

Visualising the Cluster Structure of Data Streams

Dimitris K. Tasoulis¹, Gordon Ross², and Niall M. Adams²

¹Institute for Mathematical Sciences, ²Department of Mathematics
Imperial College London, South Kensington Campus
London SW7 2PG, United Kingdom
{d.tasoulis,gordon.ross,n.adams}@imperial.ac.uk

Abstract. The increasing availability of streaming data is a consequence of the continuing advancement of data acquisition technology. Such data provides new challenges to the various data analysis communities. Clustering has long been a fundamental procedure for acquiring knowledge from data, and new tools are emerging that allow the clustering of data streams. However the dynamic, temporal components of streaming data provide extra challenges to the development of stream clustering and associated visualisation techniques. In this work we combine a streaming clustering framework with an extension of a static cluster visualisation method, in order to construct a surface that graphically represents the clustering structure of the data stream. The proposed method, OpticsStream, provides intuitive representations of the clustering structure as well as the manner in which this structure changes through time.

1 Introduction

Advances in technology have resulted in an increasing number of application areas generating streaming data, that is, data obtained by observation of multiple indefinitely long and time-evolving sequences. These applications areas include astronomy and earth sciences, telecommunications and network monitoring. Information visualisation techniques have provided a means to help the exploration of large data sets [4]. As such there is a need to develop visual data analysis methods that can deal successfully with the challenges arising from the dynamically changing nature of streaming data.

Clustering is the process of partitioning a data set into homogeneous subsets called clusters. Since the publication of the first comprehensive study of clustering algorithms [14], these methods have been applied to a wide variety of fields, ranging from text mining [6] to genomics [7].

Cluster visualisation for two dimensional data is readily achieved using simple scatter plots. To handle higher dimensional data most methods try to determine two- or three-dimensional projections of the data that retain certain properties of the high-dimensional clusters. Dendrograms, produced by hierarchical clustering algorithms, are a popular visualisation techniques despite the quadratic time required to construct them [9]. However such techniques become difficult to apply in large, high-dimensional data sets, particularly when the clusters are not clearly separated. More sophisticated techniques have been developed to address such issues [3, 8].

Although there has been some effort [10] to extend cluster visualisation to streaming data, the focus has been applications with a relatively stable, periodically updated environment. In detail they assume a permanent database to which insertions and deletion operations occur. Moreover, they are designed with the assumption that all the data can be stored and retrieved whenever it is required. Furthermore they do not incorporate any forgetting mechanism. As such, almost all the foundational streaming data problems are not addressed by such approaches.

In this paper we present an extension of the OPTICS algorithm [3] to the streaming data model. OPTICS uses a density identification method to create a one-dimensional cluster-ordering of the data. We will show that embodying this idea in a stream clustering method can provide new insights into both the current clustering structure and the structure evolution. This is a critical issue in this area since the basic assumption of a data stream model is the dynamically changing nature of the streams. Thus, such an algorithm should have the capacity to adapt its visualisation medium to rapidly changing dynamics of the sequences. Finally, scalability in the number of sequences is becoming increasingly desirable, as data collection technology develops.

The paper proceeds as follows: the next section provides a description of cluster visualisation in static data sets. Stream clustering is described in Section 3, along with the proposed visualisation method, called *OpticsStream*. Next, in Section 4, we demonstrate the behaviour of the *OpticsStream* with an experimental analysis of both artificial and real data sets. The paper concludes with a discussion in Section 5.

2 Visualising Clusters in Static Data

Unlike many other procedures, the OPTICS algorithm [3] (Ordering Points To Identify the Clustering Structure) does not produce an explicit clustering of a data set, but instead creates an augmented ordering of the data representing its density-based clustering structure. For medium sized data sets, this cluster-ordering can be represented graphically, a fact that allows interactive exploration of the intrinsic clustering structure and thereby offering additional insight into the distribution of the data

Consider a data set $X \subset \mathbb{R}^d$ of n vectors (objects), and a metric distance function $dist : X \times X \rightarrow \mathbb{R}$. Additionally, for each object $p \in X$ and a value $\epsilon \in \mathbb{R}$, we define the set $N_\epsilon(p) = \{q \in X \mid dist(p, q) \leq \epsilon\}$, as the ϵ -neighbourhood of p . ϵ is a user defined parameter closely related to the application, the distance metric and the dimensionality of the data. For more details about reasonable values for this refer to [12].

