POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

The Typical Load Days: A Clustering Problem

TESI MAGISTRALE IN ELECTRICAL ENGINEERING – INGEGNERIA ELETTRICA

AUTHOR: SIMONE PIVARI

ADVISOR: ALBERTO BERIZZI

ACADEMIC YEAR: 2021-2022

1. Introduction

The work is aimed to determine the representative or typical electrical load days along one year by applying a modern technique, under a strong development process: time series clustering. Indeed, though three different clustering algorithms, a dataset composed by 365 daily load curves is clustered into 36 or less clusters, and for each cluster a representative day is selected (typical load day). The obtained number of clusters is not random: it is derived through some quality indexes that will be further presented. In the following, first a brief theoretical explanation on time series, similarity measures and clustering will be provided. Then, the employed algorithms will be presented, and some simulations will be run on Matlab and R-studio to determine the typical load days from a real-life dataset. The result will show that it is possible to consider only the obtained representative days, instead than the complete dataset, as input to many different specific algorithms, obtaining coherent outputs in a fraction of the necessary computational time.

2. Scientific approach to clustering

First, it is convenient to provide some theoretical details on time series, similarity measures and clustering. More precisely, three subsections are considered below: time-series (section 2.1), dissimilarity measures (section 2.2) and time-series clustering (section 2.3).

2.1. Time series analysis

The term *time series* denotes a succession of values obtained by observation of a phenomenon, ordered according to the time variable. In formal terms:

$$Y(t) = \{y_1, y_2, \dots, y_t, \dots, y_N\} \tag{2.1}$$

N is called duration of the series and corresponds to the number of observations. It is important to remark that each observation y_i can be composed by multiple variables. This type of series, in which each observation y_i is a vector containing multiple variables, are called *multivariate time* series and, for

the sake of simplicity, will not be considered in this work. Two different approaches are possible to study a time series: the "classical" approach or *deterministic approach* and the "modern" approach or *statistical approach.* In the first case, it is assumed that the process represented by the series has a deterministic nature, which allows to decompose the series in four virtual components that can be estimated [1]. These components are called trend component (long-term progression of the series), cyclical component (repeated but not periodic fluctuations), seasonal component (seasonal variations) and irregular component (random or irregular behavior of the series). In formal terms, it is possible to define the following relationship for a considered time series:

$$Y_t = f(T_t, C_t, S_t, I_t) \tag{2.2}$$

where $t = 1, \dots, N$.
The second approach, instead, assumes that the series has been generated by a stochastic process, so that each observation is the realization of a random variable Y_t [2]. More formally:
A time series model for the observed data $\{y_t\}$ *is the specification of the joint distribution (or evenly of the mean and the covariance) of a sequence of random variables* $\{Y_t\}$ *of which* $\{y_t\}$ *is postulated to be a realization.*
Generally, only the first order moment and the second order moment of the joint distributions are specified. Moreover, even in this case the decomposition already presented is valid, adopting different techniques to estimate the virtual components of the series with respect to the deterministic approach.
Since time series decomposition will not be employed in this work, focused instead on time series clustering, it is convenient to move on to the next subsection, that explains the concept of similarity and dissimilarity measures between time series.

2.2 Similarity and dissimilarity measures

The theoretical issue of time series similarity/dissimilarity search has been proposed by Agrawal et al. [3] and subsequently it became one of the main research areas in data mining and clustering, since this latter relies on dissimilarity measures to perform. In time series clustering, the degree of similarity or dissimilarity between time series is calculated approximately. Indeed, typically a distance function is exploited to determine the distance measurement between all the points of the multiple time series to be compared, and an estimated distance between time series, representing the (dis)similarity, is then derived. More precisely, let Y_t be a time series representing a set of data. A function $s: Y_t \times Y_t \to \mathbb{R}$ is called *similarity* on Y_t if it satisfies the following properties, $\forall\, y_i, y_j \in Y_t$:

- Non-negativity: $s(y_i, y_j) \geq 0$ (2.3)
- Symmetry: $s(y_i, y_j) = s(y_j, y_i)$ (2.4)
- If $y_i \neq y_j$, $s(y_i, y_i) = s(y_j, y_j) > s(y_i, y_j)$ (2.5)

Instead, a function $d: Y_t \times Y_t \to \mathbb{R}$ is called *dissimilarity* on Y_t if it satisfies the following properties, $\forall\, y_i, y_j \in Y_t$:

- Non-negativity: $d(y_i, y_j) \geq 0$ (2.6)
- Symmetry: $d(y_i, y_j) = d(y_j, y_i)$ (2.7)
- Reflexivity: $d(y_i, y_i) = 0$ (2.8)

Typically, when the dissimilarity measure between two time series has a low value, the distance measure between them has a low value too. This suggests the use of distance metrics to determine the degree of similarity between time series, as already done in many papers and works [4]. Moreover, typically the methods to determine similarities between time series can be classified under three main categories:

- Similarity in time: distance measures in which the time of occurrence of patterns is crucial.
- Similarity in shape: distance measures in which the time of occurrence of patterns is

not important. More precisely, clusters of time series with similar patterns are constructed regardless of the time instants.
- Similarity in change: in this approach modelling methods are employed to represent the time series and then a similarity measure is applied on the parameters of the fitted model.

