Matching Partitions over Time to Reliably Capture Local Clusters in Noisy Domains

Frank Höppner[1] and Mirko Böttcher[2]

[1] University of Applied Sciences Braunschweig/Wolfenbüttel
Robert Koch Platz 10-14, D-38440 Wolfsburg, Germany
[2] BT Group, Intelligent Systems Research Centre,
Adastral Park, Orion Bldg. pp1/12, Ipswich, IP5 3RE, UK

Abstract. When seeking for small clusters it is very intricate to distinguish between incidental agglomeration of noisy points and true local patterns. We present the PAMALOC algorithm that addresses this problem by exploiting temporal information which is contained in most business data sets. The algorithm enables the detection of local patterns in noisy data sets more reliable compared to the case when the temporal information is ignored. This is achieved by making use of the fact that noise does not reproduce its incidental structure but even small patterns do. In particular, we developed a method to track clusters over time based on an optimal match of data partitions between time periods.

1 Introduction

Clustering is the partitioning of data into groups such that similar data objects belong to the same and dissimilar objects to different groups (according to some similarity measure). In this paper, we focus on one important topic in clustering: how to reliably detect small clusters in the presence of heavy noise.

We are particularly interested in so-called *local patterns* rather than the global structure of the data, because (a) the global structure is much better known by domain experts and thus not considered as interesting and (b) small structures may indicate niches or upcoming trends and as such are potentially of high value to businesses. Two difficulties are connected with local pattern discovery: Firstly, within the multitude of local patterns discovered most prove to be uninteresting (since in line with the global structure) or incidental on closer inspection (cf. [1]). Secondly, local patterns are easily obscured by noise. In particular, cluster algorithms often fail in discovering small clusters in noisy domains since they aim at the detection of large and well separated clusters.

In our approach, we divide up the data (based on time stamps) into several slices and analyze the data of consecutive slices. This idea and its consequences are discussed in Section 2 and the clustering algorithm we are going to use (modified OPTICS [2]) in Section 3. By matching clusters in consecutive partitions, a cluster history is obtained (cf. Sec. 4). We exploit this historical information to distinguish incidental from substantial small clusters (Sec. 5). Finally, some results are shown in Section 6.

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 479–486, 2007.

Related Work. Clustering in the presence of time-stamped and changing data has been studied in the context of moving cluster detection and data streams. Although it is possible to track the change of clusters with our proposed algorithm, we will focus on the detection of local patterns here only. In contrast to most stream mining approaches [3,4] we do not keep a condensed representation of the data stream in main memory, but analyze subsequent parts of the stream one after another. The algorithm we propose is not an online algorithm.

We will compare the partitions obtained from data of consecutive time periods, therefore *clustering ensembles* is also a related area. In the clustering ensembles literature, however, the partitions are usually obtained from identical data sets but different clustering algorithms, e.g. [5], or from the same algorithm using different subsamples of the original data set, e.g. [6]. In this paper, however, we compare partitions obtained at different points in time in order to gain information about the temporal stability of a cluster.

Brief Review of OPTICS. Since we are going to use the OPTICS algorithm we want to briefly review it here. OPTICS [2] is a density-based approach, which means that for a data point x to belong to a cluster, the density around x must exceed some density threshold ϱ. The density ϱ is given implicitly by the size of a hypersphere with radius ε (defining the neighborhood $N_\varepsilon(x)$ of volume V_ε) and a required number MinPts of data objects within this neighborhood. The condition for x to establish a cluster is $|N_\varepsilon(x)| \geq MinPts$. In terms of density this is equivalent to $\frac{|N_\varepsilon(x)|}{V_\varepsilon} \geq \frac{MinPts}{V_\varepsilon} =: \varrho$. OPTICS requires only the parameter MinPts and an upper bound for the neighborhood size. The outcome of the algorithm is an ordering of all data points together with a reachability distance for each data point (cf. Fig. 2). Simplifying, the reachability distance of x is the smallest neighborhood radius ε such that x belongs to a cluster. In the reachability plot all data objects are ordered on the horizontal axis with their reachability distance on the vertical axis, such that consecutive data objects in the plot having all their reachability distances below some given threshold belong to the same cluster at the corresponding density level.

2 Selection of the Clustering Algorithm

We start by dividing our existing data into slices S_i of approximately the same size and consider the sequence of partitions obtained from clustering a window of n consecutive slices. By comparing the clusters of consecutive partitions we trace the clusters over time and thereby also measure their stability. The rough sketch of our proposed PAMALOC algorithm (**pa**rtition **ma**tching to detect **lo**cal **c**lusters) is shown in Fig. 1. The data does not have to be timestamped, but all data in S_i should have been observed before the data in any S_j, $j > i$.