The OPTICS point ordering is constructed using two quantities, the “*core-level*” and the “*reachability-distance*”, that are computed for each object p in the database. These quantities are defined as follows:

Definition 1. (*core-level*) Let $\epsilon \in \mathbb{R}$, $\mu \in \mathbb{N}$, be user defined parameters and $dist_\mu(p)$ be the distance from p to its μ -nearest neighbour. The core-level of p , $CLev(p)$ is defined as:

$$CLev(p) \begin{cases} \infty & \text{if } |N_\epsilon(p)| < \mu \\ dist_\mu(p) & \text{otherwise} \end{cases}$$

Where $|N_\epsilon(p)|$ is the number of objects from the database in the ϵ -neighbourhood of p . μ is the user defined that combined with value of ϵ , separates the data as outliers or not. If in the ϵ -neighbourhood around a point less than μ points reside then this corresponds to an outlier [12]. Next we define the “reachability-distance”, for each pair of objects in the database.

Definition 2. (reachability) For two objects p, q in the database the reachability of p wrt q is defined as $RDist(p, q) = \max\{CLev(p), dist(p, q)\}$.

Under these definition the cluster ordering of the data is constructed through the following definition:

Definition 3. (cluster ordering) Let $\epsilon \in \mathbb{R}, \mu \in \mathbb{N}$, and CO be a totally ordered permutation of the n objects of X . Each $o \in X$, is assigned the additional attributes $Pos(o)$, $Core(o)$, and $Reach(o)$, where $Pos(o)$, symbolizes the position of o in CO . The ordering CO is called a cluster ordering wrt ϵ and μ if the following three conditions hold:

1. $\forall p \in CO : Core(p) = CLev(p)$
2. $\forall o, x, y \in CO : Pos(x) < Pos(o) \wedge Pos(y) > Pos(o) \Rightarrow RDist(x, y) \geq RDist(o, x)$
3. $\forall p \in CO : Reach(p) = \inf\{RDist(p, o) | o \in CO \text{ and } Pos(o) < Pos(p)\}$.

In the cluster ordering defined this way each object is positioned in the ordering so that it has minimum reachability distance to all preceding objects in the ordering. The cluster-ordering of a data set can be represented and understood graphically. In principle, one can see the clustering structure of a data set if the reachability-distance values r are plotted for each object against the corresponding cluster-ordering CO . An example of a very simple 2-dimensional data set is depicted in Fig. 1(a). The CO for this dataset is depicted in Fig. 1(b), where in the y -axis corresponds to the reachability distance. A detailed description of the workings of the OPTICS algorithm is given by [3].

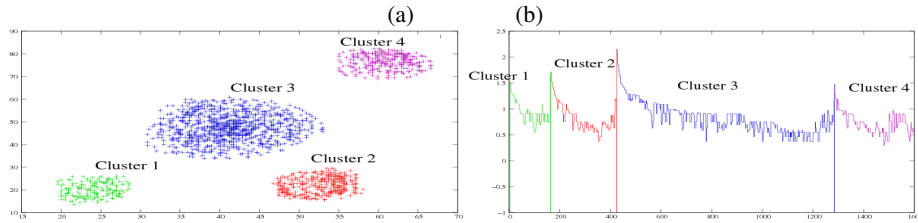


Fig. 1. Reachability Distance ordering (right) of a 2-dimensional data set with 4 clusters (left).

3 Stream Clustering

There are a large number of highly efficient and effective clustering methods [9]. However, most are *batch* methods originally designed for static data and therefore relying

on the assumption that data are available in a permanent memory structure from which information can be retrieved at any time. Streaming data is more demanding, requiring a one-pass scan of the data. Thus, many standard clustering methods are not directly applicable to streaming data. More recent approaches have focused precisely on the streaming data model – accommodating both dynamics and the memory limitation issues [5, 13].

There are very few methods for stream clustering visualisation. One example is described by [1], where so-called velocity density estimation is proposed to provide a visual diagnosis of changes in a data stream. Here, the idea is to estimate the rate of change (“velocity”) of a kernel density estimate at each spatial location. This is obviously an extremely computer-intensive procedure. Moreover, to make the method applicable in greater than two dimensions, a projection technique is proposed, which increases the computational complexity of the algorithm still further.

