1

A Clustering Approach for System Monitoring: an Application to Electric Power Production

Alzennyr Da Silva

BILab (Telecom ParisTech and EDF R&D Common Laboratory)
INFRES-Telecom ParisTech, 46 rue Barrault, 75013 Paris - France
alzennyr.gomes@telecom-paristech.fr

Yves Lechevallier
Project AxIS-INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105
78153 Le Chesnay Cedex - France, yves.lechevallier@inria.fr

Redouane Seraoui
EDF R&D-STEP, 6 Quai Wattier, 78400 Chatou - France
redouane.seraoui@edf.fr

CONTENTS

Progressive advances in hardware and software technologies have enabled the production and storage of system monitoring data streams in a wide range of fields (e.g. telecommunications, sensor networks, etc.). Traditional clustering methods are unable to deal with data of such a voluminous and dynamic nature. In this chapter, we propose an efficient clustering approach for monitoring massive time-changing data streams. This work considers a real case study on condition monitoring data streams of an electric power plant provided by EDF.

1.1 Introduction

Clustering is one of the most popular tasks in knowledge discovery and is applied in various domains (e.g. data mining, pattern recognition, computer vision, etc). Clustering methods seek to organize a set of items into clusters such that items within a given cluster have a high degree of similarity, whereas items belonging to different clusters have a high degree of dissimilarity. Partitioning clustering methods [6, 7, 9] aim to obtain a single partition of the input data into a fixed number of clusters. Such methods often look for a partition that optimizes (usually locally) an adequacy criterion function. Clustering techniques have been widely studied across several disciplines, but only a few of the techniques developed scale to support clustering of very large time-changing data streams [10][12]. The major challenge in clustering of evolving data is to handle cluster evolution: new clusters may appear, old ones may disappear, merge or split over time.

In this chapter, we propose a clustering approach based on non-overlapping windows to monitor system and detect changes in evolving data. In order to measure the level of changes, our approach uses two external evaluation indices based on the clustering extension. In order to deal with the cluster follow-up problem, the proposed approach uses the clustering extension, i.e., a rough membership function, to match clusters in subsequent time periods.

This chapter is organized as follows. Section 1.2 presents the related work. Section 1.3 presents our clustering approach to monitor evolving data streams. Section 1.4 describes the experiments carried out on real data from electric power production. The conclusion and future work are presented in Section 1.5.

1.2 Related Work

Over the past few years, a number of clustering algorithms for data streams have been put forth [10]. Five different algorithms for clustering data streams are surveyed in [12].

One of the earliest and best known clustering algorithms for data streams is BIRCH [18]. BIRCH is a heuristic that computes a pre-clustering of the data into so-called clustering feature vectors (micro clustering) and then clusters this pre-clustering using an agglomerative bottom-up technique (macro clustering). BIRCH does not perform well because it uses the notion of radius or diameter to control the boundary of a cluster. STREAM [15] is the next main method which has been designed specially for data streams clustering. This method processes data streams in batches of points represented by weighted centroids which are recursively clustered until k clusters are formed. The main

limitation of BIRCH and STREAM is that old data points are as important as new data points.

CluStream [1] is based on BIRCH. It uses the clustering feature vectors to represent clusters and expands on this concept by adding temporal features. This method takes snapshots at different timestamps, favoring the most recent data. A snapshot corresponds to q (where q depends on the memory available) micro-clusters stored at particular moments in the stream. One weakness of the algorithm is the high number of parameters which depend on the nature of the stream and the arrival speed of elements.

StreamSamp [4] is based on CluStream and combines a memory-based reduction method and a tilted window based reduction module. In StreamSamp, summaries with a constant size are maintained. The summaries cover time periods of varying sizes: shorter for the present and longer for the distant past. StreamSamp has the advantage of being fast on building the data streams summary. However, its accuracy degrades over time because old elements increase in weight for a given sample size.