Different metrics determine different similarity categories. Anyway, once a metric has been selected, the distance between the time series under consideration is typically computed as the sum of the distances between individual points. More formally, if Y_t and $Y_t{}'$ are two time series of length N representing a set of data and y_i, y_j' are generic points of the first and second series respectively:

$$D(Y_t, Y_t') = \sum_{t=1}^{N} D(y_i, y_j'), \quad \forall i, j \in [1, N] \tag{2.9}$$

Of course, equation (2.9) supposes that the series under consideration have the same number of points. If this assumption is true, typically the Euclidean distance is used as distance metric. Instead, if this assumption is not true, other measures (typically non-metric) must be adopted. More precisely, a modern technique called *Dynamic Time Warping (DTW)* is exploited in this case [5]. DTW allows to address the issue of time shift sensitivity of Euclidean distance and makes also possible to compare two time series with different length. Indeed, since with Euclidean distance method the distance measure is computed for every couple of points corresponding to the same time instants, the two considered series must have the same length. Instead, DTW computes the distance measure for every possible couple of point and determines the minimum/maximum distance between the two series, thus finding their optimal alignment regardless of their length. It is important to recall that Euclidean distance defines a similarity in time between the two considered time series. Indeed, as already discussed, the fact that it is computed only for points of a correspondent time instant implies that the time occurrence of a pattern is crucial (definition of similarity in time). This underlines the great sensitivity to time shifts of this distance metric, an open issue that affects this method. Instead, DTW is based on similarity in shape, since it determines the optimal alignment between time series regardless of their time occurrence [6]. More precisely, if Y_i and Y_j are two time series to be compared of length N and M respectively, the algorithm starts by computing the distance matrix $C: N \times M$, that contains all the pairwise distances between Y_i and Y_j. Indeed, the (h, k) element of C corresponds to the Euclidean distance between $y_{ih} \in Y_i$ and $y_{jk} \in Y_j$. The distance matrix is commonly called *local cost matrix*. Formally:

$$C: N \times M, \quad c(h,k) = |y_{ih} - y_{jk}|, \\ h \in \{1, \dots, N\}, k \in \{1, \dots, M\} \tag{2.10}$$

Once the local cost matrix has been computed, the algorithm determines the alignment path which minimizes the total global cost, given by the sum of global costs along the path. It is also possible to plot a heatmap of the global cost matrix to highlight that the optimal alignment path runs through its low-cost areas. The alignment path, also called *warping path* or *warping function*, defines the correspondence of an element $y_{ih} \in Y_i$ to an element $y_{jk} \in Y_j$. More formally, it is a sequence of points $P = (p_1, p_2, \dots, p_L)$ with $p_l = (p_i, p_j) \in [1, \dots, N] \times [1, \dots, M]$ for $l \in [1, \dots, L]$. The main steps to build up the global cost matrix and to find the optimal warping path are now discussed. First, the algorithm computes the upper left corner element of the global cost matrix, as the Euclidean distance between the first elements of the two time series to be compared. Then, the elements of the first row/column are determined as the sum of the elements of the local cost matrix in the first row/column until the position of the considered element. Successively, all the remaining elements of the DTW matrix are computed iteratively according to:

$$D(h,k) = \min \begin{Bmatrix} D(h-1, k-1) + c(y_{ih}, y_{jk}), \\ D(h-1, k), D(h, k-1) \end{Bmatrix} \\ + c(y_{ih}, y_{jk}), \\ h \in (1, \dots, N), k \in (1, \dots, M) \tag{2.11}$$

Once that the global cost matrix has been obtained, the optimal warping path and the global distance are derived as already explained.

For instance, the following figures show the heatmap of the global cost matrix and the so-called three-way plot after the application of DTW to the following series:

- First series: $Y_1 = \{7,9,6,9,12,6,4,6,8\}$
- Second series: $Y_2 = \{5,6,4,3,9,5,6,8,9\}$

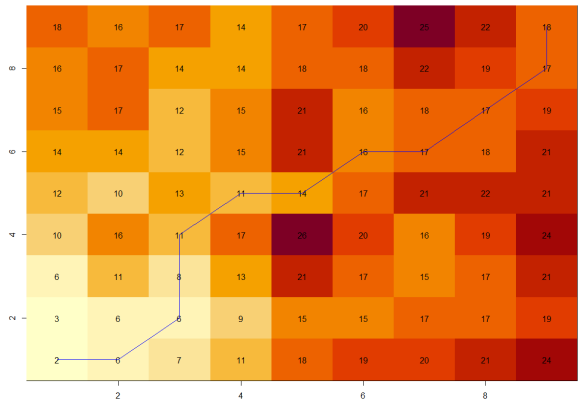

Figure 2.1: global cost matrix heatmap and optimal warping path

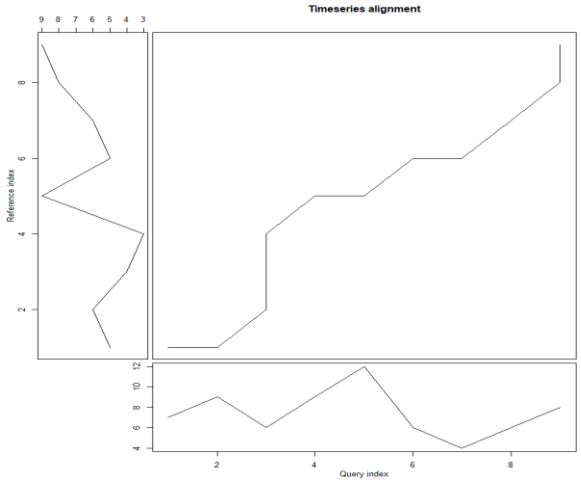

Figure 2.2: three-way plot of time series alignment

The three-way plot places one series horizontally in a small lower panel, the other series vertically on a left panel and a larger inner panel holds the warping curve. In this way, matching points can be recovered by tracing indices on the first time series, commonly called *query series*, moving upwards until the warping curve is met, and then moving leftwards to discover the index of the other matched series, commonly called *reference series*.

2.3 Time series clustering

Before analysing in detail time series clustering, it is important to mention the so-called data preparation methods, employed to normalize, scale and transform data to achieve time-shift invariance and insensibility to possible offsets. Different possible methods can be exploited, but the most common is the z-normalization, which also used in this work. This technique transforms the time series to the same time scales by normalizing them. The new scaled element y_i' can be obtained from its related original element y_i as:

$$y_i' = \frac{y_i - \mu}{\sigma} \tag{2.12}$$

In equation (2.12), μ and σ represent respectively the mean and standard deviation of the considered time series of length N. After having applied a data preparation method, a similarity measure must be selected for the clustering algorithm to perform. Indeed, before the algorithm can cluster data, it must be defined how similar pairwise elements are. As already discussed in the previous section, the most employed similarity metrics are commonly the Euclidean distance and the dynamic time warping, which determine similarity between time series in time and shape, respectively. After having prepared the data and selected a similarity measure, the clustering algorithm exploits this latter to cluster data. Eventually, it is important to carefully check the quality of clustering output; this is done by computing some quality parameters that indicates the goodness of the result. Figure 2.3 [7] shows graphically the main steps to perform the clustering process, as discussed above.

