. Similarly, hybrid Pearson framework attains lower running times than the weighted Euclidean distance measure.

The time spent outside the main loop of algorithms belonging to the K-means clustering, family except Similarity-Based K-means, contributes considerably to overall running times. This is more obvious in case of K-medoids clustering, in which almost all the time is spent in computing data point distance matrix and setting initial medoids, while the algorithm is reported to converge quickly in its main loop.

Time spent on the main loop of an algorithm similar to K-means can be tracked via the table entries regarding Hopfield-K-means clustering. For this algorithm, training step to set the cluster centers takes T_t ms, which is actually around 600 ms and is not reported in the table due to its invariant nature regarding distinct distance frameworks. Running times of Hopfield-K-means clustering point out that more than one third of the computational time must be spent on setting initial cluster centers by K-means++ and similar algorithms.

Similarity-Based K-means clustering attains substantially greater complexities than the rest of the K-means family due to vast computation resources required by the point symmetry distance. Moreover, the algorithm requires larger number of iterations prior to convergence. The results summarized in the table suggest not using Similarity-Based K-means clustering when running time issues have priority.

A comparison between running times of hybrid Pearson and weighted Euclidean distance frameworks on algorithms of the K-means family favors the weighted Euclidean distance measure for most of the cases. However, it could be claimed that average number of cluster-changing data points at each cycle of the main loop in hybrid Pearson framework is higher than weighted Euclidean, causing the metric to be computed more often than the rest. This claim is also supported by the observation regarding the case of SOM clustering, in which both distance measures are computed equal number of times and time complexity of weighted Euclidean measure supersedes that of hybrid Pearson measure. In order to make up for the gap, hybrid Pearson distance computation must have been carried out more frequently than weighted Euclidean by K-means family algorithms.

ISODATA either splits few initial clusters or merges many initial clusters to finally form desired number of clusters. Thus, its running time evaluation can differ depending on the operation performed and it is not decisive about the performance of a particular distance framework, as the corresponding table entry indicates.

Fuzzy K-means has higher running time complexity than most members of the K-means family due to necessary computation of degree of belonging matrix for all data points to all clusters at every cycle. In some cases, convergence may come after many iterations, contributing to prolonged running times.

As a deterministic clustering algorithm, Follow-the-Leader method attains low running times and converges quickly. The fact that all algorithms alike K-means are run several times in search for the clustering with the best quality due to randomness factor, or to the need for deciding the best constants to set initial cluster centers does not hold for Follow-the-Leader clustering. Therefore, Follow-the-Leader clustering turns out to be promising in terms of lower computational complexities, as well.

Hierarchical clustering running time per iteration is less than rest of the algorithms since only one cluster is updated at each iteration, the distance is computed from previous distance values in linear time at an instant, and no cluster center update is performed until the end of the algorithm. However, number of iterations is very high because the algorithm works in an agglomerative fashion, merging two closest clusters at one time. Initial running times of the algorithm is especially high since the method operates on every pair of data points at the beginning of the algorithm. One advantage of the algorithm is that, in one pass all possible number of clusters are generated, as stored in dendrograms so that several re-runs are not obligatory.

SOM clustering, similar to Hierarchical clustering, offers the possible re-use of the trained map. Training of the map in 90 iterations contributes mostly to the overall running time. SOM generates a reduced data set residing in a 2-D map from the actual data, which is further clustered by a clustering algorithm. In this study, Hierarchical clustering with UPGMA linkage is employed to do the post-clustering deed, which requires 80 iterations to generate clustering of 90 nodes in the map into 10 clusters.

Change in running times of winter weekdays data clustering with respect to increasing number of clusters is summarized in the table below.

Table 6.2 Running times of clustering of winter weekdays data in Euclidean framework with increasing number of clusters.

	Running time (ms) in Euclidean framework			
	5 clusters	10 clusters	15 clusters	20 clusters
K-Means	28	47	47	52
K-Means++	9	52	48	84
K-Medians	12	73	52	74
WFA-K-Means	23	43	54	93
Hopfield-K-Means	$16+T_t$	$29+T_t$	$18+T_t$	$19+T_t$
K-Medoids	43	53	70	94
Similarity-Based K-Means	976	7290	2342	7140
ISODATA	205	248	108	141
Fuzzy K-Means	255	347	2174	2663
Follow-The-Leader	52	57	64	N/A
Hierarchical (UPGMA)	141	157	145	162
SOM (Sequential 6 15)	6995	5712	5952	6183
SOM (Batch 6 15)	7582	7257	6935	7491

Only the Euclidean distance framework is covered in Table 6.2, because it is determined to provide the means of fastest clustering among all other frameworks, as verified by the data in Table 6.1. Most of the time, as number of clusters increase, total running times increase for K-means family, Fuzzy K-means clustering, and Follow-the-Leader algorithm.

In certain cases, convergence of the algorithm may occur in less number of iterations or less number of changes in cluster compositions per iteration for relatively higher number

of clusters, decreasing the total running time. An example case is 15 clusters generated by K-means++, K-medians, Similarity-Based K-means and Hopfield-K-means.

Running times associated with Fuzzy K-means clustering increase substantially after 10 clusters, which is a discouraging factor for adopting the algorithm if higher number of clusters is desired.

Differences in running times of SOM clustering reported in the table is mostly due to random initialization of the map; therefore the effect of increasing number of clusters is not directly observed. On the other hand, in all cases batch training of the map takes modestly longer times than sequential training.

Although non-determinism is not an issue in case of Hierarchical clustering, at one pass the algorithm generates the whole hierarchy of clusters for the data. Therefore, a healthy observation regarding the affect of increasing number of clusters over measured running times cannot be made merely by examining the corresponding table entries.

6.4 Running Times of Algorithms on Synthetic Data

Running times distinct cluster algorithms take to group instances of synthetic data sets into 10 clusters in Euclidean distance framework is summarized in Table 6.3. All running times are measured in milliseconds. Hopfield-K-means and ISODATA algorithms are excluded from tests conducted on synthetic data, since computational complexity of the first algorithm is dominated by the training step of the Hopfield network while the remaining steps are identical to K-means, and ISODATA clustering includes distinct mechanisms of producing the final clustering, which is complicated to assess.

Table 6.3 Running times of algorithms on synthetic data sets in Euclidean framework, generating 10 clusters

		Synthetic Data 1 (883 instances)			Synthetic Data 2 (1613 instances)			Synthetic Data 3 (3222 instances)		
		Run-time	Iter #	Per-iter	Run-time	Iter #	Per-iter	Run-time	Iter #	Per-iter
K-means	Init	70	-	-	100	-	-	162	-	-
	ML	415	30	13	1410	61	23	2996	63	47
K-means++	Init	124	-	-	209	-	-	426	-	-
	ML	153	12	12	296	12	24	2181	42	51
K-medians	Init	120	-	-	206	-	-	394	-	-
	ML	288	19	15	748	23	32	1483	23	64
WFA-K-means	Init	135	-	-	222	-	-	402	-	-
	ML	219	9	24	999	23	43	2575	33	78
K-medoids	Init	783	-	-	2352	-	-	9057	-	-
	ML	26	2	13	43	2	21	84	2	42
Similarity-based K-means	Init	130	-	-	211	-	-	396	-	-
	ML	67181	35	1919	500559	47	10650	2176690	48	45347
Fuzzy K-means	Init	505	-	-	668	-	-	1269	-	-
	ML	1101	5	220	4214	13	324	7773	14	555
Follow-The-Leader		422	6	70	1997	15	133	3605	17	212
Hierarchical (UPGMA)		16651	873	19	110381	1603	68	796563	3212	247
SOM (Sequential)		63066	(90, 140)	5	181973	(90, 200)	10	450002	(90, 275)	18
SOM (Batch)		68486	(90, 140)	5	171396	(90, 200)	9	456273	(90, 275)	18

The data included in Table 6.3 demonstrates changes in running times of algorithms with respect to increasing number of data set sizes. Number of iterations until convergence and running time per iteration are included together with measured running times. If the algorithm does not start directly with a main loop, i.e. centroid initialization and distance matrix computation phases precede the main loop as in K-means family and Fuzzy K-

means; the time spent prior to and within the main loop are indicated separately in the table with *Init* and *ML* marks, respectively.

Scalability of the algorithm can be tracked via per iteration time, because this value corresponds to a parameter of computational complexity of the algorithm with respect to only the size of the data set. Doing so, number of iterations is filtered out, while the number of clusters and dimensionality of data are constant factors. Change in number of iterations as data set size increases is affected by both innate mechanisms of the algorithm and statistical properties of the data set. Therefore, it cannot be manipulated directly and its effects on scalability are not covered in the scope of the thesis. Moreover, running times of each cycle are not the same due to variation in number of cluster-changing points or merging clusters. Hence, per iteration time actually corresponds to an average value. For these reasons, scalability comparison performed in this section is rather constrained.

All members of the K-means family except Similarity-Based K-means clustering and K-medoids turn out to be quite scalable, with around a two-time increase in running times as the data sizes double, suggesting the time complexities of algorithms are linear in data size. Note that algorithms utilizing centroid initialization in a K-means++ fashion tend to converge in less number of iterations compared to the original K-means algorithm initializing centroids by constants. Therefore, total running times associated to algorithms like K-means++ are less than K-means.

On the other hand, K-medoids algorithm spends most of its time in computing data point distance matrix, causing a four-time increase between running times of the second and the first, and the third and the second data sets, validating the complexity of data matrix computation to be $O(N^2)$. For this reason, per iteration running time is not enough to decide K-medoids to be scalable to increasing amounts of data. For the rest of the algorithms in K-means family, the time spent outside the main loop is also scalable with less than two times increase as data sizes double.

Similarity-Based K-means spends most of the computational time in its fine-tuning phase. Actually, the time spent for initialization and coarse-tuning is insignificant. Judging by the per iteration time included in Table 6.3, Similarity-Based K-means clustering is not at all scalable to increasing amounts of data, attaining at least a four-time increase in per iteration time as the data doubles. This verifies that the complexity of fine-tuning phase per iteration whenever number of clusters is kept fixed is at least $O(N^2)$ due to point symmetry distance.

Per iteration time comparison among K-means, K-medoids, K-means++, K-medians and WFA-K-means reveal that medoid computation and centroid computation by taking feature-by-feature means of all member data points take more or less the same computation times, while centroid computation utilizing median operator is slightly more time-consuming. On the other hand, computation of weighted fuzzy average to be set as a cluster center takes clearly more time than the rest. Yet, the affect of distinct cluster

center computation techniques over number of iterations until convergence cannot be directly observed from Table 6.3. Running time evaluation suggests that centroid initialization has a crucial impact on convergence.

Fuzzy K-means clustering, like its relatives in K-means family carrying out hard-clustering, is scalable with respect to both main loop time per iteration and the non-main loop time. However, time spent per iteration exceeds greatly that of all members in K-means family except Similarity-Based K-means clustering; decrementing the chance of selection of the algorithm, if running time is the only critical issue.

Follow-the-Leader clustering may converge after distinct number of iterations due to characteristics of the data sets when the distance threshold operates over data points in the set. The algorithm tends to converge more rapidly than the algorithms in the K-means family. Follow-the-Leader does not have a cluster center initialization phase; therefore main loop running time is equal to total running time. The algorithm is quite scalable with less-than two-time increase in per iteration running times as the data sizes double. Although per iteration time is higher than K-means family due to lack of a selectively updated distance matrix mechanism, Follow-the-Leader method has a deterministic nature and does not require being re-run several times in order to find the best clustering.

Hierarchical clustering does not scale well to larger data sets, as the corresponding running time entries in Table 6.3 demonstrate. Between a three-time and a four-time increase is observed for per-iteration computational complexity. However, this is not a good measure of time complexity with respect only data set sizes, because at the beginning of the algorithm distance matrix is of size *N*-by-*N*, shrinking towards its final size of 10-by-10 and all algorithmic processes take place at the space characterized by the matrix size. Therefore, the time spent in the first iterations of the algorithm is much greater than average running time of the last iterations.

Nonetheless, as data set sizes increase, both number of iterations before convergence and per iteration time increase, causing Hierarchical clustering to be clearly unscalable to increasing data. If running time is a prior concern, and the data size is large, it is not wise to use Hierarchical clustering, although the algorithm will be a good choice for smaller data sets due to its low running times and deterministic nature providing that only one run of the algorithm is enough before the evaluation. An example to the explained trade-off is that winter weekdays data having 305 instances is clustered by the Hierarchical algorithm within 157 ms, and the first synthetic data set having almost three times more instances is clustered in 16651 ms, demonstrating poor scalability of the algorithm.

Training of the map in SOM clustering takes most of the computational time, because post-clustering of the map using Hierarchical algorithm with UPGMA linkage is carried out on at most 270 instances, the map size for the third synthetic data, which is less than the size of the winter weekdays data set that was grouped into 10 clusters in 157 ms.

Therefore, main loop time is selected as the training time of SOM and is assumed to be equal to the total running time in Table 6.3.

Per iteration time for SOM is computed as the training time divided by number of training iterations which is 90 for all maps, to be further divided by the number of nodes in the maps. In this case, per iteration time is actually per iteration per node time so that number of iterations and the map size is filtered out to observe the change in running time with increasing sizes of data. Map sizes are 140, 200, and 275 for the maps of the first, second, and the third synthetic data sets, respectively.

Modified per iteration time fields of Table 6.3 indicate that training of the SOM is a linearly scalable process with respect to increasing amounts of data. Yet, SOM training takes longer times than most of the algorithms. Nevertheless, trained self-organizing map can be stored for post-processing by several clustering algorithms. It is merely a process of reducing data set sizes that takes relatively longer time. In our case, 883 instances of the first data set are reduced to 140 node vectors, 1613 instances of the second data set are reduced to 200 node vectors, and 3222 instances of the third data set are reduced to 275 instances; which can be clustered by Hierarchical algorithms in less than 157 ms.