Given this coarse idea of how to attack the problem, we may use a variety of clustering algorithms in line 2 and 6. They should, however, meet at least the following four requirements:

• *Flexibility.* The notion of a cluster supported by the clustering algorithm must be flexible in order to cope with any arbitrary cluster shape. This rules out

INPUT: t time interval, n number of slices to use, S_t data slice collected during t	
Set $t = n$	1
Run clustering algorithm with slices $S_{t-n+1} \cup S_{t-n+2} \cup ... \cup S_t$	2
Extract clusters, denote the set of clusters (partition) by $P^t = \{C_1^t, C_2^t, ..\}$	3
While there is another data slice S_{t+1} available {	4
t=t+1	5
Run clustering algorithm with slices $S_{t-n+1} \cup S_{t-n+2} \cup ... \cup S_t$	6
Extract clusters, denote the set of clusters (partition) by $P^t = \{C_1^t, C_2^t, ..\}$	7
Match clusters in P^t to those in P^{t-1}	8
Evaluate the stability of the cluster in the cluster's history}	9
OUTPUT: partition of stable clusters	

Fig. 1. Sketch of the proposed PAMALOC algorithm

all clustering algorithms that assume a certain model for a cluster (e.g. k-Means, mixture of Gaussians).

• *Robustness wrt. parameters.* At the time of setting up the algorithm, we do not know what clusters will show up in the future. So we cannot tailor our parameters to a 'representative dataset', but it must work with a wide range of noise and cluster densities.

• *Stability wrt. input data.* The algorithm should be as stable as possible (similar input should lead to similar output). This is important since we want to trace and compare clusters obtained from different runs of the algorithm. Lack of stability might be caused by iterative optimization, where a different initialization may already lead to a different partition (k-means and derivatives, mixture of Gaussians), or by the choice of the distance measure (in hierarchical clustering with single-linkage a single new data object may have dramatic effects).

There is probably no *truly optimal* choice, but the OPTICS algorithm [2] is at least considered being a very good candidate. The cluster's shape is highly flexible due to the concept of *density-connected regions*, that is, clusters are made up by data points close to each other with their individual data point density exceeding some threshold. Regarding robustness, the MinPts parameter of the OPTICS algorithm can be used to control the robustness of the results: With MinPts=2 small perturbations may have large effects on the connectivity (similar to single-linkage clustering), but by increasing MinPts the response to small changes is reduced. Regarding stability, as long as the clusters are clearly separated and the noise level is low, the clusters exhibit themselves quite prominently in the reachability plot and OPTICS produces stable results.

However, stability becomes a major problem if the noise level increases. The top right image in Fig. 2 shows an artificial example consisting of about 500 uniformly distributed noise objects and about 200 data objects that belong to artificially superimposed clusters. It is really hard to tell whether some of the data agglomerations are small but substantial local patterns or just occurred by chance. At this noise level the random noise points lump together into spurious clusters (e.g. shaded region in the image). If we now remove some data and add some other (replace oldest with a new data slice) this will cause considerable

changes in the *shape* of the *incidental data agglomerations* we see in this figure – but to a lesser extent to the shape of true clusters. The main idea for the distinction between (small local) patterns and agglomeration by chance is therefore the *stability of a cluster*, i.e. whether the cluster reappears in the next iteration. Even if the random points spontaneously agglomerate, it is very unlikely that this happens in the same way over multiple analyses. The longer an agglomeration – even if it is of low density – remains stable over time, the more likely it is that we observe some *true pattern* in the data.