TECNO-STREAMS [14] is an immune system inspired single pass method to cluster noisy data streams. The system continuously learns and adapts to new incoming patterns. In [13], the authors extended their work to the categorical domain. They present a case study on tracking evolving topic trends in textual data streams including an external data describing an ontology of the web content.

MONIC[16] is a framework for modelling and tracking cluster transitions over time. In this framework, an offline clustering is applied periodically on an accumulating data set and a weighting based scheme is used for processing past data. As the framework operates over an accumulating dataset and no strategy to summarize data is applied, the performance of the system can decay. Moreover, if the accumulating dataset assumes huge dimensions it may not fit the available physical memory.

1.3 Clustering Approach for System Monitoring

The clustering process is started by spliting the input data stream into batches of equal size. A data batch is also referenced as a *window*.

Individuals within a batch are chronologically sorted according to their arrival date. Let $H^{(1)}$ be the batch containing the first incoming data. Let's also consider $H^{(2)}$ the next batch containing newer data over time. $H^{(1)}$ and $H^{(2)}$ are thus sequential batches with no overlapping area.

The next step consists in applying a clustering algorithm on data of each batch. Let $R^{(t-1)} = \{r_1^{(t-1)}, \ldots, r_c^{(t-1)}, \ldots, r_k^{(t-1)}\}$ be the set of representatives (prototypes) of each cluster obtained by the clustering on batch $H^{(t-1)}$. The next step consists in assigning data in batch $H^{(t)}$ to the prototypes

of $R^{(t-1)}$, what defines a first partition $P_1^{(t)}$ containing $k^{(t-1)}$ clusters. After, we apply the same clustering algorithm on data in $H^{(t)}$ defining a second partition $P_2^{(t)}$ with $k^{(t)}$ clusters represented by the set of prototypes $R^{(t)} = \{r_1^{(t)}, \ldots, r_c^{(t)}, \ldots, r_k^{(t)}\}$. This process is repeated over all two-by-two batches. A schema of the approach described is illustrated in Figure 1.1. It is important to note that the proposed approach is totally independent of the clustering algorithm used. The only requirement is that the algorithm should furnish a prototype for each cluster.

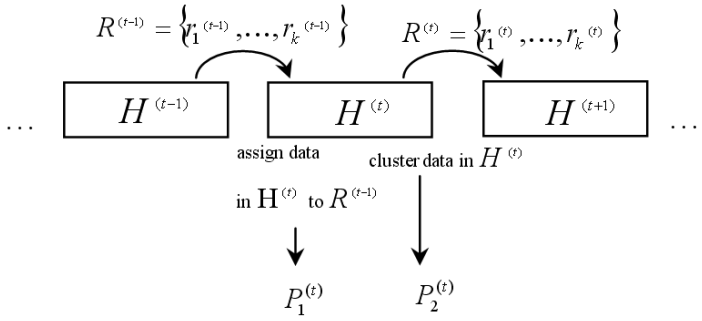

FIGURE 1.1
Schema of the clustering approach for system monitoring.

The main idea behind our change detection strategy is explained as follows. For a same data batch $H^{(t)}$, the partition $P_1^{(t)}$ is influenced by the clustering structure generated from data in the previous batch $H^{(t-1)}$. This partition will thus reflect the segmentation of the data in $H^{(t)}$ according to older patterns from $H^{(t-1)}$. On the other hand, the partition $P_2^{(t)}$ receives no influence from the past and only takes into account individuals in batch $H^{(t)}$. This partition will thus reflect the organization of current data. The occurrence of changes between two subsequent data batches can be detected by comparing partitions $P_1^{(t)}$ and $P_2^{(t)}$. In other words, if a change takes place, it will be caught by the degree of disagreement between two partitions from a same data batch.

1.3.1 Partition-Comparison Criteria

The entry n_{ij} in each cell of Table 1.1 represents the number of individuals that are in both cluster i of $P_1^{(t)}$ and cluster j of $P_2^{(t)}$. The term $n_{i.}$ is the number of individuals in cluster i of $P_1^{(t)}$ and $n_{.j}$ is the number of individuals in cluster j of $P_1^{(t)}$.