As a summary, all members of the K-means family excluding Similarity-Based K-means and K-medoids; Fuzzy K-means, and Follow-the-Leader algorithms together with SOM training tend to be linearly scalable with respect to size of the data set; whenever number of clusters and dimensionality of the data are kept constant, and the number of iterations until convergence is utilized as a normalizing factor, i.e. filtered-out from the measured running times.

6.5 Fundamental Approaches to Decrease Running Times

In this section, effects of utilizing parallel versions of clustering algorithms, designing hybrid algorithms, and reducing data size or dimensionality on running times and scalability are discussed.

6.5.1 Utilizing Parallel Versions of Algorithms

K-means clustering is scalable and offers the potential of being run in parallel (Farnstorm et al., 2000). Since K-means utilized in the thesis sets its initial centroids through constants, sequential and concurrent versions of the algorithm will behave deterministically given that the constants have the same values. Not any other member in K-means family can be controlled in this manner, as they include randomness factor. Therefore, gain obtained using parallel algorithms is exemplified through devising a parallel version of the K-means algorithm, whose sequential version is described in Chapter 4.

In order to parallelize K-means algorithm, concurrency mechanisms of Java environment (Göetz et al., 2006) is utilized, as all clustering methods are coded in Java. The idea behind a parallel version of the algorithm lies in aiming at a more efficient use of the processors by reducing the workload of time-consuming algorithmic steps through distributing data points among available processing units.

Time-consuming parts of K-means are classified as distance computation and assignment making steps. Data points residing in the set are distributed over existing processors and each processor does its work concurrently until corresponding processes meet at rendezvous points, where synchronization is controlled by a barrier construct.

In order to ensure mutually exclusive access to shared resources, related functions or code segments are marked with Java `synchronized` keyword, practically serving as a mutex construct of concurrent programming literature. Accuracy of the parallel version of the algorithm can be assessed by the output clustering generated, which is the same as clustering of sequential K-means since algorithmic flows is not altered.

Table 6.4 Running time comparison of sequential and concurrent versions of K-means partitioning synthetic data sets into 10 clusters in Euclidean framework.

	Synthetic Data 1 (883 instances)				Synthetic Data 2 (1613 instances)				Synthetic Data 3 (3222 instances)			
	Run-time	Main time	Iter #	Per iter	Run-time	Main time	Iter #	Per-Iter	Run-time	Main time	Iter #	Per-iter
K-M	485	415	30	13	1510	1410	61	23	3158	2996	63	47
C K-M	313	243	30	8	1001	905	61	14	1905	1781	63	28

Around a 40% decrease in per iteration running times is achieved by utilizing parallel version of the K-means algorithm as indicated by the data in Table 6.4, in which *K-M* stands for K-means and *C K-M* stands for concurrent K-means. The amount of decrease increases modestly as the data sizes double and is more pronounced in case of the third synthetic data set, suggesting that concurrent K-means may attain even lower running times for much larger sets.

As each thread running concurrently does its work on an equal-size portion of the data, computational complexity of the algorithm remained the same and scalability tendency of K-means is not changed by utilizing parallel version of the algorithm. However, more tests with larger data sets are essential to prove the claim, which is not covered by the scope of this work.

To sum things up, a notable decrease in running times is attained, proposing the design and use of concurrent versions of algorithms when large data segments are to be clustered.

6.5.2 Designing Hybrid Algorithms

Hierarchical clustering has been analyzed to be not scalable to larger data and it has been mentioned that running time complexity is especially higher at earlier steps of the algorithm where $O(N^2)$-computation is performed on each pair of data points. Hybridizing Hierarchical clustering with other algorithms may ensure that the algorithm will start with less number of data points.

One way to provide less number of data points as inputs to Hierarchical clustering is running a scalable clustering algorithm as the initial step of the new hybrid algorithm to generate a particular number of initial clusters proportional to the desired number of clusters. Then, Hierarchical algorithm is run using centers of initial clusters as data points, until desired number of clusters is yielded. Following the convergence of the Hierarchical clustering at this second phase, final clusters are obtained by adding all members of the clusters of initial clusters to their correct positions.

Hierarchical clustering is hybridized with K-means++ and Follow-the-Leader algorithms respectively. Hybrid Hierarchical–K-means++ clustering follows the steps included subsequently.

- **Step 1**: Use K-means++ described in Section 4.2.2 to generate cK clusters, where c is a constant used to determine the number of clusters obtained at the end of the first phase.

- **Step 2**: Utilize Hierarchical clustering as indicated in the following steps.

    - **Step 2.1**: Take clusters of the first phase as the initial clusters of the Hierarchical algorithm.

      $$K = cK. \tag{6.1}$$

      $$I_{max} = N + I_f, \;\; I = 1 + I_f \,,$$

      $$where\ I_f is\ the\ convergence\ iteration\ number\ of\ the\ first\ phase.$$

    - **Step 2.2**: Build up the distance matrix. Compute the distance between data points and clusters.

$$Construct\ an\ cK \times cK\ distance\ matrix\ D = [d_{ij}]. \qquad (6.2)$$

$$d_{ij} = d(C_i, C_j), \qquad \forall i, j \in [1\,..\,cK].$$

- **Step 2.3:** Proceed with **Step 2 - Step 5** of the Hierarchical clustering algorithm described in Section 4.2.11.

An example clustering of hybrid Hierarchical–K-means++ algorithm is illustrated in the following for 10 clusters on the second synthetic data set in Euclidean framework.

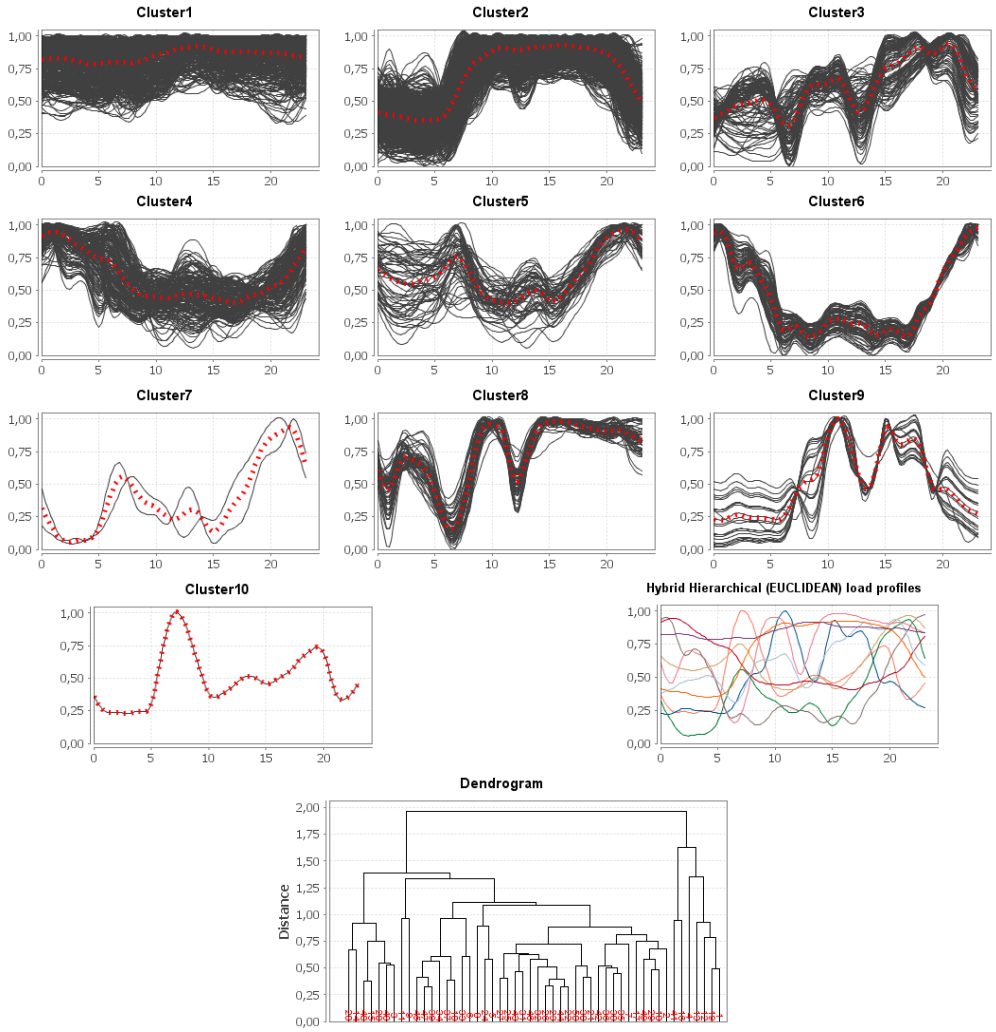

Figure 6.2 Clustering of the second synthetic data set by Hierarchical–K-means++ algorithm in Euclidean distance framework.

Dendrogram in (Fig 6.2) is marked with cluster indices of the first phase in which 50 clusters are generated. For every clustering and evaluation result, Hierarchical algorithms are run with UPGMA linkage.

As number of clusters to be generated in the first phase increases, performance of K-means++ deteriorates because more and more computational time is spent for setting initial cluster centers, which could be at worst $O(K^2)$ in only number of clusters. Another problem is the introduced randomness to deterministic Hierarchical clustering, which may require several re-runs of the algorithm. This is a burden regarding the algorithm's appliance to higher number of clusters.

Hybridizing Hierarchical clustering with Follow-the-Leader algorithm, however, may be a solution to limitations of K-means++ since Follow-the-Leader operates on a distance threshold, with no cluster center initialization phase. Moreover, the algorithm is deterministic unless the order of the data points changes.

Hybrid Hierarchical–Follow-the-Leader algorithm follows the same steps as Hierarchical–K-means++ clustering, with only the obligatory modification to the first step as follows.

- **Step 1**: Use Follow-the-Leader clustering described in Section 4.2.10 to generate at least cK clusters, where c is a constant determining the number of clusters obtained at the end of the first phase. If generated clusters are less than cK, decrement the operating distance threshold by a suitable amount and re-run the algorithm. Else continue with **Step 2** of Hierarchical–K-means++ clustering.

Hybrid Hierarchical–Follow-the-Leader clustering is illustrated subsequently.

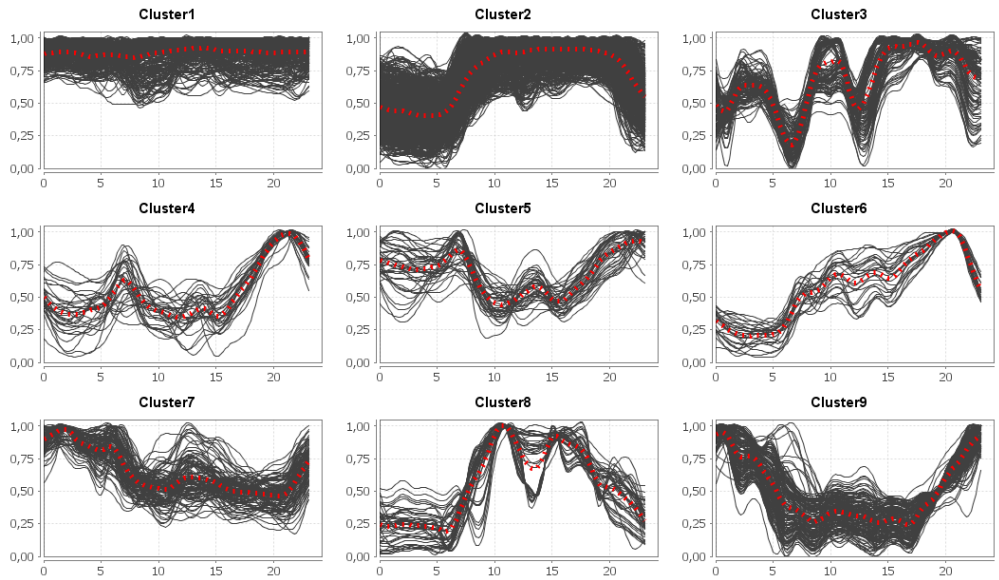

Figure 6.3 Clustering of the second synthetic data set by Hierarchical–Follow-the-Leader algorithm in Euclidean distance framework.

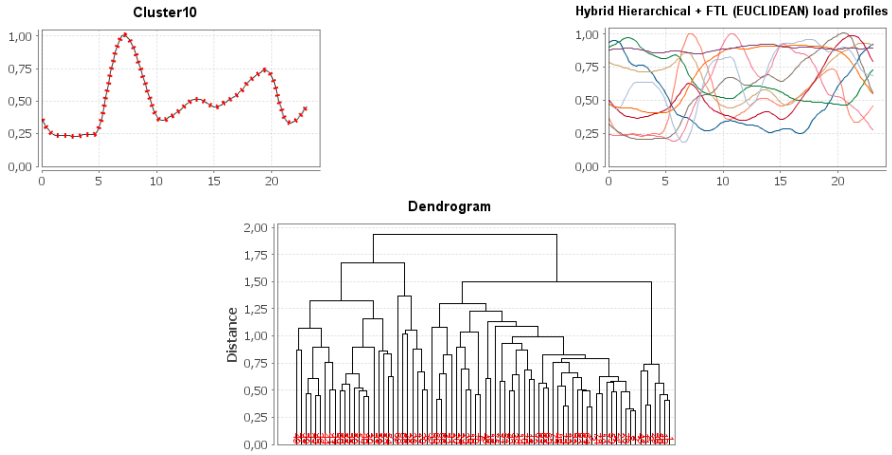

Figure 6.3 Clustering of the second synthetic data set by Hierarchical–Follow-the-Leader algorithm in Euclidean distance framework. (cont'd)

Distance threshold is set to 0.60 and 78 clusters are generated at the end of the first phase to be included as leaves of the dendrogram in (Fig. 6.3) that shows the history of the process of Hierarchical clustering in the second phase of the algorithm.

