Load Time-Series Classification Based on Pattern Recognition Methods

George J. Tsekouras[1-2], Anastasios D. Salis[2],
Maria A. Tsaroucha[2] and Irene S. Karanasiou[2]
[1]Hellenic Naval Academy,
[2]School of Electrical and Computer Engineering, National Technical University of Athens, Greece

1. Introduction

1.1 Introduction to load time series classification

The formation of typical chronological load curves is an important tool of resolution of many problems in power systems, such as the short-term and medium-term load forecasting, the adaptation of customers' tariffs and the classification of electricity customers.
Classical indexes, like maximum power or load factor, can not describe the electricity behaviour of a customer or a power system thoroughly, as it can be comprehended from the example of Fig 1.1, where the customer of Fig 1.1(a) fatigues the power system's generators less than the customer of Fig 1.1(b) for the same peak load, load factor and power factor, because the number of the load demand changes are fewer. If an energy storage system is used, the second customer will need smaller battery system than the first one. These inferences cannot be drawn without the customers' load profiles.

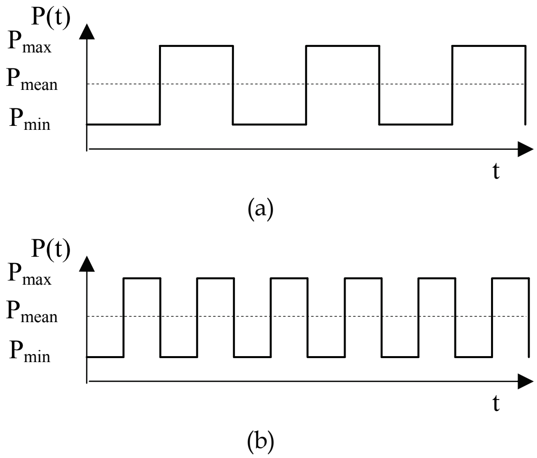


Fig. 1.1. Indicative load curves of electricity consumers with the same max load, load factor and power factor

Source: Pattern Recognition Techniques, Technology and Applications, Book edited by: Peng-Yeng Yin,
ISBN 978-953-7619-24-4, pp. 626, November 2008, I-Tech, Vienna, Austria

In the case of short-term load forecasting the use of the typical days decreases the mean absolute percentage error, especially for anomalous days (i.e. holiday periods) (Chicco et al., 2001; Lamedica et al., 1996). Through this segmentation the load forecasting models are not misled by the respective load curves of last days in which more weight is usually given. Similarly the formation of typical chronological load curves and the corresponding diachronic development can be used for medium-term load forecasting (Al - Hamadi & Soliman, 2005), so that the maintenance of the units and electric network, the fuel supply, the electrical energy imports/exports and the exploitation of the water reserves for hydrothermal scheduling can be implemented.

In a deregulated electricity market, each supplier wishes to identify his customers' electricity behaviour accurately, in order to provide them with satisfactory services at a low cost, recovering the energy and power cost and having a fair profit. So the classification of electricity customers is a necessary stage. At the same time, each consumer wants to know his electricity behaviour, in order to select the proper tariff or to apply energy efficiency measures successfully. Taking into consideration the demand-side bidding in competitive markets (Task VIII of IEA, 2002) the accurate estimation of the next day's load profile is a fundamental requirement for each large customer, so that it can find the way to minimize its electricity bill.

During the last years, a significant research effort has been focused on load curves classification regarding the short-term load forecasting of anomalous days and the clustering of the customers of the power systems. The clustering methods have been used so far are:

- the "modified follow the leader" (Chicco et al., 2003a; -,2003b; -, 2004; -, 2006),
- the self-organizing map (Beccali et al., 2004; Chicco et al., 2004; -, 2006 ; Figueiredo et al., 2003; Lamedica et al. 1996; Verdu et al., 2003),
- the k-means (Chicco et al., 2006; Figueiredo et al., 2003),
- the average and Ward hierarchical methods (Chicco et al., 2004; -, 2006; Gerber et al., 2003) and
- the fuzzy k-means (Chicco et al., 2004; -, 2006; Gerber et al., 2003; -, 2004; -, 2005).

All the above methods generally belong to pattern recognition techniques (Theodoridis & Koutroumbas, 1999). Alternatively, classification problem can be solved by using data mining (Kitayama et al., 2003; Figueiredo et al., 2005), wavelet packet transformation (Petrescu & Scutariu, 2002), frequency-domain data (Carpaneto et al., 2006), stratified sampling (Chen et al., 1997). For the reduction of the size of the clustering input data set Sammon map, principal component analysis and curvilinear component analysis have been proposed (Chicco et al., 2006).

The respective adequacy measures that are commonly used are:

- the mean index adequacy (Chicco et al., 2003a; -, 2003b; -, 2004),
- the clustering dispersion indicator (Chicco et al., 2003a; -, 2003b; -, 2004; -, 2006),
- the similarity matrix indicator (Chicco et al., 2004),
- the Davies-Bouldin indicator (Beccali et al., 2004; Chicco et al., 2003; -, 2006; Gerbec et al., 2004; -, 2005),
- the modified Dunn index (Chicco et al., 2006),
- the scatter index (Chicco et al., 2006) and
- the mean square error (Gerbec et al., 2003; -, 2004; -, 2005).

In all cases analytical chronological load curves are required, which have resulted via suitable measurements or load surveys. The use of classification methods allow us to compress data information implementing the fundamental concepts of data mining and pattern recognition.

1.2 Why should load time-series classification be realized using unsupervised pattern recognition methods? What kind of problems are we going to meet?

According to R.O.Duda, P.E. Hart, D. G. Stock (Duda et al., 2001), it is known that "*pattern recognition*" is the act of taking in raw data and making an action based on the category of the pattern. Generally any method that incorporates information from training samples in the design of a classifier employs learning. There are three kinds of learning:

- *Supervised learning*, in which a category label or cost for each pattern is provided in a training set and the sum of these patterns should be minimized using methods based on Bayesian decision theory, maximum likelihood and Bayesian parameter estimation, multilayer neural networks, probabilistic neural network etc.
- *Unsupervised learning* or clustering, where there are no any a priori category labels for patterns and the pattern recognition system forms clusters - sets of the input patterns. Different clustering algorithms, such as self organizing map, adaptive vector quantization etc, lead to different clusters.
- *Reinforcement learning*, where no desired category signal is given, but the only feedback is that the specified category is right or wrong, without saying why it is wrong.