A widely used and important concept in stream clustering is *micro-clusters*. These are quantities that try to summarise the data arriving over a continuous stream. In the next paragraph we present a micro-cluster framework.

3.1 The Micro-Clustering Framework

Among the various models developed for data stream clustering the micro-clustering framework has been successfully employed by various different algorithms [2, 5]. In this framework the weight of each data point in the stream decreases exponentially with time t via a fading function $T_\lambda(t) = 2^{-\lambda t}$, where $\lambda > 0$. The parameter λ control the rate that historic data is down-weighted. The smaller the value of λ , the greater importance is given to historical data. To this end we can extend both the notions of “core-level” and “reachability” used in static clustering algorithms [3, 12] to the data stream setting.

For a spatio-temporal data set $X' = \{\{x_1, t_1\}, \dots, \{x_n, t_n\}\}$, where x_i is a vector and t_i is the corresponding time-stamp, a micro-cluster is defined by the quantities w, c, r , which attempt to summarise information about the data density of a particular area. Two distinct types of micro-clusters are considered, based on the values of r, w , and the additional user-defined parameters ϵ and μ . If $r < \epsilon$ and $w > \mu$, then the micro-cluster is considered to be a *core-micro-cluster*, that accounts for a “dense” region of the data. Otherwise the micro-cluster is called an *outlier-micro-cluster*, that accounts for less dense regions of data. Formally we can define these micro-clusters as follows:

Definition 4. (*core-micro-cluster*) A micro-cluster $MC_t(w, c, r)$ is defined as a *core-micro-cluster* $CMC_t(w, c, r)$ at time t for a group of streaming points $\{x_i, t_i\}$, $i = 1, \dots, n$, and parameters ϵ, μ when $w \geq \mu$ and $r \geq \epsilon$. Where $w = \sum_{i=1}^n T_\lambda(t_i)$ be the micro-cluster’s weight, $c = \frac{\sum_{i=1}^n x_i T_\lambda(t_i)}{w}$, be its center and r its (weighted) mean radius, $r = \sum_{i=1}^n \frac{T_\lambda(t_i) \|c - x_i\|}{w}$.

The intuition behind the micro-clustering framework is to maintain a description of the data streams by a list of micro-clusters. As data points are instantiated at each time step from the stream, they are *merged* to their nearest micro-clusters provided that they are sufficiently close. Otherwise new micro-clusters are spawned in order to merge

possible future similar data, and thus capture the density structure of the stream. The quantities w, c, r , can be incrementally computed for each micro-cluster. Consider a micro-cluster for which no points were merged between time step t_p and the current time t_c , where the point x_c is being considered for merging. Thus $w_{t_c} = 2^{-\lambda(t_c-t_p)}w_p$, where w_p is the weight value at time t_p . Incrementally computing c, r involves the use of two quantities, $CF1_t = \{\sum_{i=1}^n x_{i,j}T_\lambda(t_i)\}$, for each coordinate j of the data point $x_i \in \mathbb{R}^d$, and $CF2_t = \{\sum_{i=1}^n x_{i,j}^2T_\lambda(t_i)\}$. The quantities $CF1_t, CF2_t$ can be incrementally computed at time t_c as $CF1_{t_c} = 2^{-\lambda(t_c-t_p)}CF1_{t_p} + x_c$, and $CF2_{t_c} = 2^{-\lambda(t_c-t_p)}CF2_{t_p} + x_c^2$.

Note that the weight of core-micro-clusters must be greater than or equal to μ and the radius must be less than or equal to ϵ , in order to represent “dense” regions of the data space.

In an evolving data stream the role of clusters and outliers, that is points that do not participate in clusters, often exchanges. To compensate for this phenomenon, two types of micro-clusters are used [5]:

- *Potential core-micro-clusters*, when $w_c \leq \beta\mu$ and $r \leq \epsilon$,
- *Outlier-micro-clusters*, when $w_c > \beta\mu$ and $r > \epsilon$,

where β is a user defined parameter. As described above, the difference between these micro-clusters relates to the constraints on the weight and the radius of each micro-cluster. Maintaining two lists; one for the potential core-micro-clusters, and one for outlier-micro-clusters, and updating them on-line can provide a density description of the data space, that can be further queried to provide knowledge of the clustering structure. These two lists are maintained using the following procedure:

Procedure ListMaintain

1. Initialise two lists PL, OUL ; one for the potential core-micro-clusters, and the other for the outlier-micro-clusters.
2. Each time a new point $p = \{x, t\}$ arrives do one of the following:
 - (a) Attempt to merge p into its nearest potential core-micro-cluster c_p : If the resulting micro-cluster has a radius $r > \epsilon$ then the merge is omitted.
 - (b) Attempt to merge p into its nearest outlier-micro-cluster o_p : if $r > \epsilon$ the merge is omitted. Otherwise, if the subsequent weight w of o_p exceeds μ , then move o_p to PL .
 - (c) A new outlier-micro-cluster is created, centered at p .
3. Periodically prune from PL and OUL those micro-clusters for which $w_c \leq \beta\mu$ and $w_c \leq \xi$.

Periodic pruning of the micro-clusters can occur sufficiently often such that the available memory is not exceeded. [5] proposes to conduct this pruning based on a so called *time-span* parameter of potential core-micro-clusters. Note also that ξ is a user-defined parameter that determines the lower limit for the weight of an outlier-micro-clusters before it is pruned, and its value can be connected with T_p [5].

Following the procedure outlined above, the micro-clusters maintained on-line capture the dense areas of data streams. An example is demonstrated in Fig. 2. Around each micro-cluster the ϵ area is depicted with a dashed circle, while the computed radius r of each micro-cluster is depicted with a solid circle.

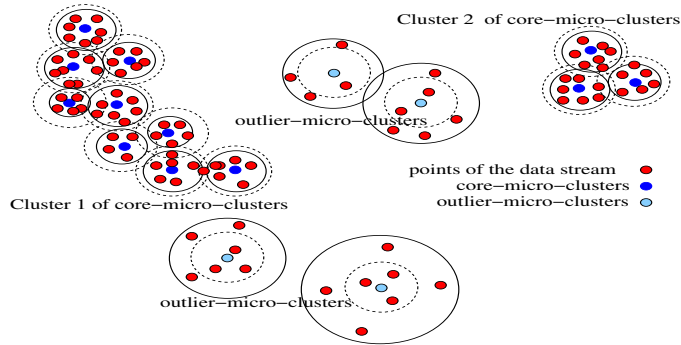


Fig. 2. An example of the result of the micro-clustering framework.

3.2 Stream Cluster Visualisation

We now propose a stream clustering visualisation methodology. This approach can potentially operate in a real-time environment and can produce a time-sensitive map representing the clustering structure in an understandable format. To achieve this, the OPTICS methodology and the micro-clustering framework described above are combined, to provide a 3-dimensional plot that depicts the evolution of the stream cluster structure over time.

In short, we apply the concepts behind the OPTICS algorithm to the potential core-micro-cluster list, translated to the streaming data context. First, we need to define the core-micro-cluster neighbourhood:

Definition 5. (*Micro-cluster neighbourhood*) Let $\epsilon \in \mathbb{R}$, be a user defined parameter and PL a potential core-micro-cluster list. Then for a potential core-micro-cluster c_p , we define the micro-cluster neighbourhood of c_p , as

$$N(c_p) = \{c_q \in PL \mid \text{dist}(c_p, c_q) \leq 3.0\epsilon\}.$$

The function $\text{dist}(c_p, c_q)$, returns the Euclidean distance between the centers of c_p and c_q .

Note that the distance between the centers of two neighbouring micro-clusters is required to be less than 3 times the ϵ parameter. Intuitively, this considers micro-clusters to constitute contiguous high density areas when the distance between the ϵ radius spheres around them is less than ϵ . Although this is a crude way of defining neighborhoods, it is possible to use other measures that take into consideration the radius of each micro-cluster. We can also extend the definition of core-level distance to the micro-cluster framework as follows:

Definition 6. (*Micro-cluster core-level*) Let $\epsilon \in \mathbb{R}$, $\beta \in \mathbb{R}$, $\mu \in \mathbb{N}$. The core-level of c_p , $CLev(c_p)$ is defined as:

$$CLev(c_p) = \text{radius of } c_p$$

The difference here is that there is no need to compute core-level of a micro-cluster from the neighbourhood of c_p , as such information has already been incorporated into the radius of c_p . Both the definitions of micro-cluster reachability and ordering remain the same as definitions 2 and 3, respectively. Assuming the existence of a potential micro-cluster list PL we can construct an order list OL from it by applying the following algorithm:

Procedure StreamOptics

1. While there is still a micro-cluster c_p in PL that has a neighbourhood size $|N(c_p)| > 1$, initialize a list S of all the micro-clusters in $N(c_p)$.
2. Remove c_p from PL and add to OL .
3. Remove all micro-clusters in $N(c_p)$ from PL .
4. For each c_l in S , compute $RDist(c_l, c_p)$.
5. For each c_l in S , insert all the micro-clusters in $N(c_l)$ to S .
6. Remove from PL all the micro-clusters in S .
7. Remove the object c_l from S with the smallest $RDist(c_l, c_p)$, and insert it OL , until S is empty.

The changes which occur at each step of the micro-cluster maintenance procedure produce insertions and deletions into the PL list that should affect only a limited subset of the ordering in OL . If we employ incremental techniques such as those proposed in [10], the computational effort to maintain the OL list though time is very small compared to the reward of the mined information, since this depends only on the change to the data structure rather than on the dimensionality or size of the data stream.

In this way the StreamOptics methodology developed here maintains an ordered list OL , of core-micro-clusters, based on their reachability distance. To this end, at each time step, we can use the methodology of OPTICS to produce a reachability plot that depicts the current micro-cluster structure. By keeping track of every such ordering, we can have a record of how the ordering of micro-clusters, and hence the clusters in the data stream, are changing in time.

The reachability plots produced by the OPTICS approach are 2-dimensional. Since we are tracking how these plots change through time, we consider time as being an extra dimension and combine these to form a 3-dimensional plot. Essentially, we combine the reachability “curve” at each time t in order to create a 3-dimensional surface plot, the *reachability surface*, $RS : T \times N \rightarrow R$, where each point (t, i) depicts the reachability distance of the i th micro-cluster in the ordering at time t . In order to better visualise the cluster structure represented by this 3-dimensional surface, we can represent it as a contour plot, to provide further insight into how the clusters are changing in time.

However we should note that the StreamOptics methodology does not guarantee that the position of the micro-clusters remains the same, even when nothing changes in the stream. However, this effect is minimized if the the order of the micro-clusters in PL is re-arranged so that it matches OL . This way when ordering is re-constructed the optical changes should be minimal.

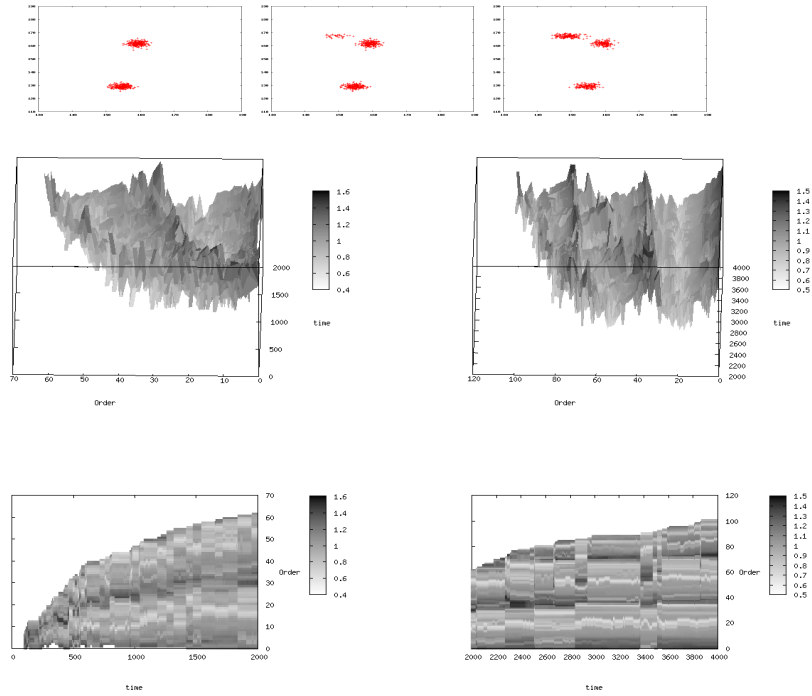


Fig.3. $Dset_{2d}$ along with the reachability surface produced by StreamOptics for different time instances.