The F-measure [17] is the harmonic mean of *Recall* and *Precision* (cf. equation 1.1). This measure seeks the best match of a cluster in partition $P_1^{(t)}$

clusters of $P_1^{(t)}$ \ clusters of $P_2^{(t)}$	1	$\cdots$	j	$\cdots$	$k^{(t)}$	
1	n_{11}	$\cdots$	n_{1j}	$\cdots$	$n_{1k^{(t)}}$	$n_{1.}$
$\vdots$						
i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	$n_{ik^{(t)}}$	$n_{i.}$
$\vdots$						
$k^{(t-1)}$	$n_{k^{(t-1)}1}$	$\cdots$	$n_{k^{(t-1)}j}$	$\cdots$	$n_{k^{(t-1)}k^{(t)}}$	$n_{k^{(t-1)}.}$
	$n_{.1}$		$n_{.j}$		$n_{.k^{(t)}}$	$n_{..} = n$

for a cluster in partition $P_2^{(t)}$. The higher the values of *Recall* and *Precision*, the more similar are the clusters compared.

$$F = \sum_{i=1}^{k^{(t-1)}} \frac{n_{i.}}{n} max_{j=1,\ldots,k^{(t)}} \frac{2Recall(i,j)Precision(i,j)}{Recall(i,j) + Precision(i,j)} \tag{1.1}$$

where:

$$Recall(i,j) = \frac{n_{ij}}{n_{i.}} \text{ and } Precision(i,j) = \frac{n_{ij}}{n_{.j}}$$

The Corrected Rand index [8] (cf. equation 1.2) - CR, for short - has its values corrected for chance agreement. The CR index applies a global analysis of the two partitions compared.

$$\text{CR} = \frac{\sum_{i=1}^{k^{(t-1)}} \sum_{j=1}^{k^{(t)}} \binom{n_{ij}}{2} - \binom{n}{2}^{-1} \sum_{i=1}^{k^{(t-1)}} \binom{n_{i.}}{2} \sum_{j=1}^{k^{(t)}} \binom{n_{.j}}{2}}{\frac{1}{2}[\sum_{i=1}^{k^{(t-1)}} \binom{n_{i.}}{2} + \sum_{j=1}^{k^{(t)}} \binom{n_{.j}}{2}] - \binom{n}{2}^{-1} \sum_{i=1}^{k^{(t-1)}} \binom{n_{i.}}{2} \sum_{j=1}^{k^{(t)}} \binom{n_{.j}}{2}} \tag{1.2}$$

The F-measure takes a value in the range $[0, 1]$ and the CR index takes a value in the range $[-1, +1]$. For both indices, value 1 indicates a perfect agreement between the partitions compared, whereas values near 0 correspond to cluster agreements found by chance.

1.3.2 Cluster Follow-up

Our approach incrementally matches clusters in subsequent data batches and characterizes their evolution over time. Cluster matches are verified by means of *Recall* (cf. equation 1.1). A cluster survives if it conserves a minimum of its original individuals (*survivalThreshold*) from a data batch to the next one over time. A cluster splits if it conserves a minimum of its original individuals (*splitThreshold*) in the new clusters it originates in the next batch.

Definition 1 *A cluster i of $P_1^{(t)}$ survives if there is a cluster j in $P_2^{(t)}$ such as $Recall(i,j) >= survivalThreshold$.*

Definition 2 *The cluster j of $P_2^{(t)}$ that matches the cluster i of $P_1^{(t)}$ is the survivor cluster that conserves the maximum number of individuals of i, i.e., $arg \max_{j \in (1,\cdots,K^{(t)})} Recall(i,j)$.*

Definition 3 *A cluster i of $P_1^{(t)}$ splits if there are two or more clusters j of $P_2^{(t)}$ such that $Recall(i,j) >= splitThreshold$.*