In our case the clusters of the load time-series are unknown. We do not know either the clusters, or their number. The danger of an inappropriate representation is big. In order to realize the categorization of the load time-series and to select the proper number of clusters we are going to study the behaviour of the adequacy measures, which show us when the proper number of clusters is determined. Other basic notions in our approach are the following:

- the modification of the clustering techniques for this kind of classification problem, such as the appropriate weights initialization for the k-means and fuzzy k-means;
- the proper parameters calibration, such as the training rate of mono-dimensional SOM, in order to fit the classification needs;
- the comparison of the performance of the clustering algorithms for each one of the adequacy measures;
- the introduction of the ratio of within cluster sum of squares to between cluster variation, which is first presented for this kind of classification.

1.3 Mathematical modeling of clustering methods and adequacy measures

1.3.1 General introduction

We assume that the classification of daily load curves is necessary using the proper pattern recognition method. Generally N is defined as the population of the input vectors, which are going to be clustered. The ℓ-th input vector is symbolized as follows:

$$\vec{x}_\ell = \left(x_{\ell 1}, x_{\ell 2}, \ldots x_{\ell i}, \ldots x_{\ell d}\right)^T \tag{1.1}$$

where d is its dimension, which equals to 96 or 24, if the load measurements are taken every 15 minutes or every hour respectively. The corresponding set of vectors is given by:

$$X = \{\vec{x}_\ell : \ell = 1,\dots,N\} \tag{1.2}$$

It is worth mentioning that $x_{\ell i}$ are normalized using the upper and lower values of all elements of the original input patterns set, aiming the achievement of the best possible results after the application of clustering methods.
Each classification process makes a partition of the initial N input vectors to M clusters, which can be the typical days of the under study customer (first example) or the customer classes (second example - the second stage of the proposed methodology of (Tsekouras et al., 2007)) or the typical days of the power system (third example). The j-th cluster has a representative, which is the respective load profile and is represented by the vector of d dimension:

$$\vec{w}_j = \left(w_{j1}, w_{j2}, \dots w_{ji}, \dots w_{jd}\right)^T \tag{1.3}$$

The last vector also expresses the cluster's centre or the weight vector of neuron, if a clustering artificial neural network is used. In our case it is also called the j-th *class representative load diagram*. The corresponding set is the classes' set, which is defined by:

$$W = \{\vec{w}_k, k = 1, \dots M\} \tag{1.4}$$

The subset of input vectors $\vec{x}_\ell$, which belong to the j-th cluster, is Ω_j and the respective population of load diagrams is N_j. More specifically Ω_j is determined as follows:

$$\Omega_j = \left\{\vec{x}_\ell, \ell = 1, \dots N \;\&\; \arg\min_{\forall k'} f\left(\vec{x}_\ell, \vec{w}_{k'}\right) \to j\right\} \tag{1.5}$$

where $\Omega_j \subseteq X$ and $\arg\min_{\forall k'} f\left(\vec{x}_\ell, \vec{w}_{k'}\right)$ the corresponding criterion of classification of the *l*-th vector in the j-th cluster.
For the study and evaluation of classification algorithms the following distances' forms are defined:

1. the Euclidean distance between ℓ_1, ℓ_2 input vectors of the set X:

$$d\left(\vec{x}_{\ell_1}, \vec{x}_{\ell_2}\right) = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(x_{\ell_1 i} - x_{\ell_2 i}\right)^2} \tag{1.6}$$

2. the distance between the representative vector $\vec{w}_j$ of j-th cluster and the subset Ω_j, calculated as the geometric mean of the Euclidean distances between $\vec{w}_j$ and each member of Ω_j:

$$d\left(\vec{w}_k, \Omega_k\right) = \sqrt{\frac{1}{N_k}\sum_{\vec{x}_\ell \in \Omega_k} d^2\left(\vec{w}_k, \vec{x}_\ell\right)} \tag{1.7}$$

3. the infra-set mean distance of a set, defined as the geometric mean of the inter-distances between the members of the set, i.e. for the subset Ω_j and for the subset *W*:

$$\hat{d}\left(\Omega_k\right)=\sqrt{\frac{1}{2N_k}\sum_{\vec{x}_\ell\in\Omega_k} d^2\left(\vec{x}_\ell,\Omega_k\right)} \tag{1.8}$$

$$\hat{d}\left(W\right)=\sqrt{\frac{1}{2M}\sum_{k=1}^{M} d^2\left(\vec{w}_k,W\right)} \tag{1.9}$$

1.3.2 Adequacy measures

In order to evaluate the performance of the clustering algorithms and to compare them with each other, six different adequacy measures are applied. Their purpose is to obtain well-separated and compact clusters to make the load diagrams self explanatory. The definitions of these measures are the following:

1. *Mean square error or error function (J)* (Gerber et al., 2003), which expresses the distance of each vector from its cluster's centre with the same value of weight:

$$J=\frac{1}{N}\sum_{\ell=1}^{N} d^2\left(\vec{x}_\ell,\vec{w}_{k:\vec{x}_\ell\in\Omega_k}\right) \tag{1.10}$$

2. *Mean index adequacy (MIA)* (Chicco et al., 2003a), which is defined as the average of the distances between each input vector assigned to the cluster and its centre:

$$MIA=\sqrt{\frac{1}{M}\sum_{k=1}^{M} d^2\left(\vec{w}_k,\Omega_k\right)} \tag{1.11}$$

3. *Clustering dispersion indicator (CDI)* (Chicco et al., 2003a), which depends on the mean infra-set distance between the input vectors in the same cluster and inversely on the infra-set distance between the class representative load curves:

$$CDI=\frac{1}{\hat{d}\left(W\right)}\sqrt{\frac{1}{M}\sum_{k=1}^{M}\hat{d}^2\left(\Omega_k\right)} \tag{1.12}$$