Figure 2.3: main steps to perform clustering on a dataset

Different classifications are possible for clustering algorithms. First of all, in this work the so-called whole time series clustering is considered. This approach is based on clustering a set of time series with respect to their similarity, in contrast with the

time-point clustering approach, in which points of a single time series are clustered basing on a combination of their temporal proximity and similarity. Moreover, whole time series clustering algorithms can be classified into:

- *Partitional* algorithms: given a dataset of N objects, a partitional algorithms creates k partitions (clusters) of them, with $k \leq N$. Typically, the number of partitions to be obtained k is an input of the algorithm that begins creating an initial partitioning and applies an iterative re-locating technique which moves objects from one cluster to another to improve the result.
- *Hierarchical* algorithms: these algorithms create a hierarchical decomposition of a set of N objects. Moreover, they can be classified in *agglomerative (bottom up)* or *divisive (top-down)*, according to how the decomposition is obtained.

Other categories can be added to the previous list. Anyway, for the purpose of this work, the two mentioned categories are sufficient. Before examining the considered algorithms, it is necessary to briefly analyse the possible "representative elements" of a cluster, formally called *cluster prototypes* or *cluster centroids*. Given a cluster $C = \{Y_1, Y_2, \dots, Y_N\}$ containing N time series, its prototype $\hat{R}$ minimizes the distance between all time series in the cluster and the prototype itself. More precisely:

$$\hat{R} = \min_R\{D(C, R)\} = \min_R\{\frac{1}{N}\sum_{i=1}^{N} D(Y_i, R)\} \tag{2.13}$$

where D indicated a generic distance measure. Generally, three main cluster prototypes have been defined in literature: the mean (average value of a cluster), the medoid (element of a cluster which minimizes the sum of squared distances to other objects within the cluster) and the local search prototype (combination of mean and medoid).

2.4 Optimal number of clusters and quality indexes

This last subsection focuses on determining the optimal number of clusters. This is a very important task, especially when there is no prior knowledge on the data. Many different methods have been proposed in literature; the most commonly employed are the so-called *elbow method* [8] and *gap statistical method* [9]. The elbow method is a very simple strategy that is based on an iterative process. In particular, the sum of squared errors is used as an indicator to detect the optimal number of clusters. Indeed, a clustering algorithm (typically partitional, to reduce the computational complexity) is iteratively performed considering a required number of clusters ranging from 1 to +infinite, and for every clustering the sum of squared errors is computed. Then, a graph of the SSE vs the number of clusters is plotted. This graph will show a strong increase in the first part, since increasing the number of clusters will add intra-cluster variance, and then after an elbow the increase is strongly reduced, meaning that increasing the number of clusters does not affect any more significantly the sum of squared errors. Therefore, the elbow point of the graph is selected as optimal and the related number of clusters K is the optimal number of clusters. Instead, the basic idea behind the Gap Statistics technique is to choose the number of K for which the biggest variation in within-cluster distance occurred, based on the overall behavior of uniformly drawn samples. In practice, the optimal number of clusters is selected such that, increasing K, the gap function starts decreasing and the maximum variation of within cluster distance is obtained. After having analyzed the techniques to determine the optimal number of clusters, it is convenient to briefly examine the parameters for the clustering evaluation. In particular, two main clustering quality indexes can be defined: the *Dunn index* [10] and the *Davies-Bouldin index* [11]. The Dunn index determines the quality of the clustering result by detecting if clusters are well-separated between each other. More precisely, it is defined as follows:

$$Q_D = \min\{separation\}/\max\{diameter\} \tag{2.14}$$

In equation (2.14), $separation$ refers to the inter-cluster distance, while $diameter$ refers to intra-cluster distance. Thus, the Dunn quality index is determined as the ratio between the minimum inter-cluster distance (distance between data points in different clusters) and the maximum intra-cluster distance (distance between data points in the same cluster). The value of this

parameter is high if clusters are well-separated, meaning that the clustering process has been efficient. The Davies-Bouldin index is a sort of inverse of the Dunn index, anyway computed with different formulas. More precisely, it is defined as the average value of the ratios of within-cluster distances and between clusters distances. The lower is the value of the Davies-Bouldin index, the higher is the quality of the clustering result. Indeed, if clusters are well-separated, the inter-cluster distance, represented by the distance between centroids (denominator) is high, while the intra-cluster distance, represented by the variance (numerator), is low.

2.5 Employed algorithms

The first considered clustering algorithm is called k-means. It can be summarized in five steps as follows:

1. Initialize the number of clusters K
2. Randomly select K datapoints as clusters centroids
3. Assign each point to the cluster with closer centroid to the considered point
4. Recalculate the centroids based on the recent cluster assignment of data points
5. Repeat steps 3 and 4 until centroids no longer vary or the convergence of a certain criterion function (typically sum of squared errors) is reached.

K-means clustering is relatively scalable, very efficient in clustering big datasets due to its quite low computational complexity and generally reaches a local optimum. However, it cannot be applied if the average value of the data cannot be defined, for instance in case of presence of categorical variables. Eventually, this algorithm is sensitive to noise and outliers, since a small number of these data can sensibly affect the average values [12].
K-medoids algorithm has been introduced to address the issues of k-means. Indeed, it is less sensitive to noise and outliers, resulting in a more robust algorithm [13]. More precisely, instead of using the mean as the centroid of a cluster, k-medoids selects an actual point in the cluster to represent it. This choice for the computation of the centroids allows to greatly reduce the sensitivity of the algorithm to outliers, since their presence affect significantly only the mean and not the medoid of the cluster, as it does not modify crucially the data distribution. However, the computational complexity of k-medoids is higher than k-means complexity, since determining the value of the median is more computationally expensive than determining the value of the mean. It is possible to summarize the main performed steps as follows:

1. Initialize the number of clusters K
2. Randomly select K datapoints as clusters centroids
3. Assign each data point to its nearest cluster by minimizing the sum of dissimilarities between each point and its medoid.
4. Recalculate the medoids of each cluster by determining the object that decreases its average dissimilarity coefficient (cost function).
5. If there is no change in the medoids the algorithm stops. If medoids still change then steps 3 and 4 must be repeated until medoids does not vary anymore or until the convergence of a criterion function (typically sum of absolute errors) is reached.