Clustering of the second synthetic data set by hybrid Hierarchical algorithms illustrated in (Fig. 6.2) and (Fig. 6.3) resemble each other. However, clusters generated by Hierarchical–Follow-the-Leader algorithm appear more compact. Quality of two sets of clusters is evaluated as follows.

Table 6.5 Evaluation of 10 clusters generated by Hierarchical and Hybrid Hierarchical algorithms on the second synthetic data set in Euclidean framework.

	DBI	MDI	CDI	SMI
Hierarchical	1.17	0.75	0.48	0.64
Hierarchical–K-means++ (50)	1.04	0.75	0.46	0.64
Hierarchical–Follow-the-Leader (78)	1.09	0.74	0.48	0.64

As evaluation results included in Table 6.5 indicate, hybrid clustering algorithms perform at least as good as the original Hierarchical clustering algorithm. Regarding DBI metric, even a moderate decrease is attained, promoting the use of hybrid algorithms for clustering large data sets.

Running time comparison of hybrid algorithms and the original Hierarchical algorithms are included in the following table.

Table 6.6 Running time evaluation of original and hybrid Hierarchical algorithms generating 10 clusters on synthetic data sets in Euclidean framework.

		Synthetic Data 1 (883 instances)			Synthetic Data 2 (1613 instances)			Synthetic Data 3 (3222 instances)		
		Phase times	Iter #	Per iter	Phase times	Iter #	Per-Iter	Phase times	Iter #	Per-iter
Hierarchical		16651	873	19	110381	1603	68	796563	3212	247
Hierarchical–K-means++ (50)	Init	1958	-	-	3659	-	-	7238	-	-
	P1	621	28	22	967	18	53	2776	23	120
	P2	348	40	8	372	40	9	343	40	8
Hierarchical–K-means++ (100)	Init	8104	-	-	14561	-	-	28580	-	-
	P1	756	16	47	1755	23	76	9123	58	157
	P2	422	90	4	404	90	4	388	90	4
Hierarchical–Follow-the-Leader (~80)	P1	963	6	160	2508	10	250	7754	19	408
	P2	380	71	5	373	68	5	396	70	5
Hierarchical–Follow-the-Leader (~160)	P1	1678	6	279	5333	12	444	11286	14	806
	P2	468	149	3	470	149	3	468	151	3

Running times of Hierarchical–K-means++ algorithm generating 50 and 100 clusters, and Hierarchical–Follow-the-Leader algorithm yielding around 80 and 160 clusters by the end of the first phase are independently shown in Table 6.6. Running times regarding the first and second phases of the hybrid algorithms are separately indicated by *P1* and *P2* marks. In case of Hierarchical–K-means++ algorithm, running time of the initialization step is also shown in distinct rows marked with *Init*, for which no information on iteration numbers or per iteration running time is included.

The values in the table demonstrates that both hybrid algorithms attain much lower running times than the original Hierarchical clustering method. Evaluation methodology dictates that for each synthetic data set, number of clusters generated by the end of the first phase should be the same, making per-iteration time regarding the second phase a constant, regardless of the size of the set. Therefore, both hybrid algorithms are scalable, as K-means++ and Follow-the-Leader algorithms have been shown to be scalable when number of iterations and clusters are assumed to be fixed.

Per iteration times regarding the first phase of the algorithm verifies the scalability of K-means++ and Follow-the-Leader methods, as previously claimed. Corresponding running time data in Table 6.6 also shows that Follow-the-Leader is slightly better scalable than K-means++.

As number of first-phase clusters increases, per iteration time regarding the second phase decreases for both hybrid methods. However, for Hierarchical–K-means++ total running time increases by a factor of four in all data sets as number of clusters doubles, due to the complexity of the initialization phase of the K-means++ algorithm. On the contrary, Hierarchical–Follow-the-Leader clustering turns out to be scalable to doubling number of clusters as well as doubling data size according to the per iteration time heuristic.

Although higher numbers of clusters are generated for test purposes following the end of the first phase of Hierarchical–Follow-the-Leader algorithm, this method has a substantially lower total running time than Hierarchical–K-means++. Moreover, it offers deterministic clustering unless the order of the data points in the set is changed. Once the distance threshold is correctly selected, there is no need for further re-runs of the algorithm in order to improve the quality of a clustering.

A crucial point to keep in mind is that number of clusters generated at the end of the first phase must never be close to the actual data size so that Hierarchical clustering will not dominate the total running time. Although this effect is not observed in Table 6.6, more elaborated tests should be conducted to provide the means of devising genuine methods that automatically select the number of clusters to be generated in the first phase of the hybrid algorithm, without increasing the running time complexity significantly.

Running time speedup acquired by hybridizing Hierarchical algorithm with K-means++ or Follow-the-Leader is much higher than the one that could be gained by utilizing a parallel version of the original method, which is a more intriguing task than parallelization of K-means due to the difficulty of effective partitioning in Hierarchical clustering (Dash et al., 2004). Nevertheless, parallel versions of the proposed hybrid algorithms may be a positive contribution to further decreasing overall running times.

6.5.3 Decreasing Data Size or Dimensionality

Data sets can be projected into lower dimensions by preserving associated topological properties. Self-organizing maps covered in the thesis effectively carries out such a projection of multidimensional space into 2-D. As previously discussed, once computed, SOM may be stored so that map node vectors, rather than actual data points, are clustered several times. Since best matching unit of multiple data points may be the same node in the map, all points that are mapped into the same node are then added to the cluster to which the vector of that node, belongs in only one pass over the data set.

Recalling that 883 instances of the first synthetic data set are reduced to 140 node vectors, 1613 instances of the second data set are reduced to 200 node vectors, and 3222 instances of the third data set are reduced to 275 instances; Hierarchical clustering as the least scalable algorithm utilized in the thesis will complete the clustering of the third synthetic data set, the largest of all, within less time than it takes the algorithm to cluster winter weekdays data set.

Working in the reduced data set space does actually diminish the burden of running times outstandingly. However, accuracy of clustering decreases as well. Trade-off regarding the accuracy of utilizing SOM node vectors in clustering instead of actual data points is reported in detail in Section 5.4.12.

Among other methods to reduce data size or dimensionality; employing nine load-shape indices rather than 24 features, other algorithms of multidimensional scaling such as Principal Component Analysis and Sammon mapping (Chicco et al., 2006), and variable subset selection techniques to eliminate redundant features present in data points exist. Nonetheless, speed-accuracy trade-off should always be taken into consideration whenever data reduction is applied.

6.6 Summary

Running time complexities and scalability of clustering algorithms are analyzed in-depth throughout the chapter, which can be crucial issues if the clustering is to be applied for a larger scale of customers.

Synthetic data sets are constructed based on models of the real electricity consumption values in order to compensate for the need of increasing amounts of data to evaluate scalability of algorithms. In addition to that, scalability of the method to increasing number of clusters is briefly discussed.

Pointing out true computational complexity of a clustering algorithm requires founding mathematical relations between number of iterations and other parameters including statistical properties of the data sets in use. Since these relations are not covered within the scope of this thesis, per-iteration running time is utilized as a measure of scalability for the majority of algorithms, explicitly stating the situations in which per iteration running time comes short in analyzing scalability.

Using parallel versions of algorithms, designing hybrid algorithms, and working in a reduced data space with possibly lower dimensionality are examined as solutions to decrease running times of algorithms. Two new hybrid algorithms, Hierarchical–K-means++ and Hierarchical–Follow-the-Leader clustering, are designed and evaluated which decrease computational complexity of Hierarchical clustering substantially.

Moreover, both hybrid methods are reported to increase scalability to increasing amounts of data compared to the original algorithm.

For elaborated studies regarding scalability of K-means++ and other clustering algorithms, please refer to (Bahmani et al., 2012) and (Farnstorm et al., 2000).

CHAPTER 7

CLASSIFICATION OF LOAD PROFILES AND RELATED APPLICATIONS

7.1 Introduction

Clustering of representative load diagrams and discovery of representative load profiles computed as centers of the generated clusters are used in a variety of applications targeted at both customers and distribution companies.

In this chapter, customer recognition based on further classification of clustered load diagrams and fundamentals of dynamic tariff design are analyzed. Requisites of other applications are also briefly discussed.

7.2 Customer Recognition

Clustering of customers can be processed in order to learn the characteristics of consumption shapes. For this purpose, clustering should be organized as labeled data, where labels are names of the clusters for each data point in the set. Labeled data is then inputted to supervised classification algorithms.

Customer classification enables one to find out reasons of existence of a data point within a particular cluster, and not in any other. Moreover, classification provides the means of generalization so that new customers are placed into the correct cluster, although they have not participated in the clustering operation.

Classification method employed in customer recognition is the well-known decision tree induction algorithm C4.5 (Quinlan, 1996). The algorithm utilizes the notion of information entropy in such a way that an attribute that yields higher information having split training instances into particular intervals of its value is placed in a position closer to the root in the tree. C4.5 can handle continuous attributes as in electricity consumption data sets. Following the creation of the tree, further pruning is performed so that tree size is reduced, over-fitting to training data is avoided and acquired learning has a more general nature in recognizing further test instances.

In the thesis, J48, an open source Java implementation of C4.5 available in WEKA machine learning framework (Witten et al., 2011) is used. Both actual features of data points and shape indices characterizing consumption styles as which are defined in

Chapter 2 are selected as attributes, as proposed in (Ramos and Vate, 2008). The control parameter of the post-pruning, namely the *confidence factor*, is set to 0.25, and minimum number of instances with a particular label is selected as 1 throughout the trials conducted in this section.

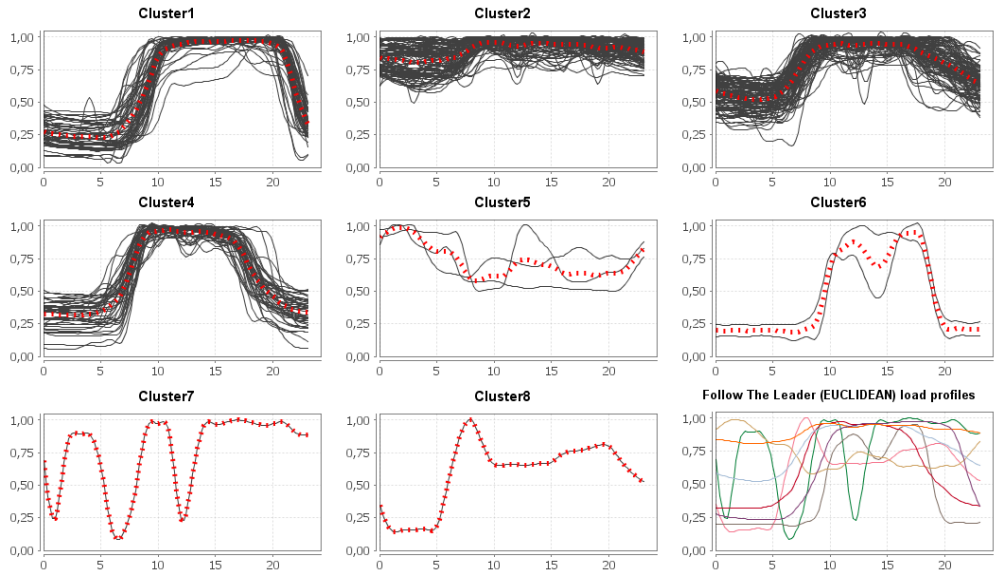

Figure 7.1 Clustering of winter weekdays data by Follow-the-Leader algorithm in Euclidean framework.

Clustering of winter weekdays data by Follow-the-Leader algorithm for eight clusters in Euclidean distance framework as illustrated in (Fig. 7.1) is selected to demonstrate customer recognition application.

Normalized features of all data points in the set and computed shape indices during the course of clustering are used as distinct sets in the classification. Labels are cluster names as previously mentioned.

The tree induced and pruned by the algorithm by using shape indices as attributes is included in the following.