4. *Similarity matrix indicator (SMI)* (Chicco et al., 2003b), which is defined as the maximum off-diagonal element of the symmetrical similarity matrix, whose terms are calculated by using a logarithmic function of the Euclidean distance between any kind of class representative load curves:

$$SMI=\max_{p\rangle q}\left\{\left(1-\frac{1}{\ln\left[d\left(\vec{w}_p,\vec{w}_q\right)\right]}\right)^{-1}\right\}: p,\, q=1,\ldots,\, M \tag{1.13}$$

5. *Davies-Bouldin indicator (DBI)* (Davies & Bouldin., 1979), which represents the system-wide average of the similarity measures of each cluster with its most similar cluster:

$$DBI = \frac{1}{M}\sum_{k=1}^{M}\max_{p\neq q}\left\{\frac{\hat{d}\left(\Omega_p\right)+\hat{d}\left(\Omega_q\right)}{d\left(\vec{w}_p,\vec{w}_q\right)}\right\} : p, q = 1,\ldots, M \tag{1.14}$$

6. *Ratio of within cluster sum of squares to between cluster variation (WCBCR)* (Hand et al., 2001), which depends on the sum of the distance's square between each input vector and its cluster's representative vector, as well as the similarity of the clusters' centres:

$$WCBCR = \frac{\sum_{k=1}^{M}\sum_{\vec{x}_\ell\in\Omega_k} d^2\left(\vec{w}_k,\vec{x}_\ell\right)}{\sum_{1\le q<p}^{M} d^2\left(\vec{w}_p,\vec{w}_q\right)} \tag{1.15}$$

The success of the different algorithms for the same final number of clusters is expressed by having small values of the adequacy measures. By increasing the number of clusters all the measures decrease, except of the similarity matrix indicator. An additional adequacy measure could be the number of the *dead* clusters, for which the sets are empty. It is intended to minimize this number. It is noted that in eq. (1.10)-(1.15), M is the number of the clusters without the dead ones.

1.3.3 K-means

The k-means method is the simplest hard clustering method, which gives satisfactory results for compact clusters (Duda et al., 2001). The k-means clustering method groups the set of the N input vectors to M clusters using an iterative procedure. The respective steps of the algorithm are the follows:

a. Initialization of the weights of M clusters is determined. In the classic model a random choice among the input vectors is used (Chicco et al., 2006; Figueiredo et al., 2003). In the developed algorithm the w_{ji} of the j-th centre is initialized as:

$$w_{ji}^{(0)} = a + b\cdot(j-1)/(M-1) \tag{1.16}$$

where a and b are properly calibrated parameters. Alternatively the w_{ji} is initialized as:

$$w_{ji}^{(0)} = a_i + b_i\cdot(j-1)/(M-1) \tag{1.17}$$

where $a_i = \min_{\forall j}\left(x_{ji}\right)$ and $b_i = \max_{\forall j}\left(x_{ji}\right)$.

b. During epoch t for each training vector $\vec{x}_\ell$ its Euclidean distances $d\left(\vec{x}_\ell,\vec{w}_j\right)$ are calculated for all centres. The ℓ-th input vector is put in the set $\Omega_j^{(t)}$, for which the distance between $\vec{x}_\ell$ and the respective centre is minimum, which means:

$$d\left(\vec{x}_\ell,\vec{w}_k\right) = \min_{\forall j} d\left(\vec{x}_\ell,\vec{w}_j\right) \tag{1.18}$$

c. When the entire training set is formed, the new weights of each centre are calculated as:

$$\vec{w}_j^{(t+1)} = \frac{1}{N_j^{(t)}} \sum_{\vec{x}_\ell \in \Omega_j^{(t)}} \vec{x}_\ell \tag{1.19}$$

where $N_j^{(t)}$ is the population of the respective set $\Omega_j^{(t)}$ during epoch *t*.

d. Next, the number of the epochs is increased by one. This process is repeated (return to step b) until the maximum number of epochs is used or weights do not significantly change ($\left|\vec{w}_j^{(t)} - \vec{w}_j^{(t+1)}\right| < \varepsilon$, where ε is the upper limit of weight change between sequential iterations). The algorithm's main purpose is to minimize the appropriate error function *J*. The main difference with the classic model is that the process is repeated for various pairs of (*a,b*). The best results for each adequacy measure are recorded for various pairs (*a,b*).

At the end of the execution of the algorithm the six adequacy measures are calculated, which are used for comparison reasons with the other clustering methods. The core of algorithm is executed from M_1 to M_2 neurons, because the necessary number of clusters is not known a priori, as it depends on the time period which is examined and the available number of patterns.

1.3.4 Kohonen adaptive vector quantization - AVQ

This algorithm is a variation of the *k*-means method, which belongs to the unsupervised competitive one-layer neural networks. It classifies input vectors into clusters by using a competitive layer with a constant number of neurons. Practically in each step all clusters compete each other for the winning of a pattern. The winning cluster moves its centre to the direction of the pattern, while the rest clusters move their centres to the opposite direction (supervised classification) or remain stable (unsupervised classification).

Here, we will use the last unsupervised classification algorithm. The respective steps are the following:

a. Initialization of the weights of *M* clusters is determined, where the weights of all clusters are equal to 0.5, that is $w_{ji}^{(0)} = 0.5, \forall j, i$.

b. During epoch *t* each input vector $\vec{x}_\ell$ is randomly presented and its respective Euclidean distances from every neuron are calculated. In the case of existence of bias factor λ, the respective minimization function is:

$$f_{winner\_neuron}\left(\vec{x}_\ell\right) = j : \min_{\forall j}\left(d\left(\vec{x}_\ell, \vec{w}_j\right) + \lambda \cdot N_j / N\right) \tag{1.20}$$

where N_j is the population of the respective set Ω_j during epoch *t-1*.