Definition 4 *A cluster i of $P_1^{(t)}$ disappears if it neither splits nor has a match in $P_2^{(t)}$.*

Definition 5 *A cluster i fuses with one or more cluster of $P_1^{(t)}$ when it matches a cluster j of $P_2^{(t)}$ which is also a match for at least another cluster $i^{'}$ of $P_1^{(t)}$, with $i^{'} \neq i$.*

Definition 6 *A cluster j of $P_2^{(t)}$ is a new cluster if it is neither a match nor a son of a cluster of $P_1^{(t)}$.*

The clustering algorithm applied in our approach to automatically discover the number of clusters is described as follows. First, a Self Organizing Map (SOM)[11] initialized by a Principal Component Analysis (PCA) [5] is used to cluster data in each batch. The competitive layer of the SOM contains 200 neurons organized in a rectangular topology. Secondly, a Hierarchical Agglomerative Clustering (HAC) using the Ward criterion is applied on the final prototypes obtained by the SOM. The resulting dendrogram is cut according to the elbow criterion[2].

1.3.3 Cluster Description

In order to determine the most distinguishing variables of a cluster, we apply the CRT measure [3] defined according to equation 1.3.

$$CRT(j,C) = \frac{B_c^j}{B_c} \text{ where } B_c^j = \mu_c d^2(r_c^j, r^j) \text{ and } B_c = \mu_c d^2(r_c, r) \qquad (1.3)$$

The CRT is based on the between-cluster inertia (B) of a partition with k clusters. This measure computes the relative contribution of variable j and cluster C to the global between-cluster inertia. In equation 1.3, B_c represents the contribution of cluster C to the global between-cluster inertia and B_c^j is the contribution of variable j to B_c, μ_c is the weight of cluster C, r_c is the representative (prototype) of cluster C, r is the gravity centre of data and d is a dissimilarity distance. Variables with greater values of CRT are the ones that better describe the cluster.

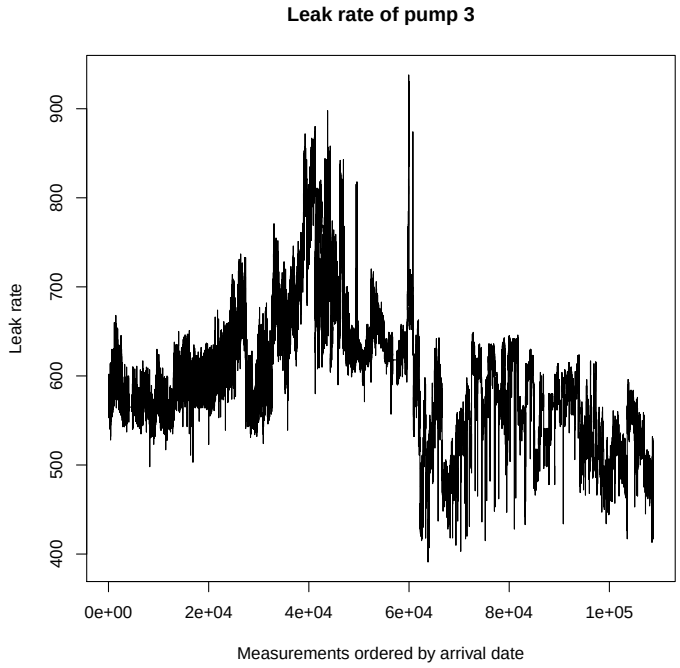

FIGURE 1.2
Leak rate of pump 3.

1.4 Experiments

We analyzed data streams describing the working phases of a nuclear power plant composed of 4 reactor coolant pumps over 5 years (from 2001 to 2005). Each pump is individually monitored by 10 sensors capturing different measurements (temperature, pressure, etc.). The global system (the reactor coolant circuit) is monitored by 7 sensors describing overall operation of the 4 pumps within the primary circuit. All sensors capture measurements at the same timestamp each 15 minutes. The data streams are then described by 10 local variables (P_{ij} with $i \in \{1, \ldots, 4\}$ and $j \in \{1, \ldots, 10\}$ represents variable j of pump i) describing each Reactor Coolant Pump and 7 global variables (referenced as G_j with $j \in \{1, \ldots, 5\}$, D and T) describing the Reactor Coolant Circuit.