4 Experimental Results

We now provide some results produced by StreamOptics, for both synthetic and real data sets.

4.1 Spawning Clusters

In this section we use the 2-dimensional synthetic data set, $Dset_{2d}$. It initially consists of random points drawn sequentially from a finite mixture of two Gaussian distributions randomly placed in $[100, 200]^2$. At time point 2000, we simulate the spawning of a new cluster by introducing one more Gaussian component into the model. In Fig 3, we illustrate the results of StreamOptics at two different time instances. In the top row, 500 data points are drawn from the Gaussian mixture at times 2000, 2100 and 3000, from left to right, respectively. Clearly at time 2100 the points instantiated from the newly introduced Gaussian component start to have an apparent effect in the mixture. In the second row of the figure, the reachability surfaces are exhibited, for the time ranges $[0, 2000]$, and $[2000, 4000]$, from left to right respectively. Finally in the 3rd row the latter surfaces are exhibited as contour plots. As illustrated, the reachability

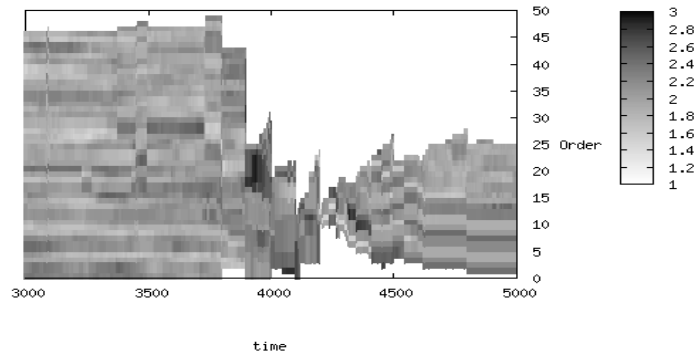


Fig. 4. The contour plot of the reachability surface produced by StreamOptics for Dset_{5d}.

surface attains a constant two valley form as long as the clustering structure remains steady. This is the expected behaviour since nothing is changing in the data stream. Interestingly, when the new component is introduced, this change affects the surface in both the *time* and *ordering* dimensions, since new micro-clusters are introduced to capture the appearance of the new cluster. Gradually a new peak is introduced in the reachability surface, that demonstrates the birth of the new component in the mixture. Finally the surface stabilises again when the new structure of the data is described adequately. There is some case where the surface seems to mess the ordering however this does not seem to affect the acquired knowledge.

4.2 Disappearing Clusters

In this section we apply the algorithm to a case in which clusters disappear. The simulated data set, Dset_{5d}, consists of 5-dimensional vectors, initially drawn from a finite mixture of 10 Gaussian distributions randomly positioned in a closed subset of \mathbb{R}^5 . From time step 3000, to time step 4000, 7 of the clusters disappear, gradually every 100 time steps. In Fig. 4, we display a contour plot of the reachability surface, as in this case it is more informative. The plot shows that micro-clusters start to disappear. However the surface finally stabilizes, resulting in a stable plot after time 4500.

4.3 The Forest CoverType Data

To examine the algorithm’s capabilities with real-world data we employ the Forest CoverType real world data set, obtained from the UCI machine learning repository [11]. This data set, Dset_{Forest}, is comprised of 581012 observations of 54 attributes, where

each observation is labeled as one of seven forest cover classes. We retain the 10 numerical attributes. In Fig. 5, the reachability surface is displayed, between time steps 2000 and 2200. Note that there is a peak in the plot around the 34th-36th micro-cluster in Fig. 5(a). To examine the class correspondence, in Fig. 5(b), the class of each point in the stream between time steps 2000 and 2200, is plotted against their nearest micro-cluster in the ordering, using a histogram. The latter figure shows that the correspondence of the classes changes at the 34th-36th micro-cluster as was anticipated by the peak present in the same position of the reachability surface. To obtain a wider view of how this structure evolves over time, in Fig 5(c) the reachability surface between times 2000 and 2500 is shown. Clearly, the data set has a somewhat stable clustering, mostly embodied in 3 different clusters, that is persistent through time.