The weights of the winning neuron (with the smallest distance) are updated as:

$$\vec{w}_j^{(t)}\left(n+1\right) = \vec{w}_j^{(t)}\left(n\right) + \eta\left(t\right)\cdot\left(\vec{x}_\ell - \vec{w}_j^{(t)}\left(n\right)\right) \tag{1.21}$$

where *n* is the number of input vectors, which have been presented during the current epoch, and $\eta\left(t\right)$ is the learning rate according to:

$$\eta(t) = \eta_0 \cdot \exp\left(-\frac{t}{T_{\eta 0}}\right) > \eta_{\min} \tag{1.22}$$

where η_0, $\eta_{\min}$ and $T_{\eta 0}$ are the initial value, the minimum value and the time parameter respectively. The remaining neurons are unchangeable for $\vec{x}_\ell$, as introduced by the Kohonen winner-take-all learning rule (Kohonen, 1989; Haykin, 1994).

c. Next, the number of the epochs is increased by one. This process is repeated (return to step b) until either the maximum number of epochs is reached or the weights converge or the error function J does not improve, which means:

$$\left|\frac{J^{(t)} - J^{(t+1)}}{J^{(t)}}\right| < \varepsilon' \text{ for } t \geq T_{in} \tag{1.23}$$

where ε' is the upper limit of error function change between sequential iterations and the respective criterion is activated after T_{in} epochs.

The core of algorithm is executed for specific number of neurons and the respective parameters η_0, $\eta_{\min}$ and $T_{\eta 0}$ are optimized for each adequacy measure separately. This process is repeated from M_1 to M_2 neurons.

1.3.5 Fuzzy k-means

During the application of the k-mean or the adaptive vector quantization algorithm each pattern is assumed to be in exactly one cluster (hard clustering). In many cases the areas of two neighbour clusters are overlapped, so that there are not any valid qualitative results.

If we want to relax the condition of exclusive partition of an input pattern to one cluster, we should use fuzzy clustering techniques. Specifically, each input vector $\vec{x}_\ell$ does not belong to only one cluster, but it participates to every j-th cluster by a membership factor $u_{\ell j}$, where:

$$\sum_{j=1}^{M} u_{\ell j} = 1 \;\;\&\;\; 0 \leq u_{\ell j} \leq 1, \forall j \tag{1.24}$$

Theoretically, the membership factor gives more flexibility in the vector's distribution. During the iterations the following objective function is minimized:

$$J_{fuzzy} = \frac{1}{N}\sum_{j=1}^{M}\sum_{\ell=1}^{N} u_{\ell j} \cdot d^2\left(\vec{x}_\ell, \vec{w}_j\right) \tag{1.25}$$

The simplest algorithm is the fuzzy k-means clustering one, in which the respective steps are the following:

a. Initialization of the weights of M clusters is determined. In the classic model a random choice among the input vectors is used (Chicco et al., 2006; Figueiredo et al., 2003). In the developed algorithm the w_{ji} of the j-th centre is initialized by eq. (1.16) or eq. (1.17).

b. During epoch t for each training vector $\vec{x}_\ell$ the membership factors are calculated for every cluster:

$$u_{\ell j}^{(t+1)} = \frac{1}{\sum_{k=1}^{M} \frac{d\left(\vec{x}_\ell, \vec{w}_j^{(t)}\right)}{d\left(\vec{x}_\ell, \vec{w}_k^{(t)}\right)}} \quad (1.26)$$

c. Afterwards the new weights of each centre are calculated as:

$$\vec{w}_j^{(t+1)} = \frac{\sum_{\ell=1}^{N} \left(u_{\ell j}^{(t+1)}\right)^q \cdot \vec{x}_\ell}{\sum_{\ell=1}^{N} \left(u_{\ell j}^{(t+1)}\right)^q} \quad (1.27)$$

where q is the *amount of fuzziness* in the range $(1, \infty)$ which increases as fuzziness reduces.

d. Next, the number of the epochs is increased by one. This process is repeated (return to step b) until the maximum number of epochs is used or weights do not significantly change.

This process is repeated for different pairs of (a,b) and for different amounts of fuzziness. The best results for each adequacy measure are recorded for different pairs (a,b) and q.

1.3.6 Self-organizing map - SOM

The Kohonen SOM (Kohonen, 1989; SOM Toolbox for MATLAB 5, 2000; Thang et al., 2003) is a topologically unsupervised neural network that projects a d-dimensional input data set into a reduced dimensional space (usually a mono-dimensional or bi-dimensional map). It is composed of a predefined grid containing $M_1 \times M_2$ d-dimensional neurons $\vec{w}_k$, which are calculated by a competitive learning algorithm that updates not only the weights of the winning neuron, but also the weights of its neighbour units in inverse proportion of their distance. The neighbourhood size of each neuron shrinks progressively during the training process, starting with nearly the whole map and ending with the single neuron.

The process of algorithm is described by the following stages:

- *Initialization stage*. The weights of the neural network are initialized connecting the neurons of the input layer with the map neurons.
- *Competition stage*. For each input pattern the map neurons calculate the corresponding value of the competition function, where the neuron with the biggest value is the winner.
- *Collaboration stage*. The winner neuron determines the territorial area of topological neighbourhood, providing the subbase for the collaboration between the neighbouring neurons.
- *Weights' adaptation stage*. The neurons that belong in the winning neighbourhood adapt their weights of winner-neuron, so that its response will be strengthened during the presentation of a training input pattern.

The training of SOM is divided to two phases:

- *rough ordering*, with high initial learning rate, large radius and small number of epochs, so that neurons are arranged into a structure which approximately displays the inherent characteristics of the input data,

- *fine tuning*, with small initial learning rate, small radius and higher number of training epochs, in order to tune the final structure of the SOM.

The transition of the rough ordering phase to the fine tuning one is happened after T_{s0} epochs.