We concentrated our experiments on data streams from pump 3 (chosen by a domain expert according to the leak rate parameter, cf. Figure 1.2) and

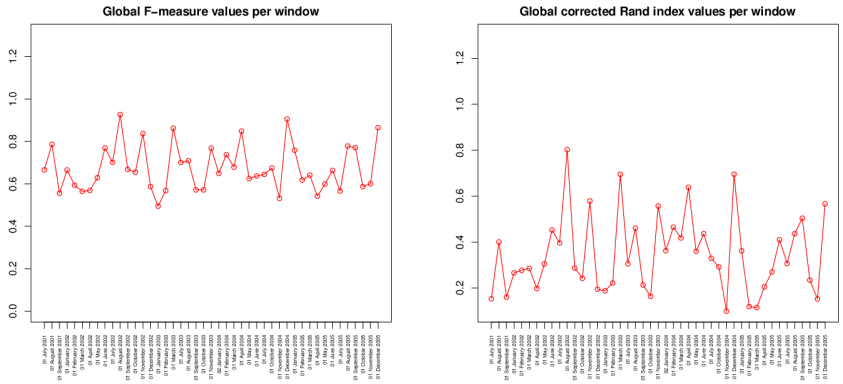

FIGURE 1.3
Values obtained by the F-measure (on the left) and by the CR index (on the right).

applied a temporal partition of the data in time window with length equal to one month, $survivalThreshold = 70\%$ and $splitThreshold = 30\%$.

The values obtained by the two partition-comparison criteria (F-measure and CR index) are presented in the graphics of Figure 1.3. The most unstable periods are the ones corresponding to the lower values of the F-measure and CR index. Notice that changes are detected by the CR index in greater widths than the F-measure. The CR index provides a global measurement based on the whole set of clusters in the two partitions compared, and it is thus more sensitive to changes.

The changes undergone by clusters throughout the analyzed months are summarized by the graphics in Figure 1.4. The total number of clusters discovered by our approach varies between 3 and 8 (cf. graphic *Total clusters* of Figure 1.4).

Let's consider consecutive months presenting great changes, for example *January 2003* compared to its previous month *December 2002* (cf. Figure 1.3: *f-measure=0.49* and *CR=0.18*). Indeed, both months present a total of 5 clusters (cf. graphic *Total clusters* of Figure 1.4) and one would expect no changes to take place. However, looking at the transition between these two months in more details, our clustering approach reveals that clusters 1, 2 and 3 of *December 2002* have no correspondent cluster in *January 2003*, they are thus considered as having disappeared. Clusters 1, 2 and 4 of *January 2003* are new clusters. Finally, cluster 4 and 5 of *December 2002* match clusters 3 and 5 of *January 2003*, respectively. Details concerning most significant variables and their prototype values within these months are given in Table 1.2.