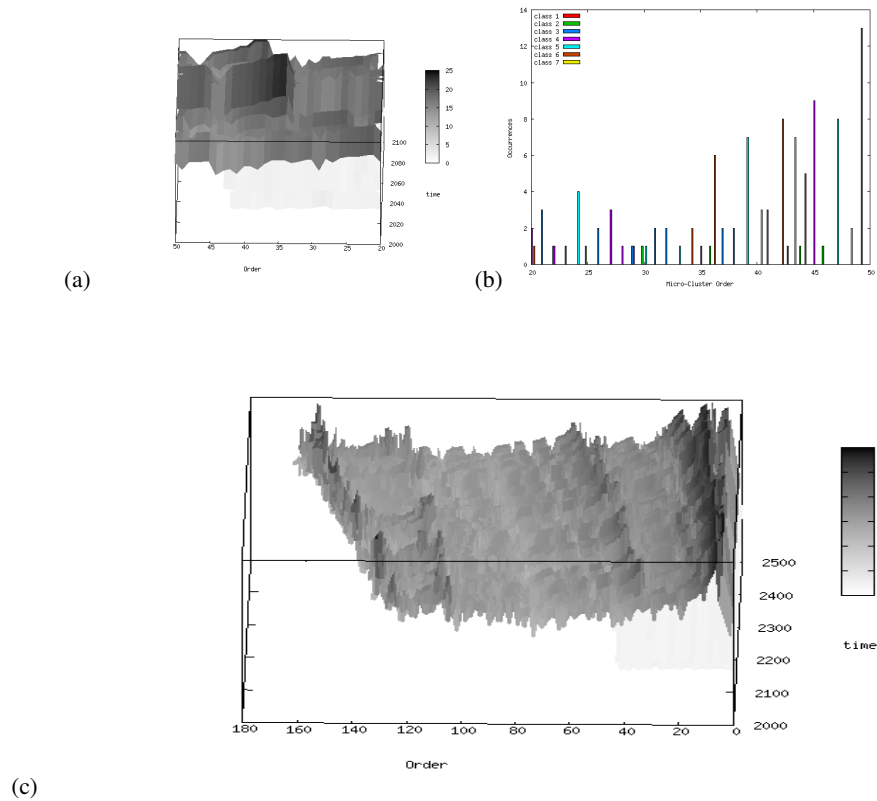


Fig. 5. The results on $Dset_{Forest}$.

5 Concluding Remarks

Clustering, as one of the most fundamental procedures for extracting information from data, plays a key role in the understanding of massive data streams. Clustering methods have recently been extended [2, 5, 13], to address some of the problems that emerge with streaming data. However, methods that can visualise the change of the clustering structure through time have only been investigated in lower dimensional situations or via projection [1].

In this work, we hybridise a stream clustering framework with an extension of OPTICS, a successful technique for the visualisation of static clustering [3]. The resulting method is shown through experimental investigation to provide insight into both the clustering structure and its evolution in time. The plots produced by our OpticsStream algorithm allow the user to identify the change in cluster structure in the case of both emerging and fading clusters. The results are also evaluated in a real world setting, where a-priori determined class information is used as a validation measure.

Acknowledgements

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership, under EPSRC grant EP/C548051/1.

References

1. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *ACM SIGMOD Conference*, 2003.
2. C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for projected clustering high dimensional data streams. In *10th VLDB conference*, 2004.
3. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1999.
4. M. Berthold and D.J. Hand. *Intelligent Data Analysis: an introduction*. Springer, 2003.
5. F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*, 2006.
6. I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.
7. D. J. Hand and N.A. Heard. Finding groups in gene expression data. *J Biomed Biotechnol.*, 2:215–225, 2005.
8. A. Hinneburg, DA Keim, and M. Wawryniuk. HD-Eye: visual mining of high-dimensional data. *Computer Graphics and Applications, IEEE*, 19(5):22–31, 1999.
9. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
10. H-P. Kriegel, P. Kroger, and I. Gotlibovich. Incremental OPTICS: Efficient computation of updates in a hierarchical cluster ordering. In *5th Int. Conf. on Data Warehousing and Knowledge Discovery*, 2003.

11. D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.
12. J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.
13. Dimitris K. Tasoulis, Niall M. Adams, and David J. Hand. Unsupervised clustering in streaming data. In *IEEE International Conference on Data Mining*, 2006.
14. C. Tryon. *Cluster Analysis*. Ann Arbor, MI: Edward Brothers, 1939.