More analytically, the respective steps of the SOM algorithm are the following:

a. The shape and the number of neurons of the SOM's grid are defined and the initialization of the respective weights is determined. Specifically, in the case of the mono-dimensional SOM the weights can be given by (a) $w_{ki} = 0.5, \forall k, i$, (b) the random initialization of each neuron's weight, (c) the random choice of the input vectors for each neuron. In the case of the bi-dimensional SOM the additional issues that must be solved, are the shape, the population of neurons and their respective arrangement. The rectangular shape of the map is defined by rectangular or hexagonal arrangement of neurons, as it is presented in Fig. 1.2. The population of the neurons is recommended to be $5\times\sqrt{N}$ to $20\times\sqrt{N}$ (Chicco et al., 2004; SOM Toolbox for MATLAB 5, 2000; Thang et al., 2003;). The height/width ratio M_1/M_2 of the rectangular grid can be calculated as the ratio between the two major eigenvalues λ_1, λ_2 of the covariance matrix of the input vectors set (with $\lambda_1 > \lambda_2$). The initialization of the neurons can be a linear combination of the respective eigenvectors $\vec{e}_1$ and $\vec{e}_2$ of the two major eigenvalues or can be equal to 0.5. It is reminded that the element s_{k_1,k_2} of the covariance matrix of the input vectors set is given by:

$$s_{k_1,k_2} = \sum_{\ell=1}^{N} \left(x_{k_1\ell} - \overline{x}_{k_1}\right)\cdot\left(x_{k_2\ell} - \overline{x}_{k_2}\right) \Big/ \left(N-1\right) \tag{1.28}$$

where $\overline{x}_{k_1}$ is the mean value of the respective k_1 dimension of all input patterns.

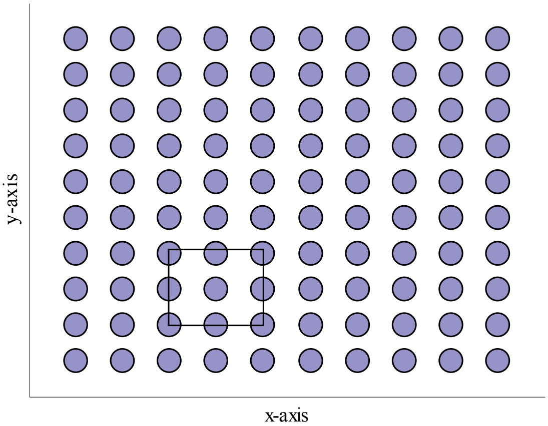
b. The SOM training commences by first choosing an input vector $\vec{x}_\ell$, at t epoch, randomly from the input vectors' set. The Euclidean distances between the n-th presented input pattern $\vec{x}_\ell$ and all $\vec{w}_k$ are calculated, so as to determine the winning neuron i' that is closest to $\vec{x}_\ell$ (competition stage). The j-th reference vector is updated (weights' adaptation stage) according to:

$$\vec{w}_j^{(t)}\left(n+1\right) = \vec{w}_j^{(t)}\left(n\right) + \eta\left(t\right)\cdot h_{i'j}\left(t\right)\cdot\left(\vec{x}_\ell - \vec{w}_j^{(t)}\left(n\right)\right) \tag{1.29}$$

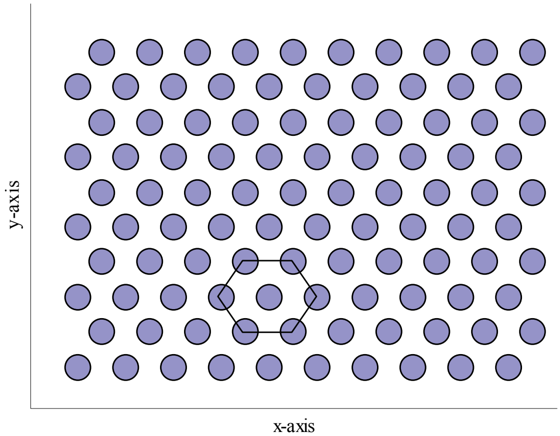
where $\eta\left(t\right)$ is the learning rate according to:

$$\eta\left(t\right) = \eta_0 \cdot \exp\left(-\frac{t}{T_{\eta_0}}\right) > \eta_{\min} \tag{1.30}$$

with η_0, $\eta_{\min}$ and T_{η_0} representing the initial value, the minimum value and the time parameter respectively. During the rough ordering phase $\eta_r, T_{\eta 0}$ are the initial value



a. Rectangular arrangement



b. Hexagonal arrangement

Fig. 1.2. Arrangement of bi-dimensional self-organized map 10x10

and the time parameter respectively, while during the fine tuning phase the respective values are $\eta_f, T_{\eta 0}$. The $h_{i'j}(t)$ is the neighbourhood symmetrical function, that will activate the *j* neurons that are topologically close to the winning neuron *i′*, according to their geometrical distance, who will learn from the same $\vec{x}_\ell$ (collaboration stage). In this case the Gauss function is proposed:

$$h_{i'j}(t) = \exp\left[-\frac{d_{i'j}^2}{2 \cdot \sigma^2(t)}\right] \tag{1.31}$$

where $d_{i'j} = \left\| \vec{r}_{i'} - \vec{r}_j \right\|$ is the respective distance between i' and j neurons, $\vec{r}_j = \left(x_{j,} y_j \right)$ are the respective co-ordinates in the grid, $\sigma(t) = \sigma_0 \cdot \exp\left(-t / T_{\sigma_0} \right)$ is the decreasing neighbourhood radius function where σ_0 and T_{σ_0} are the respective initial value and time parameter of the radius respectively.

c. Next, the number of the epochs is increased by one. This process is repeated (return to step b) until either the maximum number of epochs is reached or the index I_s gets the minimum value (SOM Toolbox for MATLAB 5, 2000):

$$I_s(t) = J(t) + ADM\left(t\right) + TE(t) \tag{1.32}$$

where the quality measures of the optimum SOM are based on the quantization error J - given by (1.10)-, the topographic error TE and the average distortion measure ADM. The topographic error measures the distortion of the map as the percentage of input vectors for which the first i_1' and second i_2' winning neuron are not neighbouring map units:

$$TE = \sum_{\ell=1}^{N} neighb\left(i_1', i_2'\right) \Big/ N \tag{1.33}$$

where, for each input vector, $neighb\left(i_1', i_2'\right)$ equals to 1, if i_1' and i_2' neurons are not neighbours, either 0. The average distortion measure is given for the t epoch by:

$$ADM\left(t\right) = \sum_{\ell=1}^{N} \sum_{j=1}^{M} h_{i' \to \vec{x}_\ell, j}\left(t\right) \cdot d^2\left(\vec{x}_\ell, \vec{w}_j\right) \Big/ N \tag{1.34}$$