Let's now consider an example of consecutive months presenting few

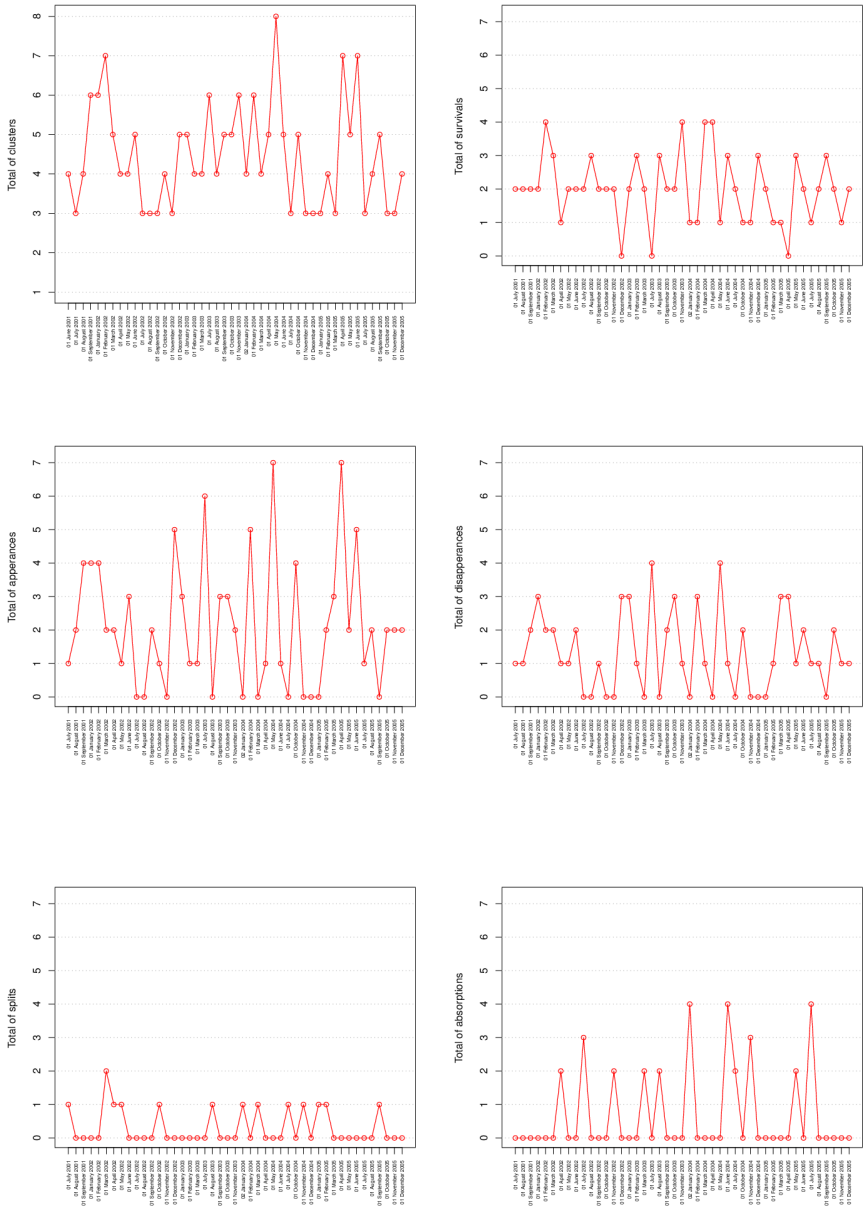

FIGURE 1.4
Summary of changes undergone by clusters over subsequent temporal data windows.

Month	Cluster	Most significant variables (and their values)
December 2002	1	$P_{32}(62.3), T(13.8)$
	2	$G_5(155.2), T(15.9)$
	3	$P_{31}(37.4), P_{32}(64.9), T(16.9)$
	4	$G_2(315.6), G_1(291.6)$
	5	$D(4898.9)$
January 2003	1	$P_{31}(35.9), T(9.3), P_{32}(60.3)$
	2	$G_4(178.8), P_{31}(37.5)$
	3	$G_2(322.3), P_{32}(64), T(16.7)$
	4	$G_4(175.15), G_3(30.3)$
	5	$D(4594.8)$

changes, for example *August 2002* compared to its previous month *July 2002* (cf. Figure 1.3: *f-measure=0.92* and *CR=0.80*). Both months present exactly 3 clusters (cf. graphic *Total clusters* of Figure 1.4). Each cluster of *July 2002* matches a cluster in *August 2002*. No cluster disappearance, appearance, split or fusions were verified. This means that the system behaviour during these two consecutive months was very similar.