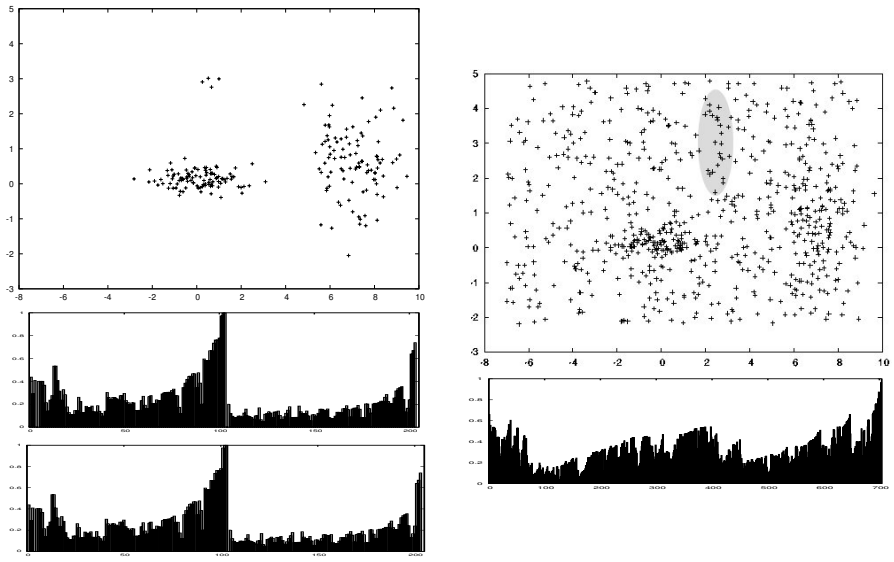

Fig. 2. The outcome of the OPTICS algorithm (reachability plot, bottom line) for the clean and a noisy data set

But still, the procedure in Fig. 1 requires that we come up with some clusters. As we have argued, we *cannot* be sure about the partition, so we are condemned to make errors. This imposes a fourth requirement:

• *Ability to preserve ambiguity in the partition.* Ambiguity in the partition can help in the sense, that we can consider more *cluster candidates* than *actually expected clusters*. We keep several possible clusters under consideration to be able to defer the decision making, whether a data agglomeration is incidental or significant, to a later point in time. Ambiguity can be achieved, for instance, by means of a hierarchical clustering (which is ambiguous in the sense that a particular data object may belong to several clusters). Since we can not think of a clustering algorithm that is capable of detecting *local patterns* without having to discover more global structure first (for a discussion see [7]), it seems that generally hierarchical approaches are better suited than partitional approaches.

3 Extraction of Clusters

OPTICS finds an ordering of the data objects such that the reachability plot has the following property: consecutive data objects in the plot having all their reachability distances below some given threshold ε belong to the same cluster (at density level $\frac{MinPts}{V_\varepsilon}$). Therefore, clusters are represented by valleys in the reachability plot and some user-specified thresholds on the steepness of the flanks of the valleys are used in [2] to decide whether a valley qualifies as a *deep valley* and thus as a cluster. Fixing such thresholds is easy, if the cluster density differs clearly from the background density. However, with noisy data and comparatively small clusters of non-uniform density, clusters do not show up that clearly in the reachability plot and fixing the thresholds is difficult if at all possible (cf. Fig. 2, left vs. right).

We therefore propose a different approach that still uses the reachability plot as its basis. If we draw a straight line in this plot, we thereby mark a certain data density level. In practice we are only interested in subclusters with significantly increased density, say, a factor of f_d higher. The same is true for the subcluster of a subcluster. Thus we set up a cascade of density levels (given by $\varrho_i = \varrho_0 \cdot {f_d}^i$ with initial density ϱ_0 and levels $i = 1...L$) and whenever the density of some subset of the data drops below such a density level we consider it as a cluster *candidate*. To depict the cluster candidates we draw a rectangle in the graphical representation (cf. Fig. 3) whenever this happens. A deep valley in the reachability plot will cross multiple lines and therefore indicate hierarchically nested rectangles in Fig. 3. By considering all these sets as cluster *candidates*, we tolerate for the moment that many of them may be incidental.

Given a data density ϱ_i, how can we match it to a vertical line in the reachability plot? Within a cluster, the reachability of a data point x corresponds to the core distance of x, which is the smallest distance such that MinPts data objects can be found in a hypersphere of this radius around x. Given the volume V of the hypersphere ($V = \frac{\sqrt{\pi^d}}{\Gamma(d/2)}\varepsilon^d$ where Γ denotes the Gamma function) we may estimate the density at x by $\varrho = MinPts/V$. Using this equation we can transform our sequence of densities into a sequence of hypersphere radii.