This process is repeated for different parameters of $\sigma_0, \eta_f, \eta_r, T_{\eta 0}, T_{\sigma 0}$ and T_{s_0}. Alternatively, the multiplicative factors ϕ and ξ are introduced -without decreasing the generalization ability of the parameters' calibration:

$$T_{s_0} = \phi \cdot T_{\eta_0} \tag{1.35}$$

$$T_{\sigma_0} = \xi \cdot T_{\eta_0} \Big/ \ln \sigma_0 \tag{1.36}$$

The best results for each adequacy measure are recorded for different parameters $\sigma_0, \eta_f, \eta_r, T_{\eta 0}, \phi$ and ξ.

In the case of the bi-dimensional map, the immediate exploitation of the respective clusters is not a simple problem. We can exploit the map either through human vision or applying a second simple clustering method. According to Chicco et al., (2002), the simple k-mean method was used, while, here, the proposed k-mean method with initialization by eq. (1.16) is used. Practically, the neurons of the map sustain a new data compression from which the final classification of the input patterns is concluded.

1.3.7 Hierarchical agglomerative algorithms

Hierarchical algorithms have a different philosophy compared to the aforementioned algorithms. Instead of producing a single clustering, they produce a hierarchy of clustering.

Agglomerative algorithms are based on matrix theory (Theodoridis & Koutroumbas, 1999). The input is the $N \times N$ dissimilarity matrix P_0. At each level t, when two clusters are merged into one, the size of the dissimilarity matrix P_t becomes $(N-t)\times(N-t)$. Matrix P_t is obtained from P_{t-1} by deleting the two rows and columns that correspond to the merged clusters and adding a new row and a new column that contain the distances between the newly formed cluster and the old ones. The distance between the newly formed cluster C_q (the result of merging C_i and C_j) and an old cluster C_s is determined as:

$$d\left(C_q,C_s\right)=f\left(d\left(C_i,C_s\right),d\left(C_j,C_s\right),d\left(C_i,C_j\right)\right) \tag{1.37}$$

Alternatively eq. (1.37) is written as:

$$d\left(C_q,C_s\right)=a_i\cdot d\left(C_i,C_s\right)+a_j\cdot d\left(C_j,C_s\right)+b\cdot d\left(C_i,C_j\right)+c\cdot\left|d\left(C_i,C_s\right)-d\left(C_j,C_s\right)\right| \tag{1.38}$$

where a_i, a_j, b and c correspond to different choices of the dissimilarity measure.

The basic algorithms, which are going to be used in our case, are:

- the *single link* algorithm (*SL*) -it is obtained from (1.38) for a_i=a_j=0.5, b=0 and $c=-0.5$:

$$d\left(C_q,C_s\right)=\min\left\{d\left(C_i,C_s\right),d\left(C_j,C_s\right)\right\}=\frac{1}{2}\cdot d\left(C_i,C_s\right)+\frac{1}{2}\cdot d\left(C_j,C_s\right)-\frac{1}{2}\cdot\left|d\left(C_i,C_s\right)-d\left(C_j,C_s\right)\right| \tag{1.39}$$

- the *complete link* algorithm (*CL*) -it is obtained from (1.38) for a_i=a_j=0.5, b=0 and $c=0.5$:

$$d\left(C_q,C_s\right)=\max\left\{d\left(C_i,C_s\right),d\left(C_j,C_s\right)\right\}=\frac{1}{2}\cdot d\left(C_i,C_s\right)+\frac{1}{2}\cdot d\left(C_j,C_s\right)+\frac{1}{2}\cdot\left|d\left(C_i,C_s\right)-d\left(C_j,C_s\right)\right| \tag{1.40}$$

- the *unweighted pair group method average* algorithm (*UPGMA*):

$$d\left(C_q,C_s\right)=\frac{n_i\cdot d\left(C_i,C_s\right)+n_j\cdot d\left(C_j,C_s\right)}{n_i+n_j} \tag{1.50}$$

  where n_i and n_j - are the respective members' populations of clusters C_i and C_j.
- the *weighted pair group method average* algorithm (*WPGMA*):

$$d\left(C_q,C_s\right)=\frac{1}{2}\cdot\left\{d\left(C_i,C_s\right)+d\left(C_j,C_s\right)\right\} \tag{1.42}$$

- the *unweighted pair group method centroid* algorithm (*UPGMC*):

$$d^{(1)}\left(C_q,C_s\right)=\frac{\left(n_i\cdot d^{(1)}\left(C_i,C_s\right)+n_j\cdot d^{(1)}\left(C_j,C_s\right)\right)}{\left(n_i+n_j\right)}-n_i\cdot n_j\cdot\frac{d^{(1)}\left(C_i,C_j\right)}{\left(n_i+n_j\right)^2} \tag{1.43}$$

  where $d^{(1)}\left(C_q,C_s\right)=\left\|\vec{w}_q-\vec{w}_s\right\|^2$ and $\vec{w}_q$ is the representative centre of the q-th cluster according to the following equation (which is similar to(1.39)):

$$\vec{w}_q = \frac{1}{n_q} \cdot \sum_{\vec{x}_v \in C_q} \vec{x}_v \tag{1.44}$$

- the *weighted pair group method centroid* algorithm (*WPGMC*):

$$d^{(1)}\left(C_q, C_s\right) = \frac{1}{2} \cdot \left(d^{(1)}\left(C_i, C_s\right) + d^{(1)}\left(C_j, C_s\right)\right) - \frac{1}{4} \cdot d^{(1)}\left(C_i, C_j\right) \tag{1.45}$$

- the *Ward or minimum variance* algorithm (*WARD*):

$$d^{(2)}\left(C_q, C_s\right) = \frac{\left(n_i + n_s\right) \cdot d^{(2)}\left(C_i, C_s\right) + \left(n_j + n_s\right) \cdot d^{(2)}\left(C_j, C_s\right) - n_s \cdot d^{(2)}\left(C_i, C_j\right)}{\left(n_i + n_j + n_s\right)} \tag{1.46}$$

where:

$$d^{(2)}\left(C_i, C_j\right) = \frac{n_i \cdot n_j}{n_i + n_j} \cdot d^{(1)}\left(C_i, C_j\right) \tag{1.47}$$

It is noted that in each level t the respective representative vectors are calculated by eq.(1.44).