Figure 1.5 shows the projection of all cluster prototypes on the first factorial plane. In this Figure, each circle represents a cluster prototype and arrows link a cluster prototype to its correspondent cluster prototype in the subsequent month. Circles with no input arrow represent new clusters and circles with no output arrow represent disappeared clusters. The legend $(w : a, p : b)$ represents cluster prototype b in window a. Notice that most prototypes are concentrated around point $(0, 0)$. Unusual working phases are represented by prototypes outside this central region. Special attention should be paid to the three marginal cluster prototypes $(w : 19, p : 3)$, $(w : 36, p : 3)$ and $(w : 45, p : 3)$. Each of these prototypes represents a particular cluster in *March 2003*, *February 2005* and *November 2005* having a total of 5, 12 and 12 individuals, respectively. These are very thin clusters which would hardly be detected by a global analysis of the data. The most significant variable of cluster 3 in *March 2003* is G_5 and its value is equal to 157.7 whereas other clusters in this same month have value 155 for this variable. The most significant variables of cluster 3 in *February 2005* are G_3 and P_{31} with respective values 32 and 40.7, whereas other clusters in the same month have values 37 and 44 for these variables. The most significant variables of cluster 3 in *November 2005* are G_1 and G_2 with respective values 294 and 306, whereas other clusters in this same month have value 290 and 326 for these variables. In this context, Figure 1.6 illustrates individual memberships of cluster 3 in *November 2005*. In this Figure, we can clearly notice 2 stable working states represented by clusters 1 and 2, whereas cluster 3 may represent a malfunctioning state.

In our approach, it is also possible to follow-up clusters over time. Let's consider cluster 5 from *January 2002*. According to the results given by our clustering approach, the cluster traceability and most significant variables (and their respective value on cluster prototype) can be described as follows. Cluster 5 (described by variables $G_3(42), G_1(293.7), G_2(308.8)$ and $P_{31}(48.3)$) in *January 2002* corresponds to cluster 6 (described by variables

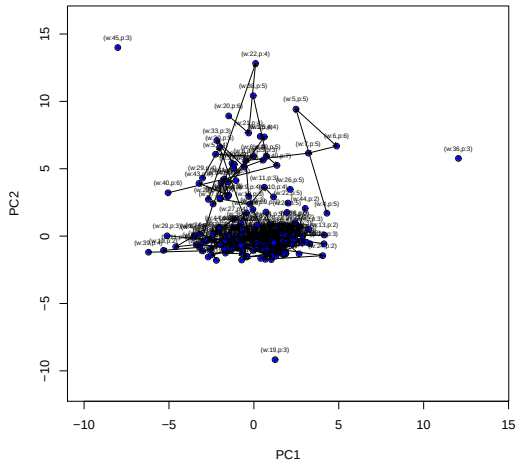

FIGURE 1.5
Projection of all cluster prototypes on the first factorial plane.

$G_3(41.7), P_{31}(48.3), G_1(292.7)$ and $G_2(312.4)$) in *February 2002* which corresponds to cluster 5 (described by variables $G_3(41.9), P_{31}(48.5), G_1(293.2)$ and $G_2(310.4)$) in *March 2002* which corresponds to cluster 2 (described by variables $G_2(310), G_1(293.2)$ and $G_3(40.5)$) in *April 2002* which corresponds to cluster 4 (described by variables $G_2(310.4), G_1(293.1)$ and $G_3(40.1)$) in *May 2002* which disappeared in *June 2002*. In the e-commerce domain, for example, this functionality would be very useful to follow-up customer purchase behaviour over time.

1.5 Conclusion

Condition monitoring is essential to detect abnormal operation conditions in critical systems in a timely way. This issue is mostly addressed by developing a (typically empirical) model of the system behaviour in normal conditions and then comparing the actual behaviour of the system with the one predicted by the model. A deviation between the measured and predicted values of system signals can reveal a component malfunctioning. In model-based strategies, the construction of an accurate model representing the system normal operation is a nontrivial and important issue.