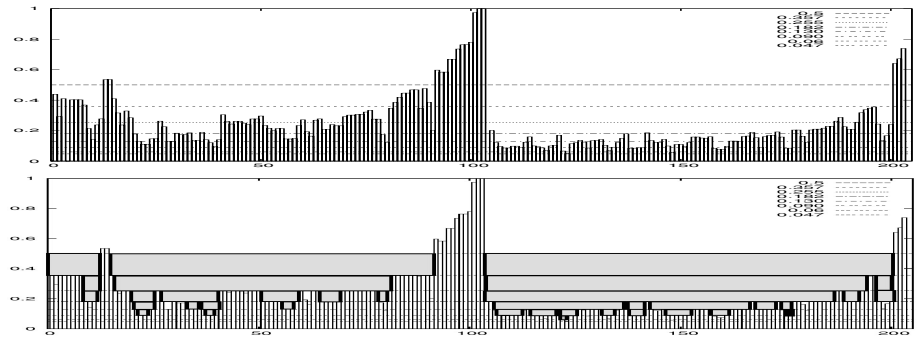

Fig. 3. Top: Reachability plot with density lines. Bottom: Whenever the reachability drops below a density line, a new cluster candidate (shown as a rectangle) is induced

4 Matching of Clusters

Once we have two hierarchies of clusters we match the clusters against each other to see if a cluster survives over time. As the rationale for this matching we need some measure that indicates for each pair of clusters (one cluster candidate from the old, one from the new hierarchy) whether it would be a good match or not. When both partitions were obtained from two completely different data sets (and the clusters cannot be represented by some model) such a match becomes somewhat difficult, because we have no means to compare the clusters directly against each other. Therefore we perform clustering on more than one data slice per clustering, such that data from some slices are contained in *both* clusterings.

Figure 4 illustrates this: For every cluster analysis we use a data set that consists of n data slices. Then, between any two consecutive runs, we have a common basis of $n-1$ slices for matching. Given clusters C_i^{t-1} and C_j^t from consecutive data sets D^{t-1} and D^t, we adopt the Jaccard measure for this purpose:

$$J_{i,j} = \frac{|C_i^{t-1} \cap C_j^t|}{|(C_i^{t-1} \cup C_j^t) \backslash (S_t \cup S_{t-n})|}$$

$J_{i,j}$ becomes 1 if both, old and new cluster, contain the same data objects in the shared part of their dataset. We have to exclude the set $S_t \cup S_{t-n}$ from the usual Jaccard denominator, because this set does not contain shared data objects.

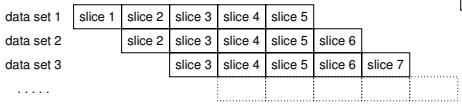

Fig. 4. Data slices used for clustering

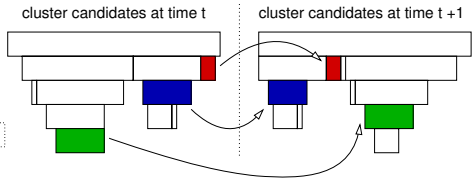

Fig. 5. Matching of rectangles in two hierarchies

We construct a $|P^{t-1}| \times |P^t|$ cost matrix M with $M_{i,j} = 1 - J_{i,j}$. To match the clusters pairwise, we use the bipartite graph matching algorithm by Munkres (also known as the Hungarian algorithm) [8]. We accept an assignment of two clusters only in case they share at least some percentage p_m of the data, otherwise most of the data has obviously been scattered among other clusters and the original cluster does no longer exist. We have selected a relatively low value of $p_m = 30\%$ to account for effects such as shrinking or varying noise levels.

5 Evaluating the Stability

Whenever a new data slice arrives, we perform a new cluster analysis, extract new clusters and match them with the old ones. To find *good clusters*, usually cluster

validity measures are employed that account for the compactness or separation of clusters (or, in case of OPTICS, the steepness of the flanks). In the presence of much noise, these measures tend to break down. We do not want to investigate clusters that are likely to vanish in the next period of time, that is, have a very short *lifetime*. By *lifetime of a cluster* we denote the number of data sets over which a cluster survives. Whenever a cluster is successfully matched into the next hierarchy, its lifetime is increased.

But not all matched clusters are matched equally well. The Jaccard measure tells us precisely how much data is shared between the old and the new cluster. Rather than requiring, e.g., three matches in a row, we may directly aggregate the matching values in the most recent r periods to a single stability value. If a cluster has been matched several times, each time with a matching value of m_i, we define its stability as: $S_t = \sum_{i=0}^{r} \beta_i \cdot m_i$. We choose β_i heuristically such that S_t becomes a weighted average. Since recent matches are more important than matches in the past, we use normalized coefficients from a Gaussian ($\beta_i = \exp(-i^2/r^2)$ normalized to $\sum_{i=1}^{r} \beta_i = 1$). The best stability value is 1.0 for a cluster that has been matched r times in the past with a Jaccard measure of 1.0. For a cluster to be stable at t, we require $S_t > 0.5$. Note that this measure does not explicitly account for the size or density of the cluster, and thus represents **no** bias towards large and dense clusters (which is important for the discovery of *local* patterns).