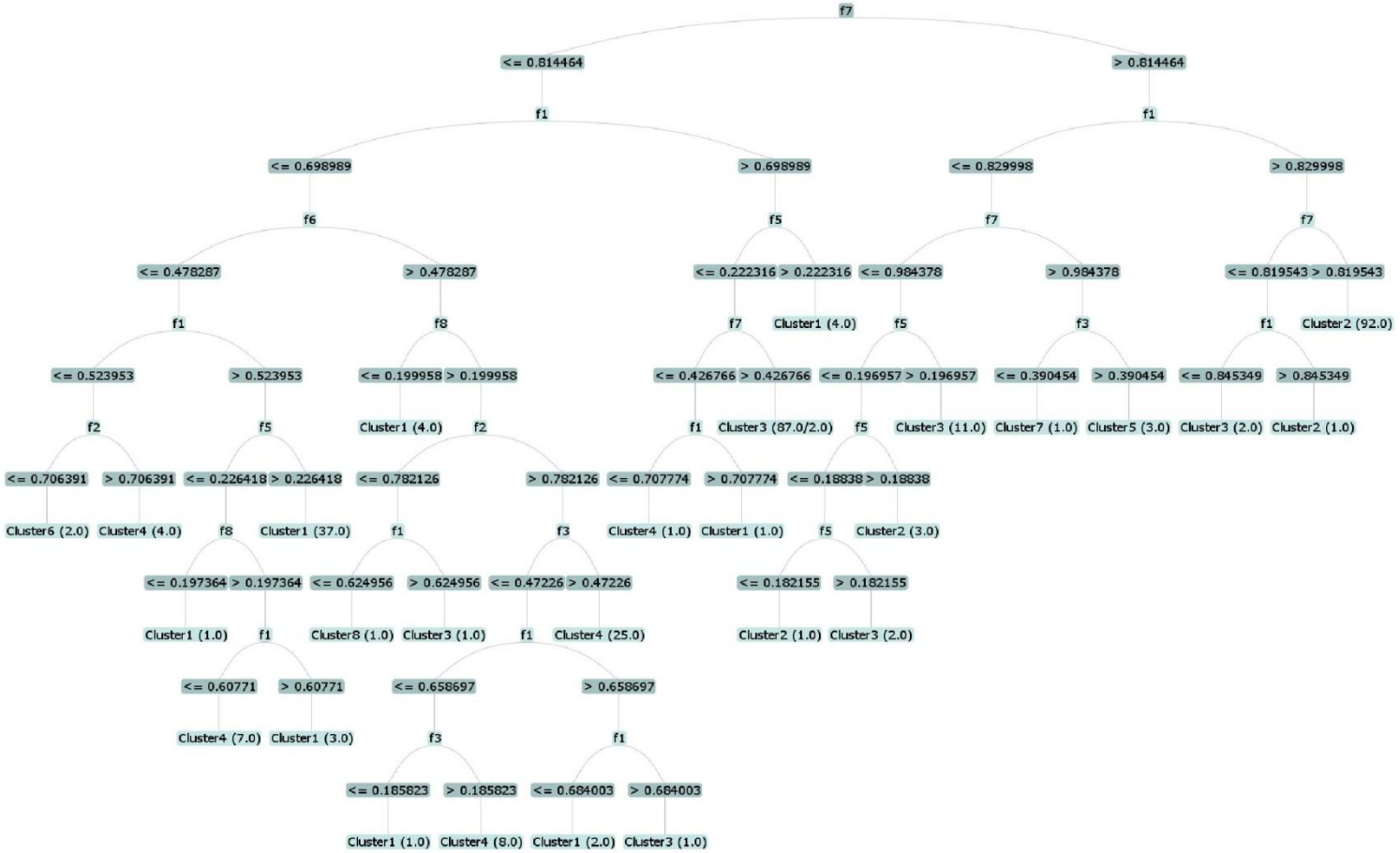

Figure 7.2 Decision tree for classifying winter weekdays consumption using clustering of Follow-the-Leader algorithm with load shape indices.

The tree included in (Fig. 7.2) is created by using all instances as the training set. For most of the clusters there are several paths. However, the majority of data points of a large cluster are all grouped in one leaf node. Classification error for each cluster is also included in (Fig. 7.2) as the number of incorrectly classified instances. If that number is greater than zero, it is placed in the leaf node just after the all number of instances residing in the node followed by a slash mark. In this case, the only error appears in a branch yielding *Cluster 3* for two instances.

The tree can be interpreted by following all the nodes starting from the root up to a certain leaf depicted by a rectangular shape with a lighter color of blue. In this manner, rules for testing particular cluster memberships are generated by taking non-leaf nodes combined through conjunction with each other in an if clause, as exemplified below for *Cluster7*, a singleton outlier cluster generated by Follow the Leader method.

$$if\ f7 > 0.814464\ and\ f1 \leq 0.829998\ and\ f7 > 0.984378\ and\ f3 \leq 0.390454\ , \tag{7.1}$$

$$then\ the\ instance\ belongs\ to\ Cluster\ 7.$$

Note that in course of rule generation for a particular class label, redundancy may be present since the tree is induced for all labels. Redundancy in (7.1) can be eliminated by reducing two appearances of $f7$ in the if-clause into one, as in the following.

$$if\ f7 > 0.984378\ and\ f1 \leq 0.829998\ and\ f3 \leq 0.390454 \tag{7.2}$$

$$then\ the\ instance\ belongs\ to\ Cluster\ 7.$$

The decision tree yielded by C4.5 does its partition of clusters for the shape index $f7$ at the root position. This index is the rate of average night consumption to average daylight consumption. If average daylight consumption exceeds that of night more than a certain threshold value, a data point is assigned to a different set of clusters most of the time. However, *Cluster 1* and *Cluster 3* hold quite a few members in both sides of the branching of $f7$ values. Lunch time impact, daily average consumption with respect to the peak demand, minimum consumption divided by average of all hours are among other shape indices appearing in the tree that characterize the mechanisms behind clustering.

According to the definitions of load shape indices $f7$, $f1$, and $f3$, a possible capture of the semantics regarding the rule generated for the label *Cluster 7* may correspond to the following degrees of power consumptions during particular hours.

$$if\ P_{av,night}\ is\ high\ and\ P_{av,daylight}\ is\ medium\ and\ P_{av,day}\ is\ medium \quad (7.3)$$

$$and\ P_{max,day}\ is\ high\ and\ P_{min,day}\ is\ low, then\ the\ instance\ belongs\ to\ Cluster\ 7.$$

The justification of the rule (7.3) can be made as follows. The index $f7$ is the ratio of average night to average daytime consumption. Since it should be high for an instance of *Cluster 7*, average night consumption of the customer shall be high, while average day light consumption could be at most medium. The shape index $f1$ is computed as the ratio of average over maximum daily consumption. The rule indicates that $f1$ shall be high but not too high. Thus, average daily consumption could be at least a medium value, while maximum daily consumption is high. Lastly, $f3$ is the ratio of minimum over average daily consumption amounts. Since this index should have a low value according to the rule, minimum daily consumption can only be low, because average consumption is decided to be medium before.

If a new customer whose data is unavailable is to enter the system, their own opinions on consumption amounts during night, daylight, and lunch times together with minimum, maximum, and average amounts of electricity they consume daily can be determined through a questionnaire so that an initial prediction is made about the cluster they most likely belong to.

Discrete values of *high*, *medium* and *low* can be defined in accordance with the branching criteria of the decision tree. In order to reveal more rules concerning all customers, more elaborated degrees than only high, medium and low shall be utilized depending on the distinct number of values an attribute can take at branching nodes. Moreover, a conflict resolution strategy shall be developed to avoid contradictions. However, it would be wise to keep the number of degrees limited so that questionnaires will not be too detailed, since this may result in distracting the customers and causing higher errors in predicting the initial cluster.

Decision tree obtained by using all features of the data points as attributes and names of the clusters generated by Follow-the-Leader algorithm as labels is presented in the following.

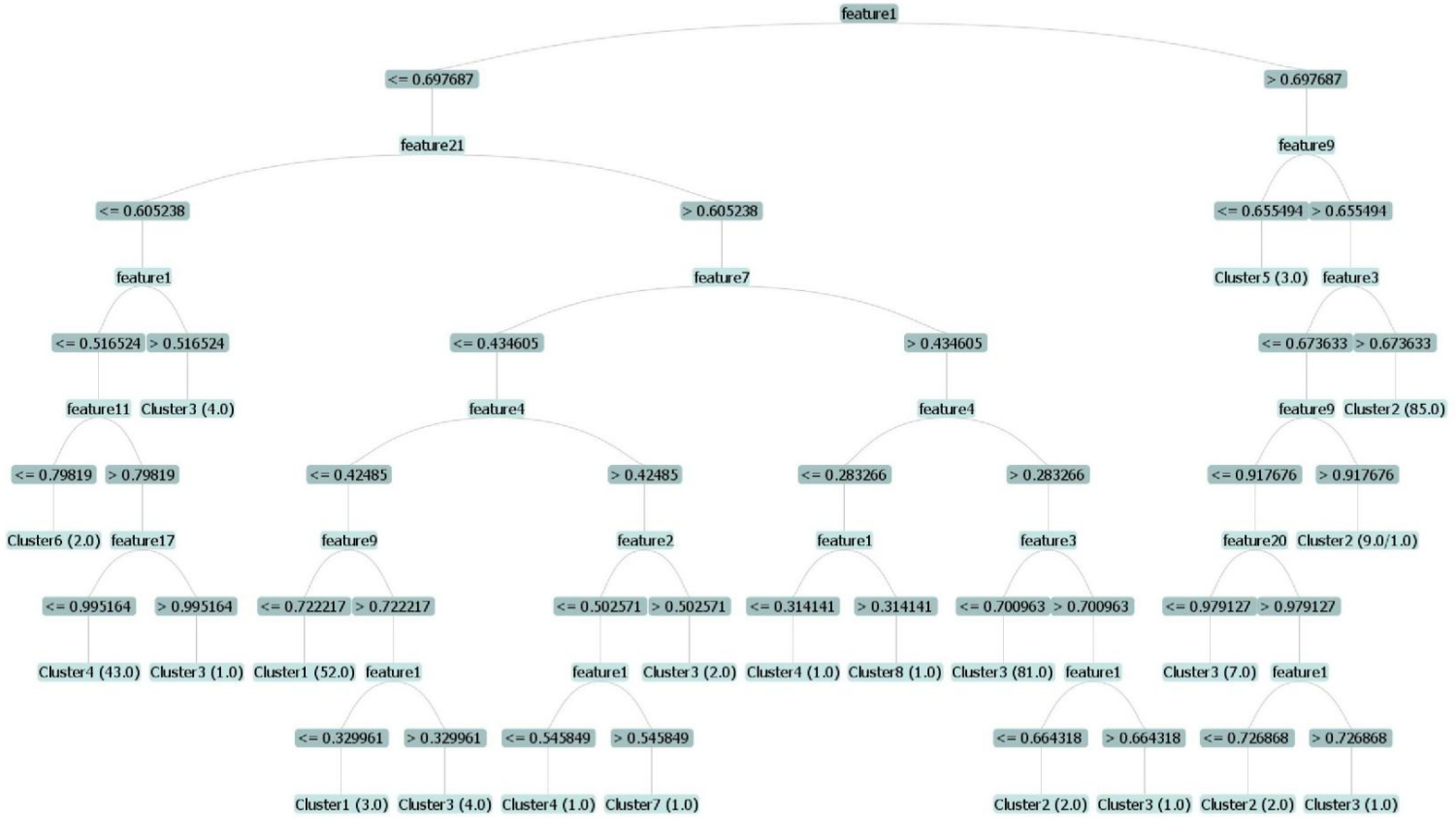

Figure 7.3 Decision tree for classifying winter weekdays consumption using clustering of Follow-the-Leader algorithm with actual features.

As the tree in (Fig. 7.3) suggests, the consumption value at the end of the first hour resides in the root position to characterize few members of the large clusters, and one of the large clusters at the right branch while the majority of the leaves are located at the left branch. Consumption amounts between nine in the evening until nine in the morning could be considered to differ among data points at greater amounts than the rest of the features, as the labels of non-leaf nodes of the tree are interpreted. This concurs with the clustering illustrated in (Fig. 7.1), in which most of the data points indicate steadily high consumptions during late morning, lunch and afternoon hours, yet consumption amounts differ among data points during the rest of the day.

Accuracy of the classification is assessed by cross-fold validation, in which data set is split into folds of equal sizes and one fold is utilized as test instances while the others constitute the training set. 10-fold stratified cross validation is adopted, in which each fold corresponds to 10% of the whole data set. Each fold is selected in a way such that they have more or less the same distribution of the labeled data as the rest of the data set, which names the validation technique as stratified. Evaluation results are presented as averages obtained in each fold of the cross validation.

Clusters having less than 10 elements cannot be evenly distributed among folds, which is a particular reason of error in classification. Although stratified validation stresses on even distributions of differently-labeled data, in particular folds outlier clusters may not reside and hence their instances are necessarily misclassified. Confusion matrix clarifies the number of incorrect classifications of instances label by label. Considering the case of cross validation of the decision tree in (Fig 7.1) of winter weekdays data set classification by employing load shape indices, the following confusion matrix is generated by the evaluation.

Table 7.1 Confusion matrix for classification of winter weekdays data clustered by Follow-the-Leader using load shape indices.

a	b	c	d	e	f	g	h	Classified as
46	0	3	6	0	0	0	0	a = Cluster1
0	92	5	0	0	0	0	0	b = Cluster2
3	7	89	2	1	0	0	0	c = Cluster3
7	0	3	35	0	0	0	0	d = Cluster4
0	0	1	0	2	0	0	0	e = Cluster5
0	0	0	1	0	1	0	0	f = Cluster6
0	0	0	0	1	0	0	0	g = Cluster7
0	0	0	1	0	0	0	0	h = Cluster8

Singleton outlier clusters, *Cluster* 7 and *Cluster 8* in (Fig. 7.1), are incorrectly classified, as Table 7.1 indicates. Other outlier clusters, *Cluster 5* and *Cluster6* also include classification errors. Although the tree in (Fig. 7.2) constructed by all instances do not incorrectly classify instances of the outlier clusters, cross validation methodology does. In

order to discuss further the nature of misclassifications due to clustering errors and comment about inter-cluster similarity, confusion matrix for the classification of the same clustering by using all data features as attributes is included subsequently.