The respective steps of each algorithm are the following:

a. *Initialization*: The set of the remaining patterns $\Re_0$ for zero level ($t = 0$) is the set of the input vectors X. The similarity matrix $P_0 = P\left(X\right)$ is determined. Afterwards t increases by one (t=t+1).
b. During level t clusters C_i and C_j are found, for which the minimization criterion is satisfied $d\left(C_i, C_j\right) = \min_{r,s=1,\ldots,N, r\neq s} d\left(C_r, C_s\right)$.
c. Then clusters C_i and C_j are merged into a single cluster C_q and the set of the remaining patterns $\Re_t$ is formed as: $\Re_t = \left(\Re_{t-1} - \left\{C_i, C_j\right\}\right) \cup \left\{C_q\right\}$.
d. The construction of the dissimilarity matrix P_t from P_{t-1} is realized by applying eq.(1.37).
e. Next, the number of the levels is increased by one. This process is repeated (return to step b) until the remaining patterns $\Re_{N-1}$ is formed and all input vectors are in the same and unique cluster.

It is mentioned that the number of iterations is determined from the beginning and it equals to the number of input vectors decreased by 1 (N-1).

1.4 A pattern recognition methodology for evaluation of load profiles and typical days of large electricity customers

1.4.1 General description of the proposed methodology

The classification of daily chronological load curves of one customer is achieved by means of the pattern recognition methodology, as shown in Fig. 1.3.

The main steps are the following (Tsekouras et al., 2008):

a. *Data and features selection*: Using electronic meters, the active and reactive energy values are registered (in kWh and kvarh) for each time period in steps of 15 minutes, 1 hour, etc. The daily chronological load curves are determined for the study period.

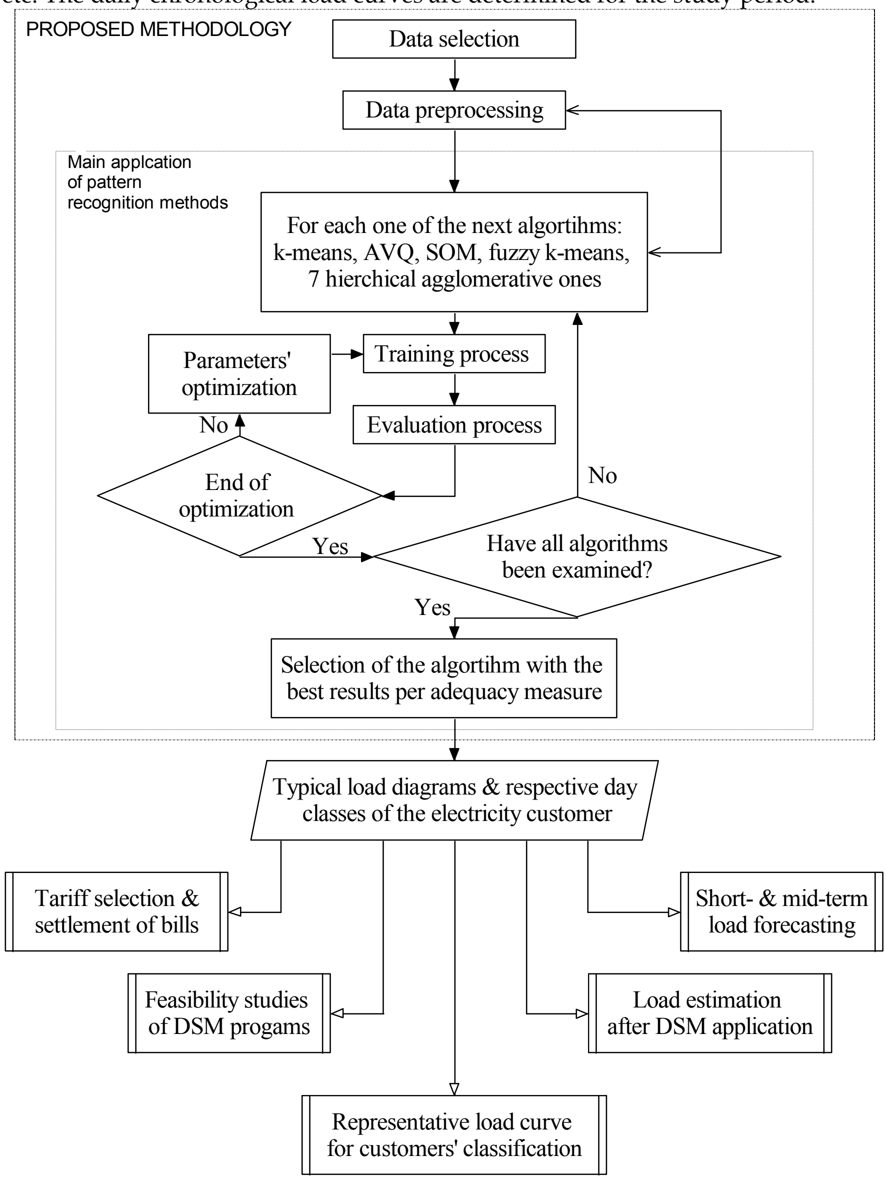


Fig. 1.3. Flow diagram of pattern recognition methodology for the classification of daily chronological load curves of one large electricity customer

b. *Data preprocessing:* The load diagrams of the customer are examined for normality, in order to modify or delete the values that are obviously wrong (*noise suppression*). If it is necessary, a preliminary execution of a pattern recognition algorithm is carried out, in order to track bad measurements or networks faults, which will reduce the number of the useful typical days for a constant number of clusters, if they remain uncorrected. In future, a filtering step can be added using principal component analysis, Sammon map, and curvilinear component analysis (Chicco et al., 2006), for the reduction of the load diagrams dimensions.

c. *Main application of pattern recognition methods*: For the load diagrams of the customer, a number of clustering algorithms (k-means, adaptive vector quantization, self organized map, fuzzy k-means and hierarchical clustering) is applied. Each algorithm is trained for the set of load diagrams and evaluated according to six adequacy measures. The parameters of the algorithms are optimized, if it is necessary. The developed methodology uses the clustering methods that provide the most satisfactory results. It should be noted that conventional methods, like statistical tools, supervised techniques, etc., cannot be used, because the classification of the typical days must be already known.