The base strategy of the k-medoids algorithm is to partition the N object into K clusters. As for k-means, K is an input parameter and must be defined a priori by the user. As the previously presented steps highlight, K objects are first randomly selected as medoids of the clusters. Then, each remaining object is inserted in the cluster related to the most similar medoid to the object itself. The similarity is typically computed through Euclidean distance or dynamic time warping measures. Successively, the algorithm iteratively substitutes medoids o_j with all the non-medoids objects o_{random}, stopping when the clustering quality cannot be improved anymore. This quality is estimated through a cost function that measures the average dissimilarity of an object and the medoid of its cluster. Every time that a reassignment of data occurs, the cost function varies from its previous value; this variation can be positive or negative. The sum of all the variations of the cost functions obtained after the reassignment of all the non-medoids objects according to the previous cases is called *total swapping cost*. If the total swapping cost is negative, o_j is substituted with o_{random} since the total error is reduced. Instead, if the total swapping cost is positive, the current medoid o_j is considered acceptable and is not substituted. The examined

procedure is repeated until medoids do not vary anymore or until the sum of absolute errors is lower than a defined threshold.

Ward hierarchical algorithm [14] is an agglomerative hierarchical algorithm. More precisely, it starts by considering each object as a separate cluster and then merges these atomic clusters until some determined conditions are satisfied. The various algorithms inside this category differ just for the computation of the similarity measure between clusters (inter-cluster similarity). If p and p' are two generic objects belonging to different clusters C_i and C_j, $mean_{C_i}$ is the mean value of the cluster C_i and N_i is its number of objects, the distance measure employed in Ward algorithm is:

$$d_{Ward}(C_i, C_j) = \frac{N_i N_j}{N_i + N_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|^2 \quad (2.15)$$

Moreover, the main steps to be performed are:

1. Set the initial number of clusters n equal to the total number of objects N.
2. Determine the centroid of each cluster as the mean value of the cluster itself.
3. Compute the distance between each pair of clusters according to Ward's method.
4. Merge the two closest clusters.
5. Update $n = n - 1$
6. If a predefined condition is satisfied (for instance, the number of clusters is equal to a desired value) go to step 7, otherwise go to step 2.
7. Determine the centroid of each cluster as the cluster medoid.

The previous steps can be graphically summarized by the following example:

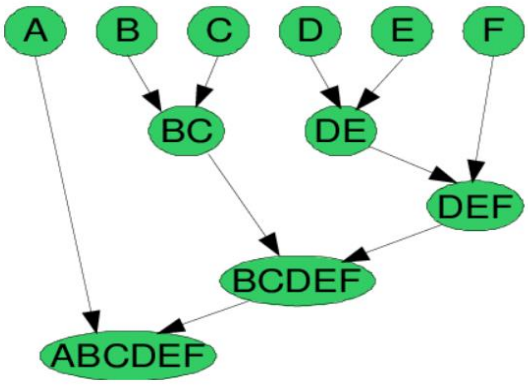

Figure 2.4: agglomerative hierarchical clustering example

3. Clustering implementation: an electrical engineering problem

The considered dataset of load duration curves can be downloaded from the ENTSO-E power statistics website [15]. Once on the website, many sections are available; the set of curves under analysis can be found in "Monthly Hourly Load Values", that reports the aggregated monthly electrical power consumption of different countries on an hourly basis, from 2015 to 2019. The 365 selected load curves correspond to the electrical consumption of Germany in 2015 (one curve per day, with 24 points per curve).

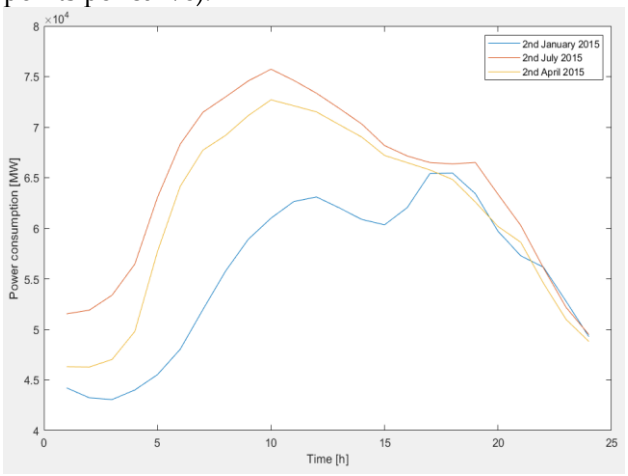

Figure 3.1: examples of the considered load duration curves

Figure 3.1 shows three days of different months. For instance, the blue curve represents the aggregated electrical power consumption of Germany on the 2nd January 2015, the yellow curve represents the aggregated power consumption of Germany on the 2nd April 2015 and the red curve represents the aggregated power consumption of Germany on 2nd July 2015. As it is possible to observe, in July the electrical energy consumption is higher than in January or April due to the usage of air conditioning systems. This is generally true for all summer months with respect to winter months, in which the power consumption is lower. Once having examined the considered dataset, as already discussed, the first task to be performed is a data-preparation method in order to make the clustering process less sensitive to scaling, time shifting and offsets. For the sake of simplicity, the z-normalization method has been selected for this practical implementation, since it allows to obtain

anyway a clustering model with low sensitivity. After having normalized the dataset, it is possible to perform the clustering process to determine the typical load days. Three main clustering algorithms have been employed, as already defined in the previous chapter: k-means, k-medoids and Ward's hierarchical algorithm. Moreover, two softwares have been considered: Matlab and R-Studio.