This chapter presents a clustering approach for system monitoring which is

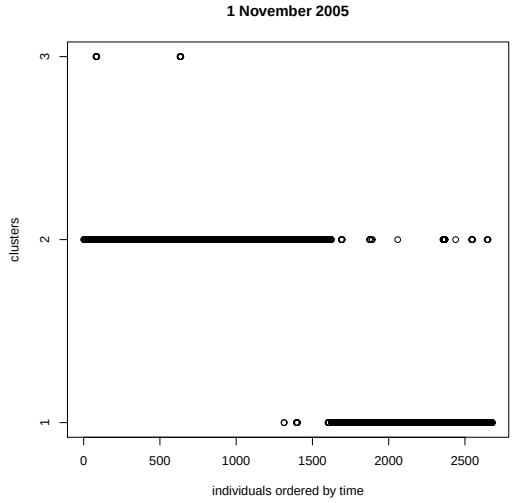

FIGURE 1.6
Memberships of individuals in November 2005.

able to detect changes over time. The main novelty in this approach concerns the change detection strategy which is based on the clustering extension, i.e., individual memberships reflecting the current system state. The changes are measured by the disagreement level between two partitions obtained from the same temporal data window. The proposed approach is positioned in a higher abstraction level of the clustering process and is totally independent of the clustering algorithm applied. It tracks cluster changes (splits, scissions, appearance and disappearance) in subsequent windows and provides a snapshot of data evolution over time. Within this framework, non-incremental methods can be used in an incremental process which can be stopped and restarted any time.

The proposed approach is generic and can be applied in different domains (e-commerce, web usage mining, industrial process monitoring, etc.) such as the one described in our experiments where a cluster represents a working state composed of different sensor measurements.

Future works may include the exploration of other clustering methods and the application of other partition-comparison criteria.

Acknowledgements

This work was supported by the ANR MIDAS project under grant number ANR-07-MDCO-008, CAPES-Brazil and INRIA-France.

Bibliography

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *VLDB'2003: Proceedings of the 29th international conference on Very large data bases*, pages 81–92. VLDB Endowment, 2003.

[2] M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis*. Sage Publications, Beverly Hills, California, 1984.

[3] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, and H. Ralambondrainy. *Classification automatique des données*. Dunod, Paris, 1989.

[4] Baptiste Csernel, Fabrice Clerot, and Georges Hebrail. Streamsamp: Datastream clustering over tilted windows through sampling. In *ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams*, 2006.

[5] O. Elemento. Apport de l'analyse en composantes principales pour l'initialisation et la validation de cartes topologiques de kohonen. In *SFC'99*, Nancy, France, 1999.

[6] Brian Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Oxford University Press, 2001.

[7] A. D Gordon. *Classification*. Chapman and Hall/CRC, Boca Raton, Florida, 1999.

[8] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[9] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Survey*, 31(3):264–323, 1999.

[10] Madjid Khalilian and Norwati Mustapha. Data stream clustering: Challenges and issues. In *The 2010 IAENG International Conference on Data Mining and Applications*, Hong Kong, March 2010.

[11] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, third edition, 1995. Last edition published in 2001.

[12] Alireza Rezaei Mahdiraji. Clustering data stream: A survey of algorithms. *Int. J. Know.-Based Intell. Eng. Syst.*, 13(2):39–44, 2009.

[13] Olfa Nasraoui, Maha Soliman, Esin Saka, Antonio Badia, and Richard Germain. A web usage mining framework for mining evolving user profiles in dynamic web sites. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):202–215, 2008.

[14] Olfa Nasraoui, Cesar Cardona Uribe, and Carlos Rojas Coronel. Tecnostreams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model. In *ICDM 2003: Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 235–242. IEEE Computer Society, Los Alamitos, 2003.

[15] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings of IEEE International Conference on Data Engineering*, pages 685–694, 2001.

[16] Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult. Monic: modeling and monitoring cluster transitions. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA*, pages 706–711. ACM, 2006.

[17] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[18] Miron Livny Zhang, Miron@cs. Wisc. Edu, Tian Zhang, Tian Zhang, Raghu Ramakrishnan, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1:141–182, 1997.

Index