Table 7.2 Confusion matrix for classification of winter weekdays data clustered by Follow-the-Leader using actual features.

a	b	c	d	e	f	g	h	Classified as
44	0	8	2	0	1	0	0	a = Cluster1
0	88	9	0	0	0	0	0	b = Cluster2
1	9	87	4	0	0	1	0	c = Cluster3
2	0	2	40	0	1	0	0	d = Cluster4
0	2	1	0	0	0	0	0	e = Cluster5
1	0	0	0	0	1	0	0	f = Cluster6
0	0	1	0	0	0	0	0	g = Cluster7
0	0	1	0	0	0	0	0	h = Cluster8

Confusion matrix of classification with actual features of the data points in Table 7.2 reveals particular clues about the similarity of large clusters *Cluster 1*, *Cluster 3*, and *Cluster 4*. Although most instances labeled with the names of these clusters are correctly classified, particular instances have been mistakenly labeled as one of the other large clusters. Another large cluster, *Cluster 2*, whose elements have high hourly consumption values throughout the day, is incorrectly classified only with *Cluster 3*, which has higher consumptions during nighttime and early morning hours than *Cluster 1* and *Cluster 4* and otherwise as high values as *Cluster 2* in the remaining hours of the day.

Outlier clusters are poorly classified, as the matrix indicates that only one instance of *Cluster 6* is correctly classified in successive folds during the course of evaluation. In fact, a quick comparison of the two confusion matrices reveals that classification utilizing shape indices outperforms the one using actual features on the clustering of winter weekdays by Follow-the-Leader algorithm.

In order to evaluate the quality of the classification, several performance metrics existing in the literature are employed by the WEKA framework. These metrics are defined briefly in the following.

Kappa statistics measures the degree to which the classification has correctly computed the labels of the instances by discarding the chance factor.

$$\kappa = \frac{P(a)-P(e)}{1-P(e)} \tag{7.4}$$

In the formula in (7.4), $P(a)$ corresponds to the percentage of correctly classified instances by the classifier and $P(e)$ represents the hypothetical chance of agreement,

which is computed as the product of probabilities of every possible classification covered in the confusion matrix. In other words, Kappa statistics compares the accuracy of the classifier with the random classifier to measure the improvement obtained by adopting the classifier and is more sensitive than the directly measuring the percentage of correctly classified data points since, it filters out the agreement achieved by chance.

Root mean squared error (RMSE) is the square root of the mean of the squared difference between the prediction probability, which is either 0 or 1 implying whether an instance is correctly classified; and prior class probability computed by calculating the frequencies of classes. It measures accuracy. Lower RMSE values indicate better performance.

True Positive (TP) Rate regarding a class is the percentage of correctly classified instances with the class label. For example, *Cluster 1* in the confusion matrix in Table 7.2 has its TP rate as 44 / (44+8+2+1) = 0.8. TP rate is equivalent to *recall*.

False Positive (FP) Rate regarding a class with label *c* is the proportion of instances incorrectly classified as *c* but originally belonging to another class over all instances other than *c*. For example, the FP rate for *Cluster1* in the confusion matrix in Table 7.2 is (1+2+1) / (251) = 0.1594.

Precision of a class having label *c* is the proportion of correctly classified instances with label *c* over all instances with label *c*. For example, precision of *Cluster 1* in the confusion matrix in Table 7.2 is 44 / (44+1+2+1) = 0.917.

Receiver operating characteristics (ROC curve) of a class with label *c* is the graph depicting the false positive rate on the horizontal axis and the true positive rate on the vertical axis by manipulating an operating threshold parameter, which is a probability threshold that deals with the proportion of instances classified as *c* in the data, while keeping the model fixed. In other words, operating threshold controls the probability of particular predictions via the probability assigned by the classifier to *c*. For further details, please refer to (Davis and Goadrich, 2006). ROC curve of *Cluster1* in the confusion matrix in Table 7.2 is as follows.

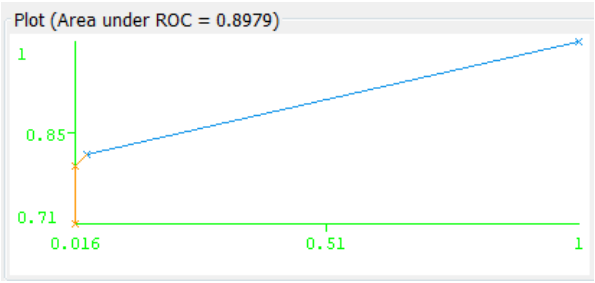

Figure 7.4 ROC curve for *Cluster1* residing in Table 7.2.

The color transition of the curve in (Fig. 7.4) displays the change in the threshold value such that brown values imply that the threshold is close to 1, while blue values imply that the threshold is closer to 0. Area under ROC curve is utilized as an accuracy measure whose higher values are appreciated. In case of *Cluster1*, cross fold validation yields a value of 0.8979. The upper limit of the ROC area is 1.

Table 7.3 Behavior of *Cluster1* of Table 7.2 with changing threshold as marked on ROC curve.

Threshold	TP Rate	FP Rate	TP	FN	FP	TN	Precision
1.0	0.71	0.016	39	16	4	247	0.907
0.98	0.8	0.016	44	11	4	247	0.917
0.024	0.82	0.04	45	10	10	241	0.82
0.0	1.0	1.0	55	0	251	0	0.18

As the threshold goes from 1 to 0, number of true positives and false positives increase. On the other hand, number of false negatives, namely the misclassifications of *Cluster1* as other classes, and number of true negatives, correctly classified instances as not belonging to *Cluster1* diminish as operating threshold decreases. Thresholds 1 and 0 can be interpreted as the behavior of *Cluster1* at worst and at best according to the TP rate. However, the reverse situation will be valid, if evaluation is done based on the FP rate.

ROC curve in (Fig. 7.4) demonstrates trade-offs by manipulating the test data, as values in Table 7.3 indicate. The highest TP rate with respect to the lowest FP rate constitutes the main trade-off, which results in the threshold corresponding to the highest precision being included in the final evaluation of the classifier, which is 0.98 in this case.

Weighted averages of TP rate, FP rate, precision and area under ROC curve of each class label are computed to evaluate the classifier.

Classification of all illustrated clusterings of winter and summer weekdays data in all frameworks but the weighted Euclidean that are covered in Chapter 5 are evaluated in the subsequent table.

Table 7.4 Evaluation of classifications performed on the illustrated clusters covered in Chapter 5.

			Kappa Statistics	RMSE	TP Rate (Recall)	FP Rate	Precision	ROC Area
WFA-K-means	Winter (hp/9)	indices	0.83	0.1632	0.876	0.04	0.87	0.925
		features	0.8261	0.168	0.873	0.046	0.871	0.918
	Summer (e/6)	indices	0.8468	0.1829	0.896	0.032	0.898	0.936
		features	0.8882	0.1585	0.925	0.032	0.921	0.948
K-medoids	Winter (hp/8)	indices	0.8115	0.1791	0.869	0.061	0.867	0.916
		features	0.8642	0.1535	0.905	0.044	0.897	0.933
	Summer (hp/11)	indices	0.7716	0.1701	0.84	0.048	0.833	0.894
		features	0.7708	0.1706	0.84	0.06	0.837	0.889
Similarity-based K-means	Winter (e/13)	indices	0.718	0.1847	0.775	0.053	0.76	0.865
		features	0.8494	0.1363	0.879	0.027	0.866	0.922
	Summer (e/14)	indices	0.7556	0.1678	0.802	0.033	0.789	0.879
		features	0.7762	0.1612	0.818	0.033	0.818	0.895
Fuzzy K-means	Winter (se/6)	indices	0.7608	0.238	0.827	0.06	0.829	0.892
		features	0.8363	0.1976	0.882	0.053	0.881	0.92
	Summer (hp/14)	indices	0.7587	0.1682	0.802	0.026	0.801	0.886
		features	0.8349	0.139	0.865	0.024	0.86	0.921
Follow-The-Leader	Winter (e/8)	indices	0.8171	0.1797	0.866	0.043	0.863	0.922
		features	0.7936	0.1922	0.85	0.059	0.843	0.901
	Summer (hp/14)	indices	0.7743	0.1607	0.818	0.031	0.807	0.892
		features	0.7939	0.1543	0.833	0.034	0.824	0.9
Hierarchical (UPGMA)	Winter (e/9)	indices	0.775	0.1615	0.876	0.068	0.869	0.914
		features	0.8592	0.1319	0.922	0.052	0.909	0.921
	Summer (e/10)	indices	0.7368	0.1734	0.846	0.069	0.833	0.896
		features	0.8402	0.1373	0.906	0.045	0.898	0.925
SOM (batch 6 15)	Winter (e/6)	indices	0.7866	0.2148	0.859	0.064	0.853	0.893
		features	0.8662	0.1709	0.912	0.045	0.91	0.933
	Summer (hp/8)	indices	0.7766	0.2063	0.827	0.035	0.828	0.896
		features	0.8335	0.1793	0.871	0.03	0.874	0.927

Winter and summer weekdays data sets are represented in the Table 7.4 with *Winter* and *Summer* keywords. Euclidean, hybrid Pearson, and squared Euclidean distance frameworks are abbreviated as *e*, *hp*, and *se*, followed by number of clusters generated in the framework on the data set. Whether a classification is done by using load shape indices or actual data features are also indicated in the table.

As discussed in Chapter 5, WFA-K-means, K-medoids, and Similarity-Based K-means are K-means family members that achieve high quality clustering; UPGMA linkage is chosen to represent Hierarchical clustering family, and batch training is selected to represent SOM clustering. Fuzzy K-means and Follow-the-Leader are also included. Only, ISODATA clustering is excluded from classification because of inadequacy of the evaluation methodology regarding this algorithm.

A good classification manages to get high values in Kappa statistics, TP rate, precision, and area under ROC curve and low values in root mean squared error and FP rate.

Using actual features results in more adequate classifications than employing shape indices for most of the cases in Table 7.4. Nonetheless, shape indices are able to explain nature of classifications well enough and even in some cases their utilization is evaluated to surpass that of data features, such as the classification of Follow-the-Leader clustering exemplified previously.

Regarding the clustering operations listed in Table 7.4, the one carried out by WFA-K-means appears to yield the most promising classification according to most of the evaluation metrics. However, since the inspected groupings of data points are for distinct number of clusters, such a comparison may come short in explaining every instant of the behavior of clustering algorithms. Nonetheless, it would not be wise not to take into consideration this comparison.

Decision tree algorithm and the cross validation technique tend to produce more classification errors when number of clusters increase to include outlier groups with few members. As discussed in case of Follow-the-Leader clustering on winter weekdays data, it is not always possible to distribute evenly the instances of a label among the folds when the label corresponds to outlier clusters. Moreover, during induction and post-pruning phases, the splits necessary to generate conditions of classification of an outlier clusters may have not been performed, due to not yielding substantial information gain or coming short in assuring the confidence interval. However, trees generated on all instances correctly classify even the singleton outlier clusters.

Every reported classification in Table 7.4 passes the adequacy evaluation with at least substantial success if not outstanding. This verifies the importance of clustering step in developing applications based on load profiles of electricity consumers.

7.3 Basic Class-Based Dynamic Tariff Design

One of the applications that could be devised as a direct consequence of the load profiles clustering is designing tariffs that are specific to each customer class. In this chapter, utilization of features of cluster centers in the formation of specific tariffs and the conditions of obtaining at least the same revenues as fixed tariffs that apply to each customer of the company will be discussed. Detailed economic background and implications, as well as constituents of an elaborated business strategy are beyond the scope of the thesis.

The process flow regarding tariff design and proposal is included in the following.

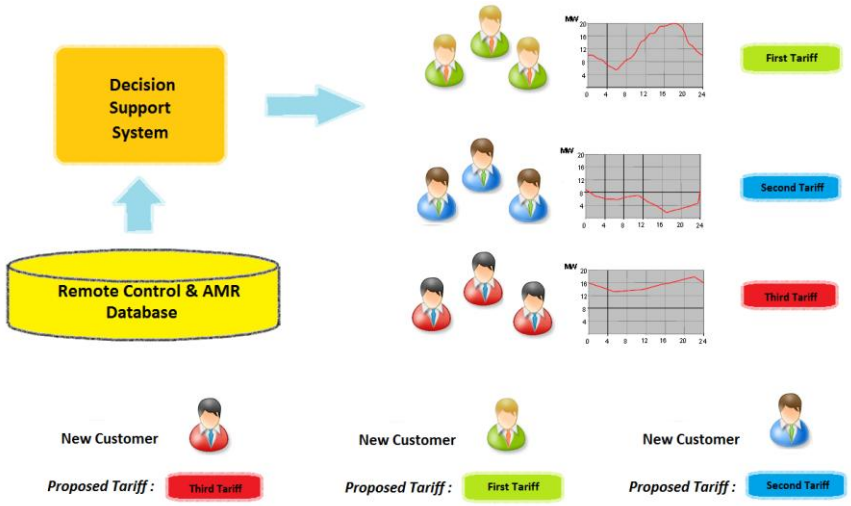

Figure 7.5 Process flows of tariff design and appropriate tariff proposal to new customers.

Tariff design is based on clustering of customers as described in (Fig 7.5). Customer recognition outlines the conditions of class membership for new customers. As these customers are assigned to the most appropriate class, corresponding tariffs are proposed automatically.

Features of cluster centers can be used as direct weight factors of a base unit price that affects the amount a member customer has to pay at a certain hour of day for their consumption of electricity. A consumer has to pay more for the peak hours of the day but much less than the fixed price for the rest of the day. Follow-the-Leader clustering of winter weekdays data reveals the following tariffs per cluster.