3.1 Matlab implementation

The algorithms needed to perform clustering can be found in the "Statistical and Machine Learning Toolbox". The Matlab function to perform k-means or k-medoids algorithms receives as input a matrix containing the dataset, the desired number of clusters and the similarity measure to be used. Instead, it produces as output a vector containing the cluster indexes for each of the load duration curves, a matrix containing the centroids of the obtained clusters, a vector containing the sum of within-cluster series-to-centroids distances. and a matrix containing distances between each time series of the dataset and every centroid. By clustering through k-means algorithm and k-medoids algorithms the set of 365 load curves, 36 clusters are obtained. The figures below report some results of the clustering process, showing the curves belonging to a cluster and their related cluster centroid, represented by a line with a bigger width.

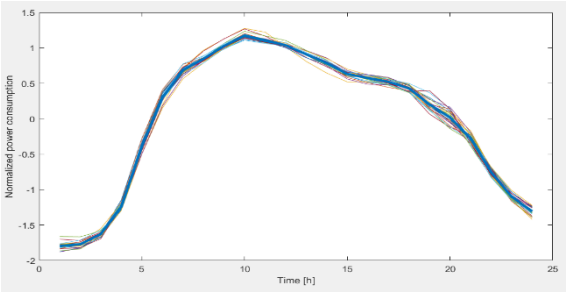

Figure 3.2: cluster1 obtained by k-means

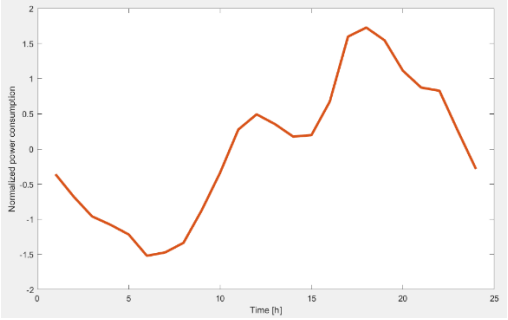

Figure 3.3: cluster21 obtained by k-means

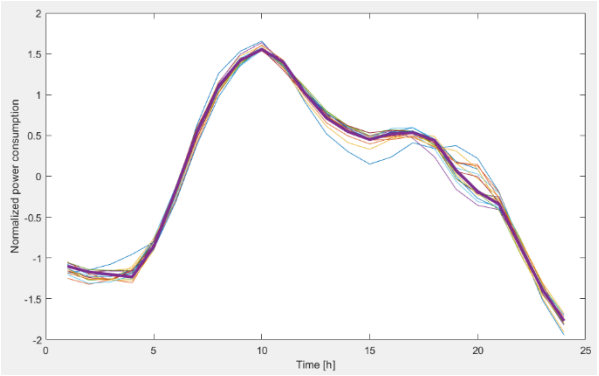

Figure 3.4: cluster1 obtained by k-medoids

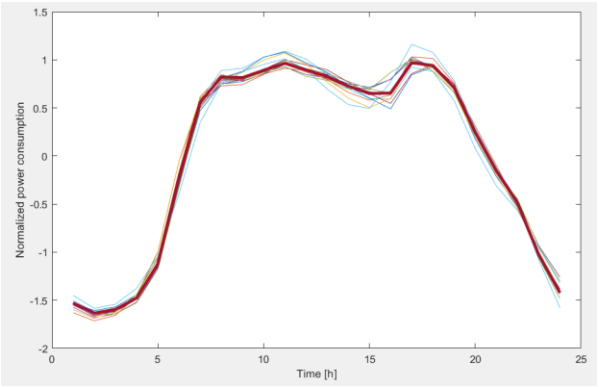

Figure 3.5: cluster21 obtained by k-medoids

The centroids of the determined clusters are called *typical days* or *representative days* of electrical consumption. It is important to remark that in the k-means algorithms, the typical days that are obtained are not actual days of electrical consumption in Germany, since they are computed as the average values of the curves belonging to a cluster. Instead, in the k-medoids algorithms, the obtained typical days are actual days of electrical consumption. This allows to define the most important days of the year in term of representation. Two different indexes will be analysed in order to determine the optimal number of clusters and the quality of clustering result. These indexes have a different nature: the first is the percentage mean error computed on the load duration curves (the real one and another one built up with clustered data), while the second is the already presented Dunn index. Figure 3.6 shows the real load duration curve and the reconstructed one from clustered data in 36 clusters. The reconstructed curve is sometimes higher, sometimes lower than the real load duration curve, so it is possible to compute the mean percentage error between the two curves as:

$$MPE = \frac{1}{N}\sum_{t=1}^{N}\frac{|dur_{rec,t} - dur_{real,t}|}{dur_{real,t}} \times 100 \quad (3.1)$$

For a number of clusters $K = 36$, the computation of the mean percentage error gives $MPE = 0.17\%$. Therefore, the real load duration curve is well approximated by the reconstructed load duration curve using only the clustered data. If the number of clusters is set to an extremely low value $K = 2$, the computation of the mean percentage error gives $MPE \approx 2\%$. The MPE has increased by more then ten times with respect to considering 36 clusters, thus reducing the accuracy of the clustering model and the goodness of fit of the typical days.