6 Example and Conclusion

Due to lack of space, we discuss the performance of PAMALOC for an artificial data set. We consider a sequence of 11 data slices S_t with $t = 0...10$. Every slice contains up to four (normally distributed) moving clusters, two of them are moving (at different speed) and the others (denoted by A and D) are static. The size of the clusters is also varying: The moving clusters are expanding and shrinking, respectively (by 5 data points per time step t). Cluster D will appear at $t = 7$ for the first time and its initial size is 13; with every time step t its size will increase by 3 data points. Cluster A is a very small one, it consists of 4 data points per time slice only. In the final slice ($t = 10$) the clusters consist of about 250 data points in total. We add 500 uniformly distributed data points, such that we have at least twice as much noise as substantial data (cf. Figure 2 (right) for the first data slice). In particular, cluster A, which consists of 4 data points per slice only, is impossible to detect in the right image.

Using five time slices for each OPTICS run, PAMALOC reaches the last time slice ($t = 10$) after 6 iterations, but only the last iteration is shown in Fig. 6 ($MinPts = 3$, $f_d = 1.3$). The stable clusters ($S_t > 0.5$) have been shaded: the darker the rectangle the more stable the cluster. Compared to experiments with less noise (not shown) two observations can be made: (1) due to the high noise level, the whole data set is density-connected for the first 5-6 data density levels and (2) we have more 'small clusters' (small rectangles), which is due to spontaneous data agglomerations in the noise. Most of these agglomerations vary from time slice to time slice that much, that they do not become stable.

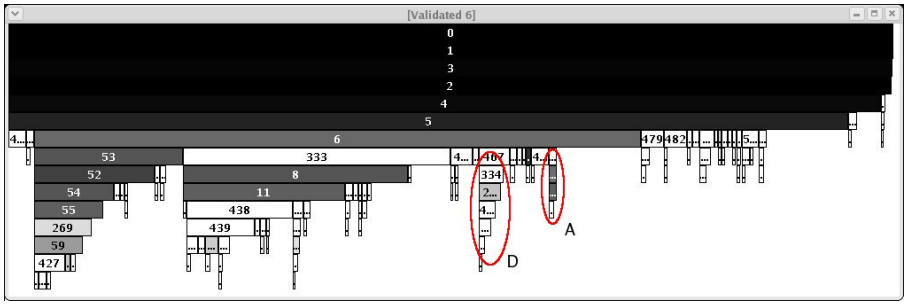

Fig. 6. Final hierarchy for example 2 (67 % noise)

At the final hierarchy only very few of these small rectangles are stable with the rectangles belonging to cluster A being the most stable among them. Cluster D is also difficult to detect, because its initial small size ($t = 7$) must compete against all the noise around it. Nevertheless, in the final hierarchy of Fig. 6 it has been recognized as a stable cluster (light shaded rectangle).

Conclusion. We have presented the PAMALOC algorithm for detecting small patterns in very noisy data. Our initial experiments show, that even in cases where we have much more noise than substantial data points, we are capable of identifying very small local patterns. We consider the fact that only a very small number of false positives were flagged as very encouraging.

References

1. Hand, D.J., Adams, N.M., Bolton, R.J. (eds.): Pattern Detection and Discovery. LNCS (LNAI), vol. 2447. Springer, Heidelberg (2002)
2. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, Philadelpha, pp. 49–60. ACM, New York (1999)
3. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, Montreal, pp. 103–114. ACM Press, New York (1996)
4. Guha, S., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams. In: Proc. Ann. Symp. Foundations of Computer Science, pp. 359–366 (2000)
5. Ghosh, J., Strehl, A., Merugu, S.: A consensus framework for integrating distributed clusterings under limited knowledge sharing. In: Proc. NSF Workshop on Next Generation Data Mining, pp. 99–108 (2002)
6. Topchy, A., Minaei-Bidgoli, B., Jain, A.K., Punch, W.F.: Adaptive clustering ensembles. In: Proc. of the 17th Int. Conf. on Pattern Recognition, pp. 272–275 (2004)
7. Höppner, F.: Local pattern detection and clustering – are there substantive differences? In: Morik, K., Boulicaut, J.-F., Siebes, A. (eds.) Local Pattern Detection. LNCS (LNAI), vol. 3539, pp. 53–70. Springer, Heidelberg (2005)
8. Munkres, M.: Algorithms for the assignment and transportation problems. Journal of the Society of Industrial and Applied Mathematics 5(1), 32–38 (1957)