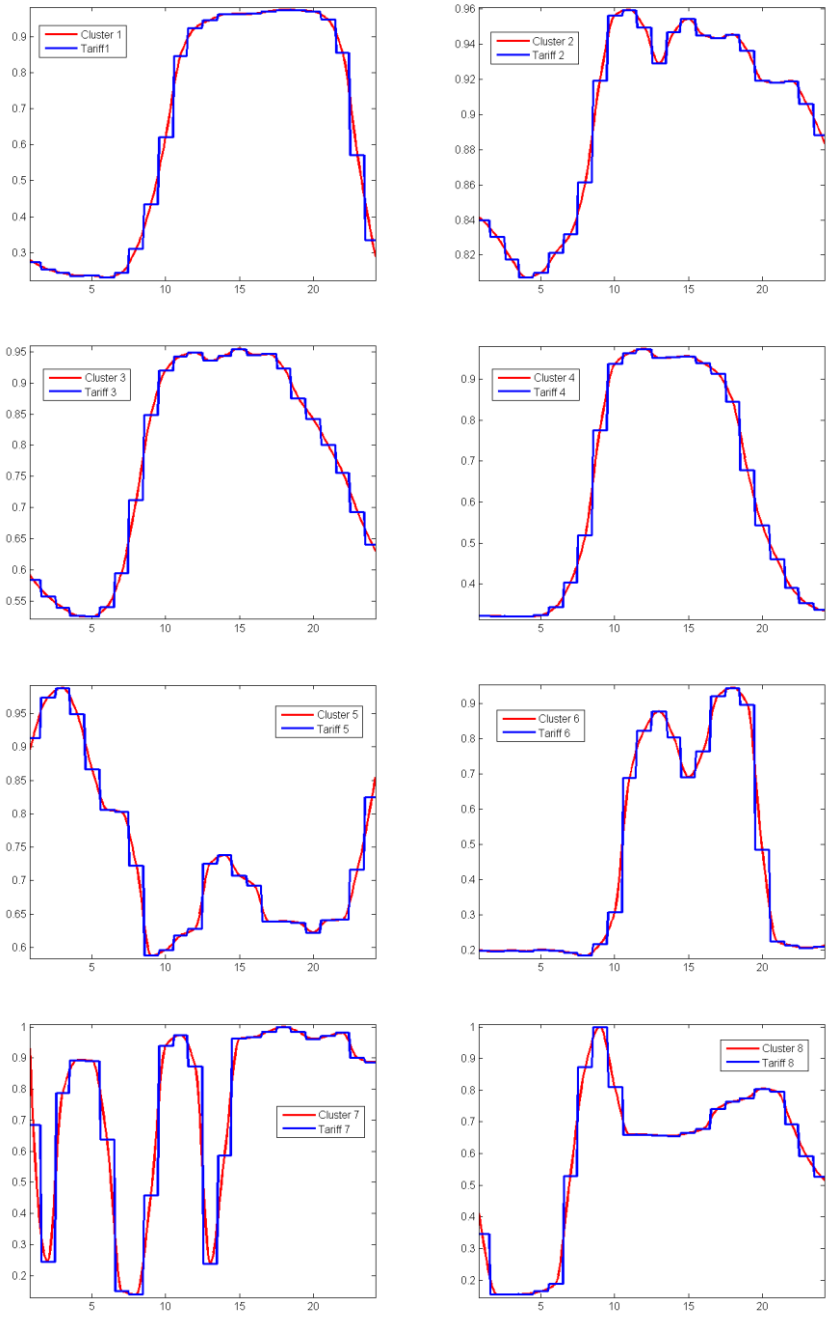

Figure 7.6 Tariffs proposed to customers for electricity consumption in winter weekdays and generated by Follow-the-Leader clustering.

Using cluster center features as weight factors of cluster-specific tariffs is exemplified for Follow-the-Leader clustering in (Fig. 7.6). Revenue computation is carried out as follows.

$$V_j = \sum_{X_i \in C_j} \sum_{q=1}^{Q} X_i^d \, T_j^d \, \tau_c \quad , \tag{7.5}$$

where T_j is the tariff generated for C_j and
τ_c is a fixed unit price in some currency and V_j is cluster revenue.

Revenue obtained from a particular tariff is computed as in (7.5). Base costs, taxes, etc are not covered by the equation. Since true consumption levels of all customers are close to each other, normalized features of data points together with features of the cluster centers is used in the equation.

Next, total revenue is calculated by summing all tariff revenues as in the following.

$$TV_c = \sum_{j=1}^{K} V_j \quad , \textit{where } TV_c \textit{ is the total revenue based on clustering.} \tag{7.6}$$

As all customers are high-power consumers, currently an unweighted fixed price is applied to everybody regardless of any clustering. Following the computation in equation (7.6), current employed revenue collection is assumed to be carried out as follows.

$$TV_f = \sum_{i=1}^{N} \sum_{q=1}^{Q} X_i^d \, \tau_f \quad , \tag{7.7}$$

where TV_f is the total fixed revenue, τ_f is a hypothetical fixed unit price applying to all customers regardless of natural clusters.

Ratio of the two total revenues gives clues about developing a simple business strategy. If a company hopes to collect as much revenue as before by adopting tariffs formed by the clustering operation, then the new fixed unit price which ia multiplied by tariff features to determine hourly price shall be proportional to the current fixed unit price, as covered in the following equation.

$$\textit{Let } TV_c = TV_f. \quad \textit{Then,} \tag{7.8}$$

$$\frac{\tau_c}{\tau_f} = \frac{\sum_{i=1}^{N} \sum_{q=1}^{Q} X_i^d}{\sum_{j=1}^{K} \sum_{X_i \in C_j} \sum_{q=1}^{Q} T_j^d \, X_i^d}$$

The clustering whose tariffs are illustrated in (Fig. 7.6) gives the ratio formulated in (7.8) as 1.20 which shall be further multiplied by the centroid features generated by Follow-the-Keader clustering. As the graph of *Tariff 1* can tell, members of *Cluster 1* should pay as much as less-than-1.20 times they currently pay for their peak demands, while should pay only less-than-0.36 times they currently pay for hours at which they consume the least amount of energy. Average, minimum and maximum amount they should pay for electricity consumption in winter weekdays is included in the following table.

Table 7.5 Statistics regarding the ratio of new hourly prices to currently-utilized fixed unit price for Follow-the-Leader clustering.

Follow-the-Leader (winter weekdays)	Size	Minimum	Maximum	Average	Deviation
Cluster 1	55	0.28	1.17	0.77	0.38
Cluster 2	97	0.97	1.15	1.08	0.06
Cluster 3	103	0.63	1.15	0.93	0.2
Cluster 4	45	0.38	1.17	0.74	0.33
Cluster 5	3	0.71	1.19	0.88	0.15
Cluster 6	2	0.22	1.13	0.54	0.36
Cluster 7	1	0.17	1.2	0.91	0.34
Cluster 8	1	0.19	1.2	0.7	0.3
Weighted Average		0.64	1.16	0.91	0.21

Results in Table 7.5 demonstrates that for most of the clusters average hourly prices are considerably less than the fixed price when it is taken as the unity. At most 30% and at least 7% decrease in average hourly prices has occurred concerning all clusters except *Cluster 2*, whose data points correspond to generally high consumption values throughout the day. For customers categorized to be in *Cluster 2*, the prices have risen by 8%. That rise concerns less than one third of all customers. Weighting the computed statistics with cluster sizes, a 9% decrease in overall average hourly prices is obtained.

Inter-cluster statistics for several clustering operations carried out by distinct algorithms illustrated in Chapter 5 are enlisted in the following table for both winter and summer weekdays data sets.

Table 7.6 Weighted average statistics regarding the ratio of new hourly prices to currently-utilized fixed unit price of all clusters by several algorithms.

		Minimum	Maximum	Average	Max Avg	Deviation	Ratio
WFA-K-means	Winter (hp | 9)	0.628	1.168	0.916	1.08	0.214	1.211
	Summer (e | 6)	0.649	1.158	0.917	1.06	0.2	1.205
K-medoids / PAM	Winter (hp | 8)	0.64	1.191	0.917	1.118	0.212	1.191
	Summer (hp | 11)	0.646	1.172	0.916	1.09	0.202	1.172
Similarity-based K-means	Winter (e | 13)	0.631	1.159	0.913	1.114	0.211	1.198
	Summer (e | 14)	0.647	1.152	0.913	1.105	0.2	1.18
Fuzzy K-means	Winter (se | 6)	0.641	1.156	0.917	1.104	0.208	1.197
	Summer (hp | 14)	0.649	1.15	0.913	1.107	0.2	1.173
Follow The Leader	Winter (e | 8)	0.635	1.16	0.914	1.078	0.209	1.201
	Summer (hp | 14)	0.647	1.151	0.913	1.103	0.201	1.18
Hierarchical (UPGMA)	Winter (e | 9)	0.639	1.164	0.922	1.018	0.21	1.21
	Summer (e | 10)	0.658	1.155	0.923	1.034	0.197	1.191
SOM (batch 6 15)	Winter (e | 6)	0.637	1.162	0.92	1.05	0.207	1.207
	Summer (hp | 8)	0.655	1.15	0.92	1.097	0.196	1.183

Attributes in Table 7.6 are weighted average statistics such as the ones computed at the last row of the table in order to clarify the average case regarding clusters generated by the algorithms enlisted.

All clustering results for the same data sets covered in Table 7.6 yield more or less the same average hourly prices no matter how different the ratios are, since total revenues are assumed the same. Hierarchical clustering offers the highest average price, yet has the lowest maximum average price applied to a particular cluster generated by the algorithm, which could be regarded as a positive factor in the first adoption of dynamic tariffs to ensure trust in customers. Statistics presented in the table offers many possible trade-offs to be used as guidelines in selecting a particular consumption data clustering.

Tariffs reflecting particular business strategies can also be designed based on clustering of customers. One particular strategy aims at reducing the peak of load profiles and making the consumption curves as flat as possible. For this purpose, price of electricity service at peak hours shall be higher than the rest, while the rest shall be priced as constantly as possible.

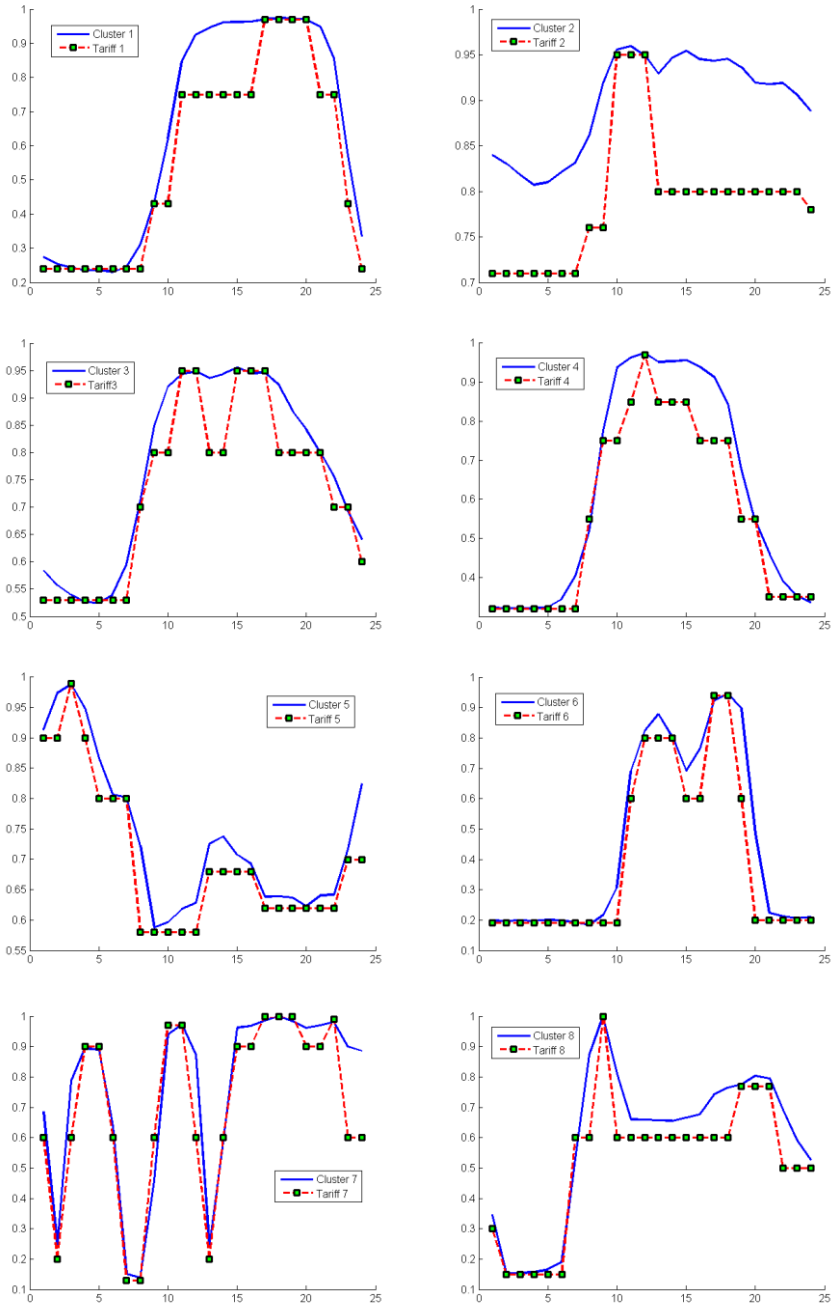

Figure 7.7 Tariffs aiming at reducing peak load and stabilizing consumption based on Follow-the-Leader clustering of winter weekdays data.

Ratio between unit prices of several tariffs based on natural clusters to one global tariff applying to all customers for tariffs illustrated in (Fig. 7.7) is calculated as 1.33, which is higher than the ratio computed previously for tariffs designed by directly employing cluster center features, because tariff constituents displayed in (Fig. 7.7) are generally of less values than centroid features. Since all features of *Cluster 2* is greater than 0.8,

namely all hourly consumption values are very high, the gap between the centroid and the tariff is increased on purpose so that pricing of the peak hours demand can be more significant.