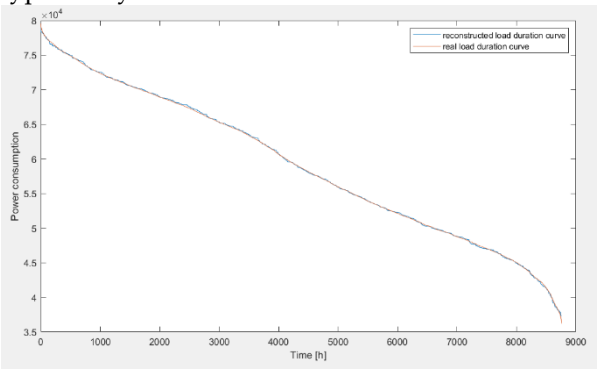

Figure 3.6: real and reconstructed load duration curves for K=36

The other parameter exploited to determine the optimal number of clusters is the Dunn index. As already mentioned, the clustering algorithms has been performed with a variable number of clusters ranging from 1 to 40 and for every clustering output the Dunn index has been computed. The result can be observed in figure 3.7:

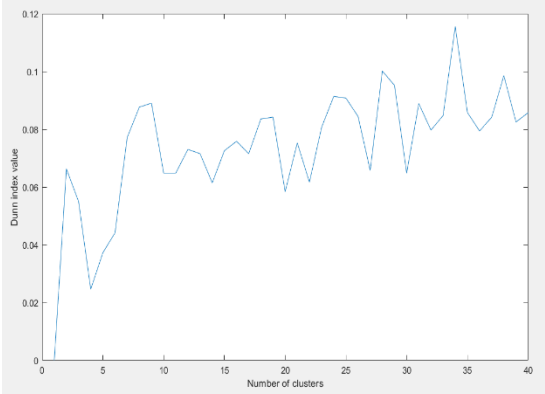

Figure 3.7: Dunn index values for different number of clusters

Figure 3.7 shows that the optimal number of clusters is $K = 34$, since the Dunn index reaches its highest peak for this number of clusters. The performed analysis considering 36 clusters is of course valid, since with $K = 36$ the Dunn index has still a high value that reflects the goodness of fit of the clustering model.

3.2 R-Studio implementation

R-studio includes a library for the computation of DTW distance measure and its use is very simple. The following figure shows the obtained optimal alignment between two time series (the same load curves reported in figure 3.1) by means of the three-way plot.

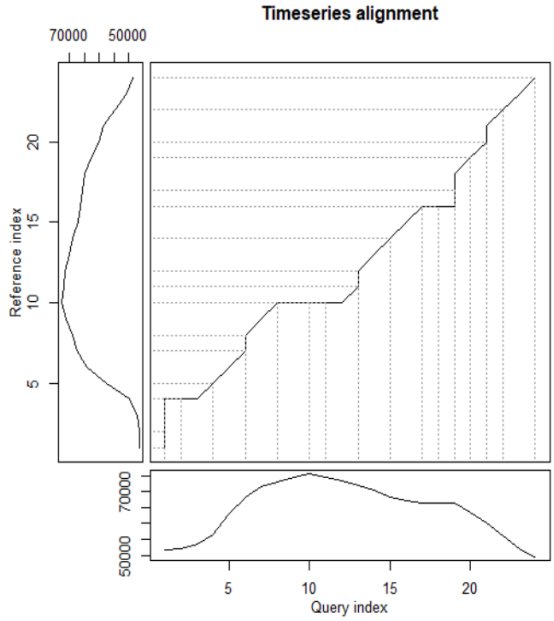

Figure 3.8: three-way plot of the optimal alignment

The function exploited to cluster the dataset is denoted as "tsclust". More precisely, it produces the same outputs as Matlab (vector of clusters belonging indexes, centroid matrix, distance matrix and sum) while the required inputs are the dataset containing the normalized load curves, the desired category of clustering algorithms, the desired number of clusters and the selected similarity measure. Some of the obtained clusters for $K = 2$ and for $K = 36$ are reported in the next figures.

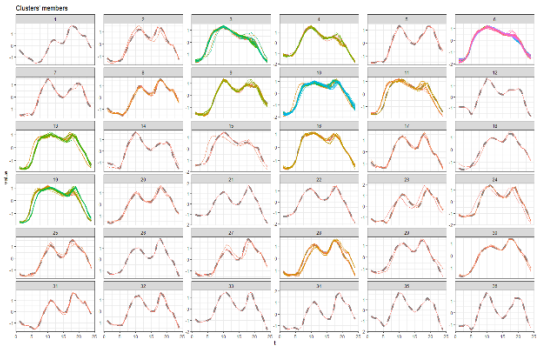

Figure 3.9: overview of the obtained clusters for K=36

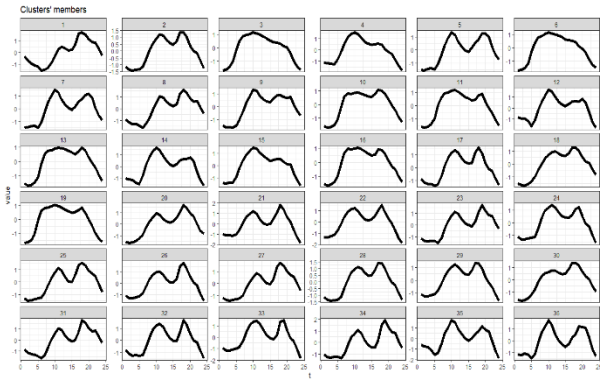

Figure 3.10: centroids (typical days) of the 36 obtained clusters

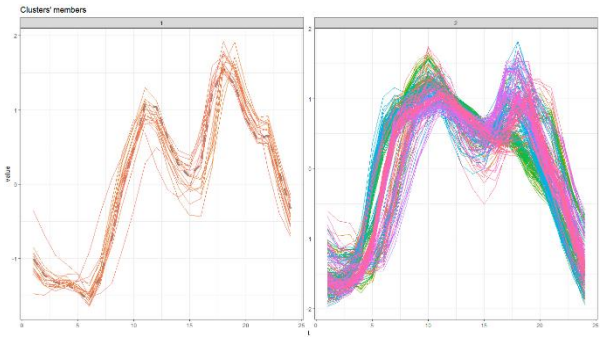

Figure 3.11: overview of the obtained clusters for K=2

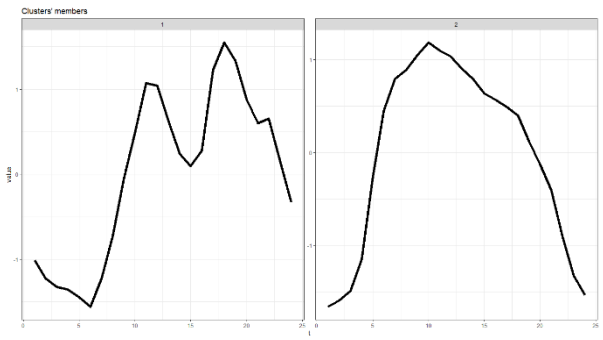

Figure 3.12: centroids of the 2 obtained clusters

As it is possible to note from figures 3.9 and 3.11, clustering the load curves in a too low number of clusters (K=2) results in a bad definition of the typical load days. Indeed, even if cluster1 centroid catches well the shape of its cluster participants, cluster2 centroid misses an important load peak around 19 p.m., thus it does not represent well the behavior of the load during the days belonging to cluster2 itself. Typically, exploiting hierarchical algorithms the quality of clustering results is higher and the optimal number of clusters is lower than using partitional algorithms. This can be proved by computing quality indexes as done for Matlab implementation. In particular, the following figures show the Dunn index, the Davies Bouldin index and the gap statistic behavior for different numbers of clusters.