1.4.2 Application of the proposed methodology to a medium voltage customer

1.4.2.1 General

The developed methodology was analytically applied on one medium voltage industrial paper mill customer of the Greek distribution system. The data used are 15 minutes load values for a period of ten months in 2003. The respective set of the daily chronological curves has 301 members. Nine curves were rejected through data pre-processing, while the remaining 292 diagrams were used by the aforementioned clustering methods. The last diagrams are registered in Fig. 1.4 and Fig. 1.5, in which the load variability is also presented. The load behaviour is significantly decreased during holiday time. The mean load demand is 6656 kW and the peak load demand is 9469 kW during the period under study.

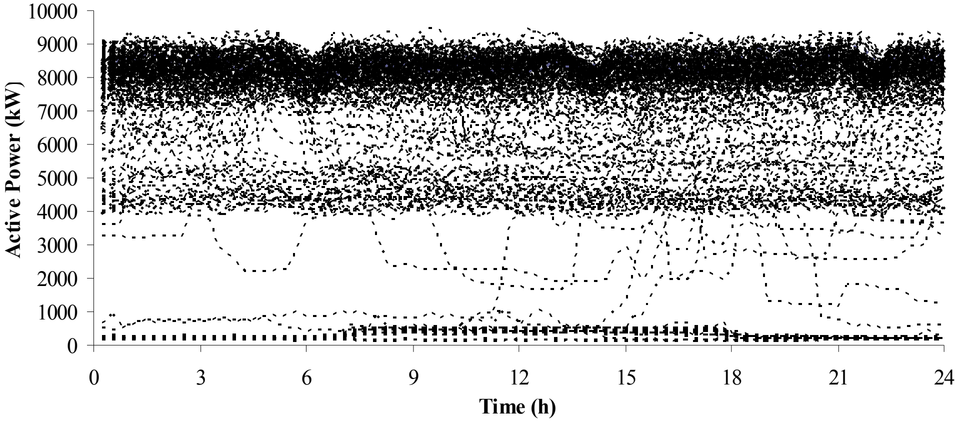


Fig. 1.4. Daily chronological 15-minutes load diagrams for a set of 292 days for the industrial medium voltage customer for each day (February – November 2003)

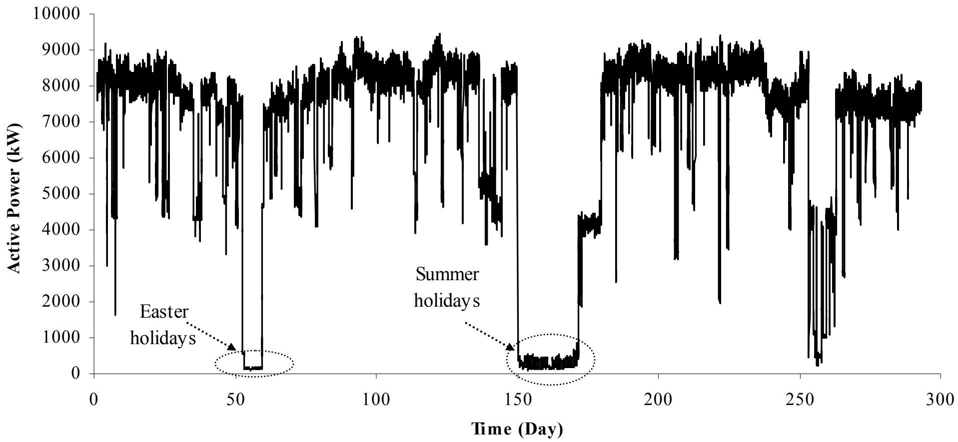


Fig. 1.5. Chronological 15-minutes load diagram for a set of 292 days for the industrial medium voltage customer for the time period under study (February-November 2003)

The main goal of the application of this methodology is the representation of the load behaviour of the customer with typical daily load chronological diagrams. This is achieved through the following steps:

- The calibration of the parameters of each clustering method is realized for every adequacy measure separately and the performance for different number of clusters is registered.
- The clustering models are compared to each other using the six adequacy measures, the behaviour of these measures is studied and the appropriate number of the clusters is defined.

The representative daily load chronological diagrams of the customer are calculated for the best clustering techniques and the proposed number of clusters.

1.4.2.2 Application of the k-means

The proposed model of the *k*-means method (*k*-means-scenario 1 with the weights initialization based on eq.(1.16)) is executed for different pairs (*a*,*b*) from 2 to 25 clusters, where a={0.1,0.11,…,0.45} and $a+b$={0.54,0.55,…,0.9}. For each cluster, 1332 different pairs (*a*,*b*) are checked. The best results for the 6 adequacy measures do not refer to the same pair (*a*,*b*). The second model of the *k*-means method (*k*-means-scenario 2) is based on eq. (1.17) for the weights initialization. The third model (*k*-means-scenario 3) is the classic one with the random choice of the input vectors during the centres' initialization. For the classic *k*-means model, 100 executions are carried out and the best results for each index are registered. In Fig. 1.6, it is obvious that the proposed k-means is superior to the other two scenarios of *k*-means. The superiority of the proposed model applies in all cases of neurons.

A second advantage comprises the convergence to the same results for the respective pairs (*a*,*b*), which cannot be reached using the classic model.