Table 7.7 Statistics regarding the ratio of new hourly prices to currently-utilized fixed unit price for Follow-the-Leader clustering.

Follow-the-Leader (winter weekdays)	Size	Minimum	Maximum	Average	Deviation
Cluster 1	55	0.32	1.3	0.74	0.38
Cluster 2	97	0.95	1.26	1.05	0.1
Cluster 3	103	0.71	1.26	0.97	0.21
Cluster 4	45	0.43	1.29	0.74	0.31
Cluster 5	3	0.77	1.32	0.93	0.16
Cluster 6	2	0.25	1.25	0.53	0.37
Cluster 7	1	0.17	1.33	0.93	0.38
Cluster 8	1	0.2	1.33	0.69	0.3
Weighted Average		0.67	1.28	0.92	0.22

As the values in Table 7.7 indicate, there is a moderate increase in minimum and maximum hourly prices with respect to centroid-feature utilization analyzed in Table 7.6. However, overall average hourly price is still lower than the unity, which is assumed to be equal to the unique fixed price currently in use. Moreover, average hourly prices of *Cluster 1* and *Cluster 2* whose number of data points are almost half of the data set, have decreased.

Relatively high hourly prices regarding *Cluster 2* can be resolved by directly decreasing prices for this cluster, while increasing the hourly prices of other clusters to compensate for the change. Moreover, if the company aims at eliminating outlier clusters, values of tariff features regarding these clusters can be substantially increased, which will be reflected as a slight decrease of prices in remaining large clusters so that the total revenues is maintained unchanged. As a matter of fact, the company may decide to accept even lower revenues at the beginning of the transition to dynamic tariffs to result in lower unit prices regarding all clusters, expecting to attract more customers and more revenues at longer terms.

Since the second tariff design intends to achieve a particular business strategy, continuous monitoring of customers is essential in order to evaluate behavioral changes for deciding on re-clustering and renewing tariffs, since a dynamic tariff, instead of the static one, is more likely to change consumption shapes and amounts.

In other words, dynamic tariffs like the ones illustrated in (Fig. 7.7) do not necessarily result in cheaper tariffs for all hours of the day, yet they intend to manipulate consumption of customers to realize particular economical and/or technical goals. Cluster-based tariffs tend to decrease the economic burdens at the customer side as well by reducing average prices and by giving them the opportunity to change their consumption behaviors accordingly such as consuming less electricity at hours that are charged with higher prices.

More factors exits in setting up a business strategy, which are not covered merely by the anonymous consumption data of certain dates as used in this study. Among these factors, there are parameters risen from the history of the customer with the company, goals of the company to be covered in particular deadlines, labor power that could be assigned to clustering applications, the scale to which automatic meter reading devices are utilized, the amount of data that has already been collected and is planned to be collected, etc. Real life matters are essential to elaborate on any business strategy that could be developed on top of clusters of consumption data.

7.4 Other Applications

Load forecasting is another example of applications of load profiling (Zhang et al., 2013). The process flow of load forecasting is included in the following.

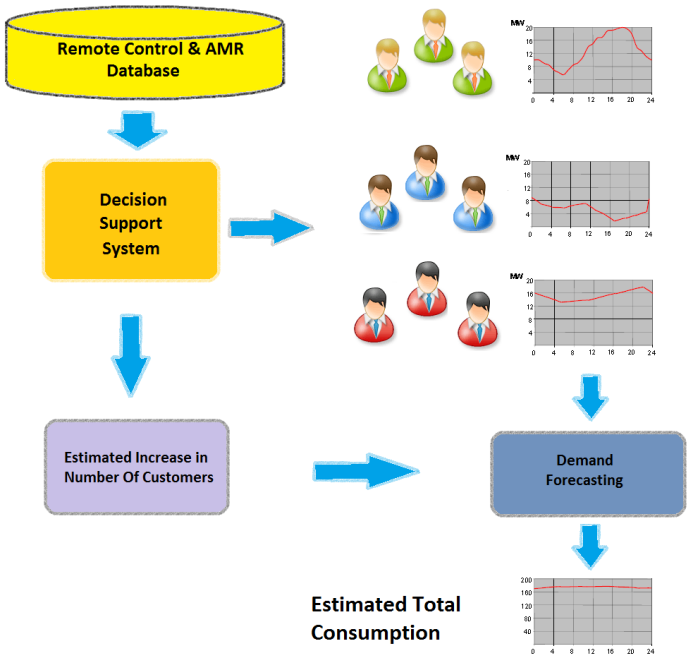

Figure 7.8 Process flow of load forecasting.

Estimating future consumption values, whose process flow is illustrated in (Fig. 7.8), requires building an accurate model of seasonal and yearly consumptions based on load profiles generated by clustering algorithms. Corresponding model should inspect consumption data of several years together with change in external factors such as the climate in this period. Load forecasting also requires statistics regarding number of customers a company has within several years so that an estimation regarding future customer sizes is made. Since the data used in this study does not include the necessary information for deploying an application of load forecasting, please refer to (Labeeuw and Deconinck, 2013) for further details.

Another application based on load profile clustering is fraud detection. In order to detect anomalous use of electricity for possible illegal purposes, continuous monitoring of customers through AMR devices is essential so that deviations from representative load profiles could be detected.

The process flow of fraud detection is illustrated in the following.

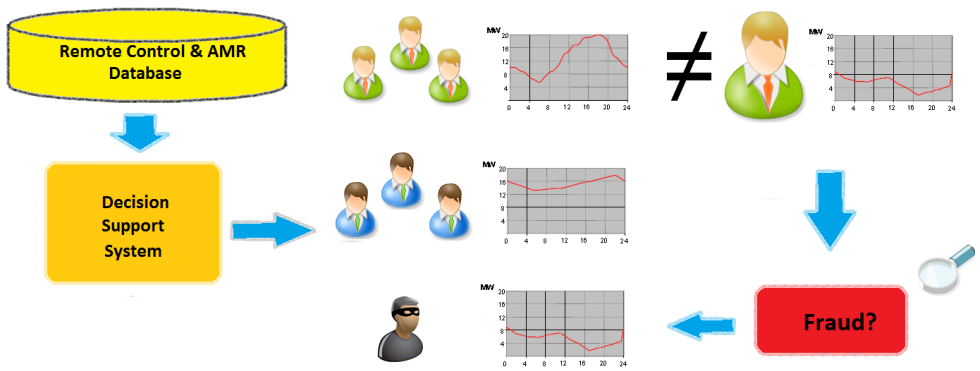

Figure 7.9 Process flow of fraud detection.

As flow in (Fig. 7.9) indicates, deviation of updated load diagram of a customer from its original cluster is a sign for suspecting fraud. The amount of deviation that could be judged as fraud should be large enough whose actual value depends on the choice of the designer.

Fraud detection should be put into use following the computation of representative load profiles and assuring continuous data flow from AMR devices so that continuous monitoring of customers will be possible. Since, this study does not have the obligatory resources for deploying a fraud detection application, please refer to (Nizar et al., 2006) for further details.

7.5 Summary

Clustering of seasonal load diagrams of customers of an electricity distribution company provides the basis for deploying applications that offer various solutions to the problems in power domain.

Throughout the chapter, customer recognition involving further classification of clustered customers, which aims at discovering the conditions of membership of clusters and correctly placing new customers into classes; and designing class-specific tariffs, which constitutes the basis of devising business strategies with pre-defined goals is analyzed.

Moreover, mechanisms of load forecasting and fraud detection applications are briefly described. This study focuses on computational infrastructure for deploying such applications and further elaboration is mandatory for both the design and evaluation of applications, which can be attained by collecting more comprehensive data of consumption over several years and discussing the crucial parts of possible business strategies with domain experts.

CHAPTER 8

DISCUSSION AND CONCLUSION

Clustering and classification of electricity consumption data has emerged as a recent research topic. Traditional data mining and machine learning methods are used extensively in the problem domain together with particular modifications and hybridization.

Electricity consumption data of numerous customers that are collected at hourly intervals by AMR devices over long periods are stored in centralized databases of electricity distribution companies. These vast amounts of data can be used to generate crucial information that provides the basis for developing business strategies and planning electricity usage through processing with suitable techniques.

In this study, a prototype framework that is capable of preparing and clustering data, evaluating clusters, classifying cluster-representing load profiles and deploying related applications is proposed.

Data preparation step focuses on forming seasonal customer-representative load diagrams that summarize hourly consumption of the customer on a typical day in the corresponding season. Along with seasons, days of the week and holidays are used as data divisors, because they are among the loading conditions that effect consumption amounts and shapes.

Data is prepared for further processing by filling missing feature values, detecting outliers and smoothing. Normalizing diagram features is necessary so that consumption shapes are the discriminating characteristics to be used in clustering. An alternative to normalization is using discriminative load shape indices such as measured impact of lunchtime over daily average consumption or night impact, which assess consumption behaviors at particular periods during a typical day

Various methods are described to cluster representative load diagrams. Included algorithms are analyzed within six families composing of K-means family, Hierarchical family, Fuzzy clustering, ISODATA, Follow-the-Leader clustering and SOM clustering.

Families of algorithms have distinct mechanisms that enable them to cluster data sets. K-means family algorithms require exact number of clusters, which are initialized and then refined iteratively in successive cycles. Fuzzy clustering imposes a soft clustering of data

points via computing a fuzzy degree-of-belonging matrix. ISODATA clustering does not require number of clusters to function, yet it uses successive merges or splits depending on its several parameters until some desired number of clusters are generated.

Follow-the-Leader clustering does not require exact information of number of clusters nor does it perform cluster initialization. The algorithm traverses data points iteratively and forms clusters depending on an operating distance threshold. Hierarchical clustering arranges data points into singleton clusters and iteratively merges clusters according to a distance function until cutoff is reached. Finally, SOM clustering projects data points into a 2-D lattice having fewer nodes than the size of the data set and then calls another clustering algorithm to partition the map.

Three distance measures are used as heuristics that guide clustering algorithms: Euclidean distance metric, weighted Euclidean distance and hybrid Pearson distance. In particular algorithms, the first two measures are used in squared forms. Hybrid Pearson distance takes into consideration Euclidean distance along with correlation so that shape of consumption contributes to computed distance. In most of the cases, hybrid Pearson measure performs as well as Euclidean metric and in particular cases the hybrid measure outperforms the latter. On the other hand, performance of weighted Euclidean distance tends to be inferior compared to other measures. A future study regarding distance heuristics shall include learning weights to be used distance computations, since pre-defined weights have been shown not to necessarily increase performance quality.

Point symmetry distance heuristic is also covered in the study as an alternative to the described distance measures in Similarity-Based K-means clustering algorithm, showing that genuine measures of distance and similarity can be devised by using mathematical constructs that are not extensively employed in data mining.

Generated clusters are evaluated for quality of performance by validity indices. More compact and well-separated clusters are considered to have higher quality. Evaluation of algorithms through their products has revealed that superiority of an algorithm over another cannot be proven, yet general tendencies can be discussed. Distinct methods yield their best performance at differing number of clusters, which may be assessed changeably depending on the goals of performed load profile classification. Therefore, testing numerous methods over a data set is essential in discovering the clustering that fits best to both objective and subjective purposes.

Running time complexity of algorithms is a severe issue that needs attention. If the proposed framework is intended to be applied on massive data sets, used algorithms should convergence in reasonable time just like they should produce high-quality clustering. All clustering methods are evaluated for elapsed running times and scalability tendencies.

Parallel versions of algorithms are studied as a way of decreasing running times in the example of K-means clustering. Hybrid Hierarchical–K-means and Hierarchical–Follow-the-Leader algorithms are designed as scalable alternatives to the original Hierarchical methods that are just as accurate and attain substantially less running times on larger data sets. Effect of reducing data size as in case of SOM clustering is also discussed as means of reducing total running times, yet at the expense of reduced accuracy.

The effect of number of iterations until convergence on the scalability to larger data is not included within the scope of this study. Algorithmic improvements such as using single-pass algorithms in massive data sets are necessary to obtain absolutely scalable clustering.

Classification of load diagrams composed of hourly features or load shape indices by using cluster names as labels puts forward the adequacy of shape indices in explaining consumption styles. For most cases, adequacy evaluation of load shape indices has yielded closer results to that of actual load features.

Classification of load diagrams paves the way for customer recognition that makes it possible to correctly classify new customers and customer-specific tariff design that offers the possibility of designing various business strategies that have consequences on both the company-side and the customer side. Foundations for load forecasting and fraud detection are also discussed among deployable applications that make use of load profiling.

Considering future studies, several improvements can be realized in each sub-step covered. More comprehensive data preparation requires more sophisticated treatment of missing attributes and detecting outliers. Moreover, modeling external conditions such as temperature throughout the year is obligatory to filter out the climate effect from consumption data in order to arrive at annual frameworks that come handy in applications such as forecasting revenues and consumption amounts.

Numerous clustering algorithms in the literature can be tested on the proposed framework. Improvements over the performance of clustering algorithms can be attained through designing hybrid algorithms and employing other distance metrics.

Access to data having vast amounts of instances and collected over several years is the most crucial ingredient for future studies. Processing large amounts of data will give a more concrete idea about quality of performance and scalability of algorithms. Proposed modifications to popular clustering algorithms and designed hybrid methods require to be tested on new consumption data sets.

Load forecasting and fraud detection applications demand data collected over several years so that continuous monitoring can be emulated by building up and testing models over seasonal profiles of distinct years. In addition to this, all applications based on load profiling necessitates collaboration among specialist from several domains including

national energy policy, macroeconomics and sensor technology so that the end product could actually take place in the market.