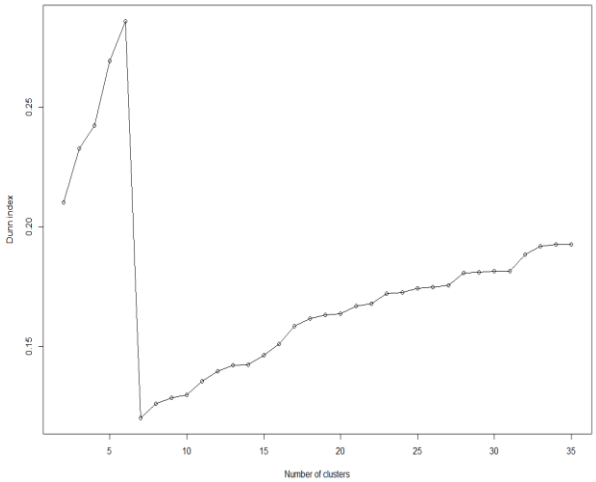

Figure 3.13: Dunn index for different numbers of clusters

As it is possible to note from figure 3.24, the highest value of the Dunn index is obtained for k=6. However, this is in contrast with the already used criterion of matching as much as possible the real load duration curve with a reconstructed one from clustered data. Making several attempts, it is possible to determine a good compromise between the Dunn index value and the error on the load duration curves with a number of clusters $K \geq 31$, coherently with the result obtained in the previous section.

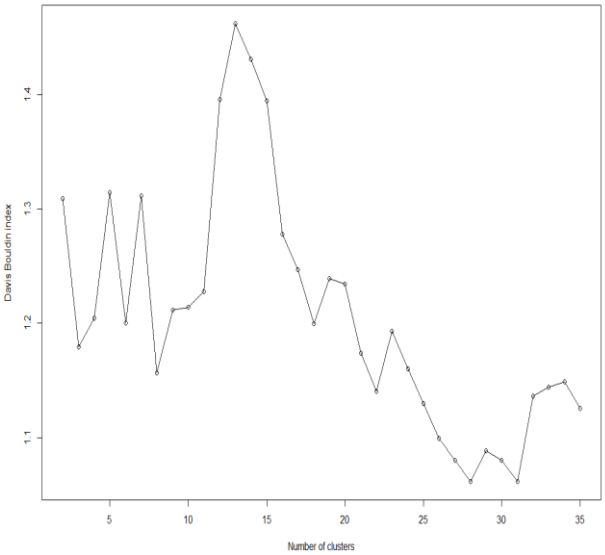

Figure 3.14: Davies-Bouldin index for different numbers of clusters

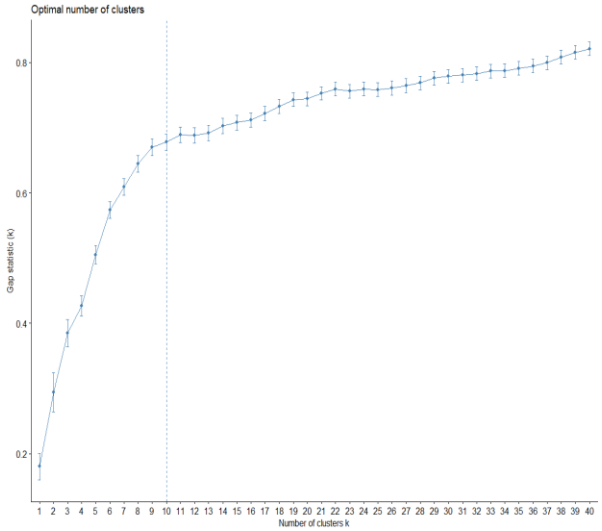

Figure 3.15: gap statistic value for different number of clusters

According to the Davies-Bouldin index, the optimal number of clusters is 31. Indeed, it is recalled that the best clustering model is the one that minimizes the Davies-Bouldin index and maximizes the Dunn index. This value is coherent with the tradeoff discussed in the previous page (it is possible to determine a good compromise between the Dunn index value and the error on the load duration curves with a number of clusters $K \geq 31$) and allows to obtain a very precise reconstructed load duration curve.

Eventually, figure 3.15 shows the gap statistic behavior for different number of clusters. As it is possible to note, the optimal value of clusters suggested by this parameter is K=10. Indeed, for $K \geq 10$ the gap statistic does not exhibit a great variation, while it increases rapidly until 10 clusters are considered. Of course, increasing the number of clusters increases the gap statistic value and the matching degree between the load duration curves (advantages) but the computational effort increases too and the values of other quality indexes can be reduced (disadvantages). Again, after several attempts, it is possible to determine a good compromise between the gap statistic value and the error on the load duration curves with a number of clusters $K \geq 21$,

3.3 Comparison of results

After having examined the obtained results with Matlab and R-Studio, it is convenient to make a brief comparison between them. First, as it is possible to note from the computation of the quality indexes, there is never a purely correct answer to a clustering problem. Different parameters provide different results and a compromise between them must be always found, without focusing too much on one of them but rather searching for the optimal tradeoff solution. For instance, with R-Studio a minimum number of clusters $K = 31$ has been determined as a good compromise between the Dunn index value and the error on the load duration curves. The same considerations can be made after the computation of the Davis-Bouldin index and the gap-statistic index. This result is coherent with the one obtained with Matlab. Indeed, as shown by the Dunn index plot (figure 3.15), the optimal number of clusters found with this software is $K = 35$. Moreover, the computation of the quality indexes has shown that, with a hierarchical clustering algorithm, the necessary number of clusters is lower than the one with a partitional clustering algorithm. However, the need to correctly represent the load duration curve increases this number up to the already determined value.