Framework proposed in this study offers the generality so that it can be applied to similar problem domains such as telecommunications and natural gas as long as corresponding data is measured within a pre-defined time interval. Although particular modifications are obligatory, cross-appliance of the framework shall make it possible to discuss characteristics of problem domains. Furthermore, exchange of ideas regarding classification and application deployment may result in collective enhancements.

As a summary, this study has shown the potentials of load profiles classification and advocates the use of such a framework in electricity domain in order to develop sophisticated solutions to encountered problems.

REFERENCES

Anderson, B. J., Gross, D. S., Musicant, D. R., Ritz, A. M., Smith, T. G., & Steinberg, L. E. (2006). Adapting k-medians to generate normalized cluster centers. In *Proceedings of the Sixth SIAM International Conference on Data Mining* (Vol. 124, p. 165). Society for Industrial and Applied Mathematics (SIAM).

Arthur, D., & Vassilvitskii, S. (2007, January). K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., & Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, *5*(7), 622-633.

Ball, G. H., & Hall, D. J. (1965). *ISODATA, a novel method of data analysis and pattern classification*. STANFORD RESEARCH INST MENLO PARK CA.

Bandyopadhyay, S., & Saha, S. (2007). GAPS: A clustering method using a new point symmetry-based distance measure. *Pattern Recognition*, *40*(12), 3430-3451.

Cabral, J. E., & Gontijo, E. M. (2004, October). Fraud detection in electrical energy consumers using rough sets. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on* (Vol. 4, pp. 3625-3629). IEEE.

Cannon, R. L., Dave, J. V., & Bezdek, J. C. (1986). Efficient implementation of the fuzzy c-means clustering algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2), 248-255.

Chen, C. S., Hwang, J. C., & Huang, C. W. (1997). Application of load survey systems to proper tariff design. *Power Systems, IEEE Transactions on*, *12*(4), 1746-1751.

Chang, R. F., & Lu, C. N. (2003, July). Load profiling and its applications in power market. In *Power Engineering Society General Meeting, 2003, IEEE* (Vol. 2). IEEE.

Chicco, G., Napoli, R., Postolache, P., Scutariu, M., & Toader, C. (2001). Electric energy customer characterisation for developing dedicated market strategies. In *Power Tech Proceedings, 2001 IEEE Porto* (Vol. 1, pp. 6-pp). IEEE.

Chicco, G., Napoli, R., Piglione, F., Postolache, P., Scutariu, M., & Toader, C. (2002, September). A review of concepts and techniques for emergent customer categorisation. In *Proc. Telmark Discussion Forum* (pp. 2-4).

Chicco, G., Napoli, R., & Piglione, F. (2003, June). Application of clustering algorithms and self organising maps to classify electricity customers. In *Power Tech Conference Proceedings, 2003 IEEE Bologna* (Vol. 1, pp. 7-pp). IEEE.

Chicco, G., Napoli, R., Piglione, F., Postolache, P., Scutariu, M., & Toader, C. (2004). Load pattern-based classification of electricity customers. *Power Systems, IEEE Transactions on*, *19*(2), 1232-1239.

Chicco, G. (2005, June). Comparing load profiling techniques for revenue and load deviation assessment. In *Power Tech, 2005 IEEE Russia* (pp. 1-6). IEEE.

Chicco, G., Napoli, R., Piglione, F., Postolache, P., Scutariu, M., & Toader, C. (2005, March). Emergent electricity customer classification. In *Generation, Transmission and Distribution, IEE Proceedings*- (Vol. 152, No. 2, pp. 164-172). IET.

Chicco, G., Napoli, R., & Piglione, F. (2006). Comparisons among clustering techniques for electricity customer classification. *Power Systems, IEEE Transactions on*, *21*(2), 933-940

Chicco, G., & Akilimali, J. S. (2010). Renyi entropy-based classification of daily electrical load patterns. *IET generation, transmission & distribution*, *4*(6), 736-745

Cho, H., Goude, Y., Brossat, X., & Yao, Q. (2013). Modeling and forecasting daily electricity load curves: a hybrid approach. *Journal of the american statistical association*, *108*(501), 7-21.

Chouakria-Douzal, A., & Nagabhushan, P. N. (2007). Adaptive dissimilarity index for measuring time series proximity. *Advances in Data Analysis and Classification*,*1*(1), 5-21.

Dash, M., Petrutiu, S., & Scheuermann, P. (2004, January). Efficient parallel hierarchical clustering. In *Euro-Par 2004 Parallel Processing* (pp. 363-371). Springer Berlin Heidelberg.

Davis, J., & Goadrich, M. (2006, June). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233-240). ACM.

Farnstrom, F., Lewis, J., & Elkan, C. (2000). Scalability for clustering algorithms revisited. *ACM SIGKDD Explorations Newsletter*, *2*(1), 51-57.

Farvaresh, H., & Sepehri, M. M. (2011). A data mining framework for detecting subscription fraud in telecommunication. *Engineering Applications of Artificial Intelligence*, *24*(1), 182-194.

Felea I., Dan F., & Dzitac S. (2012). Consumers load profile classification correlated to the electric energy forecast. *Proceedings of The Romanian Academy, Series A,* (Vol. 13, No. 1, pp. 80-88).

Figueiredo, V., Rodrigues, F., Vale, Z., & Gouveia, J. B. (2005). An electric energy consumer characterization framework based on data mining techniques.*Power Systems, IEEE Transactions on*, *20*(2), 596-602.

Gabaldon, A., Guillamon, A., Ruiz, M. C., Valero, S., Alvarez, C., Ortiz, M., & Senabre, C. (2010). Development of a methodology for clustering electricity-price series to improve customer response initiatives. *Generation, Transmission & Distribution, IET*, *4*(6), 706-715.

Gerbec, D., Gasperic, S., Smon, I., & Gubina, F. (2005). Allocation of the load profiles to consumers using probabilistic neural networks. *Power Systems, IEEE Transactions on*, *20*(2), 548-555.

Göetz, B., Peierls, T., Bloch, J., Bowbeer, J., Holmes, D., & Lea, D. (2006).*Java concurrency in practice*. Addison-Wesley.

Gullo, F., Ponti, G., Tagarelli, A., Ruffolo, M., & Labate, D. (2009, September). Low-voltage electricity customer profiling based on load data clustering. In*Proceedings of the 2009 International Database Engineering & Applications Symposium* (pp. 330-333). ACM

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*(2-3), 107-145.

Kitayama, M., Matsubara, R., & Izui, Y. (2002). Application of data mining to customer profile analysis in the power electric industry. In *Power Engineering Society Winter Meeting, 2002. IEEE* (Vol. 1, pp. 632-634). IEEE.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264-323.

Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, *1*(2), 111-117.

Labeeuw, W., & Deconinck, G. (2013). Residential Electrical Load Model based on Mixture Model Clustering and Markov Models.

Looney C. G. (2002) *Pattern Recognition*. *Opto-mechatronic systems handbook: techniques and applications*. CRC Press.

López, J. J., Aguado, J. A., Martín, F., Muñoz, F., Rodríguez, A., & Ruiz, J. E. (2011). Hopfield–K-Means clustering algorithm: A proposal for the segmentation of electricity customers. *Electric Power Systems Research*,*81*(2), 716-724.

Mahmoudi-Kohan, N., Moghaddam, M. P., Bidaki, S. M., & Yousefi, G. R. (2008, September). Comparison of modified k-means and hierarchical algorithms in customers load curves clustering for designing suitable tariffs in electricity market. In *Universities Power Engineering Conference, 2008. UPEC 2008. 43rd International* (pp. 1-5). IEEE.

Mahmoudi-Kohan, N., Moghaddam, M. P., & Bidaki, S. M. (2009, March). Evaluating performance of WFA K-means and modified follow the leader methods for clustering load curves. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES* (pp. 1-5). IEEE.

Mahmoudi-Kohan, N., Moghaddam, M. P., Sheikh-El-Eslami, M. K., & Bidaki, S. M. (2009, July). Improving WFA k-means technique for demand response programs applications. In *Power & Energy Society General Meeting, 2009. PES'09. IEEE* (pp. 1-5). IEEE.

Mahmoudi-Kohan, N., Moghaddam, M. P., & Sheikh-El-Eslami, M. K. (2010). An annual framework for clustering-based pricing for an electricity retailer.*Electric Power Systems Research*, *80*(9), 1042-1048.

Morais, H., Vale, Z., Faria, P., & Ramos, S. (2013, April). Defining electricity tariffs using the knowledge about the consumers profiles in ELECON project. In*Innovative Smart Grid Technologies Latin America (ISGT LA), 2013 IEEE PES Conference On* (pp. 1-7). IEEE.

Mori, H. (2006, October). State-of-the-art overview on data mining in power systems. In *Power Systems Conference and Exposition, 2006. PSCE'06. 2006 IEEE PES* (pp. 33-34). IEEE.

Murtagh, F., & Contreras, P. (2011). Methods of Hierarchical Clustering. *arXiv preprint arXiv:1105.0121*.

Mutanen, A., Ruska, M., Repo, S., & Jarventausta, P. (2011). Customer classification and load profiling method for distribution systems. *Power Delivery, IEEE Transactions on*, *26*(3), 1755-1763.

Nizar, A. H., Dong, Z. Y., Jalaluddin, M., & Raffles, M. J. (2006, November). Load profiling method in detecting non-technical loss activities in a power utility. In *Power and Energy Conference, 2006. PECon'06. IEEE International* (pp. 82-87). IEEE.

Paetz, A. G., Kaschub, T., Jochem, P., & Fichtner, W. (2012). Demand response with smart homes and electric scooters—an experimental study on user acceptance. In *ACEEE Summer Study*.

Panapakidis, I. P., Simoglou, C. K., Alexiadis, M. C., & Papagiannis, G. K. (2012, September). Determination of the optimal electricity selling price of a retailer via load profiling. In *Universities Power Engineering Conference (UPEC), 2012 47th International* (pp. 1-6). IEEE.

Pao, Y. H., & Sobajic, D. J. (1992). Combined use of unsupervised and supervised learning for dynamic security assessment. *Power Systems, IEEE Transactions on*, *7*(2), 878-884.

Piao, M., Lee, H. G., Park, J. H., & Ryu, K. H. (2008). Application of Classification Methods for Forecasting Mid-Term Power Load Patterns. In*Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques* (pp. 47-54). Springer Berlin Heidelberg.

Piao, M., Li, M., & Ryu, K. H. (2010, June). Using Significant Classification Rules to Analyze Korean Customers' Power Consumption Behavior: Incremental Tree Induction using Cascading-and-Sharing Method. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*(pp. 1649-1653). IEEE.

Pölzlbauer, G. (2004, June). Survey and comparison of quality measures for self-organizing maps. In *Fifth Workshop on Data Analysis (WDA 2004)* (pp. 67-82).

Quinlan, J. R. (1996). Improved use of continuous attributes in C4. 5. *arXiv preprint cs/9603103*.

Ramos, S., & Vale, Z. (2008, April). Data mining techniques application in power distribution utilities. In *Transmission and Distribution Conference and Exposition, 2008. T&D. IEEE/PES* (pp. 1-8). IEEE.

Ramos, S., & Vale, Z. (2008, July). Data mining techniques to support the classification of MV electricity customers. In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* (pp. 1-7). IEEE.

Räsänen, T., Voukantsis, D., Niska, H., Karatzas, K., & Kolehmainen, M. (2010). Data-based method for creating electricity use load profiles using large amount of customer-specific hourly measured electricity use data. *Applied Energy*, *87*(11), 3538-3545.

Su, M. C., & Chou, C. H. (2001). A modified version of the K-means algorithm with a distance based on cluster symmetry. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *23*(6), 674-680.

Takefuji, Y., Lee, K. C., & Also, H. (1992). An artificial maximum neural network: a winner-take-all neuron model forcing the state of the system in a solution domain. *Biological Cybernetics*, *67*(3), 243-251.

Theodoridis, S., Pikrakis, A., Koutroumbas, K., & Cavouras, D. (2010).*Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*. Access Online via Elsevier.

Tsekouras, G. J., Kanellos, F. D., Kontargyri, V. T., Karanasiou, I. S., Salis, A. D., & Mastorakis, N. E. (2008). A new classification pattern recognition methodology for power system typical load profiles. *WSEAS Transactions on Circuits and Systems*, *7*(12), 1090-1104.

Verdú, S. V., García, M. O., Franco, F. J. G., Encinas, N., Marin, A. G., Molina, A., & Lazaro, E. G. (2004, October). Characterization and identification of electrical customers through the use of self-organizing maps and daily load parameters. In *Power Systems Conference and Exposition, 2004. IEEE PES*(pp. 899-906). IEEE.

Verdú, S. V., Garcia, M. O., Senabre, C., Marín, A. G., & Franco, F. J. G. (2006). Classification, filtering, and identification of electrical customer load patterns through the use of self-organizing maps. *Power Systems, IEEE Transactions on*, *21*(4), 1672-1682

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map.*Neural Networks, IEEE Transactions on*, *11*(3), 586-600.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Elsevier.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, *16*(3), 645-678.

Zareipour, H., Janjani, A., Leung, H., Motamedi, A., & Schellenberg, A. (2011). Classification of future electricity market prices. *Power Systems, IEEE Transactions on*, *26*(1), 165-173.

Zhang, R., Qi, G., Li, C., Li, L., Bao, Y., & Zhu, Y. (2013, March). Forecasting of load model based on typical daily load profile and BP neural network. In *Fifth International Conference on Machine Vision (ICMV 12)* (pp. 87841G-87841G). International Society for Optics and Photonics.