The previous comparison and the different results show that, with both softwares, the centroids of the obtained clusters (typical load days) are able to well-represent the behavior of the load along the year, leading to a small error on the load duration

curve and on its total area. To properly remark this conclusion, it is possible to check if the minimum/maximum load ramps are still present after having clustered the load curves. The results are reported in the following table.

	Maximum positive ramp [MW]	Maximum negative ramp [MW]
Original dataset	1.031×10^4	-0.612×10^4
Clustered dataset	1.056×10^4	-1.811×10^4

Table 3.1: Maximum positive and negative ramps for original and clustered dataset

As it is possible to observe from table 3.1, similar values of the maximum ramps are obtained after the clustering process. More precisely, it is possible to compute the percentage of error as:

$$E_{ramp,\%} = \frac{|ramp_{original} - ramp_{clustered}|}{|ramp_{original}|} * 100 \quad (3.12)$$

The computation of the previous error index using the values from table 3.1 gives $E_{ramp\ pos,\%} = 2.45\%$ and $E_{ramp\ neg,\%} = 66\%$. While the error on the maximum positive ramp is very low, the error on the maximum negative ramp is quite high. Anyway, the maximum negative ramp after clustering is higher (more negative) than the one before clustering. This can be a conservative hypothesis (assuming that the load varies more than its actual variation) not to undersize machines and components and to make worst-case scenario economical evaluations. On the other side, if this error must be reduced, it is necessary to include constraints in the clustering algorithm (for instance: impose a maximum negative ramp rate). This approach is not easy at all since it need to consider the chronological occurrence of data points, an issue that still affects the existent clustering algorithms.
The previous comparison shows that there is never a purely correct answer to a clustering problem. Different parameters provide different results and a compromise between them must be always found, without focusing too much on one of them but rather searching for the optimal tradeoff solution. Moreover, the clustering algorithms need a starting condition to perform and some crucial decisions are taken during operations. The facts that the starting condition is typically random and that the merging/splitting decisions are irreversible and can be affected by errors highlight the importance of interpreting the results. Due to this, data scientists are currently studying and implementing new clustering techniques, addressing all the issues that are still present in the clustering process.

4. Conclusions

The purpose of this work was to highlight the extreme importance that clustering has in everyday life. Moreover, it has provided a solid theoretical basis for understanding and implementing the main time-series clustering algorithms, applying them to a real-life electrical engineering problem by determining the representative load days, useful in many planning, sizing and development algorithms. Indeed, by considering only the representative days instead of the original curves, coherent results can be achieved in a fraction of the required computational time. This shows that clustering is essential also in electrical engineering or more generally in all the scientific areas in which big datasets are used as input of subject-specific algorithms. The goodness of the clustering model has been estimated through the computation of the presented quality indexes, and an optimal number of clusters has been found. Eventually, a comparison between the obtained results has been examined, showing that even if they are coherent between different softwares, there is never a purely correct answer to a clustering problem. Different parameters provide different results and a compromise between them must be always found, without focusing too much on one of them but rather searching for the optimal tradeoff solution.

5. Bibliography - References

[1] A. Marliani. *Serie storiche*, chapter 3: *metodi di scomposizione*, Università di Firenze, 2014.

[2] P.J. Brockwell, R. A. Davis. *Introduction to Time Series and Forecasting*. Springer-Verlag, New York, 1996.

[3] R. Agrawal., C. Faloutsos, Swami. *Efficient Similarity Search in Sequence Databases.* 4th International Conference, 1993.

[4] E. Keogh., S. Kasetty. *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration*. Data Mining and Knowledge Discovery, 2003.

[5] R. Bellman, R. Kalaba. *On adaptive control processes*. Automatic Control, IRE Transactions on, vol. 4, no. 2, 1959.

[6] P. Senin. *Dynamic Time Warping Algorithm Review.* Information and Computer Science department, University of Hawaii at Manoa, 2008.

[7] URL https://developers.google.com/machine-learning/clustering/overview

[8] R. L. Thorndike. *Who Belongs in the Family?* Psychometrika, 1953

[9] *R. Tibshirani, G. Walther, T. Hastie. Estimating the number of clusters in a dataset via the gap statistic.* Department of Health Research and Policy and Department of Statistics, Stanford University, Stanford, CA 94305, USA, 2001.

[10] *J. C. Dunn. Well-Separated Clusters and Optimal Fuzzy Partitions. Journal of Cybernetics, 4:1, 95-104, DOI: 10.1080/01969727408546059, 1974.*

[11] D. L. Davies, D. W. Bouldin. *A Cluster Separation Measure.* In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224-227, DOI: 10.1109/TPAMI.1979.4766909, 1979.

[12] Elbiomy, Alaa & Mousa, Salwa & Abo-Hussien, Amina. *A comparison study of k-means and k-medoids Algorithms With an application on COVID-19 Data as an example*. 2022.

[13] X. Jin., J. Han. *K-Means Clustering*. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA, 2011.

[14] J. H. Ward. *Hierarchical Grouping to Optimize an Objective Function.* Journal of the American Statistical Association, 1963.

[15] URL https://www.entsoe.eu/data/power-stats

6. Acknowleggements

I would like to thank my advisor Alberto Berizzi and co-advisor Marina Petrelli for this challenge, that has given me the opportunity to make a journey in the data-science world, a very fascinating research area that I'm glad to have deepened. Most importantly, I would like to thank my parents Attilia Ottoboni and Alberto Pivari for the love and support, also economical, that they always gave to me, without any back expectation. I love you, thanks for all.