Efficient Incremental Subspace Clustering in Data Streams

Maria Kontaki
Department of Informatics,
Aristotle University
54124, Thessaloniki, Greece
kontaki@delab.csd.auth.gr

Apostolos N. Papadopoulos
Department of Informatics,
Aristotle University
54124, Thessaloniki, Greece
apostol@delab.csd.auth.gr

Yannis Manolopoulos
Department of Informatics,
Aristotle University
54124, Thessaloniki, Greece
manolopo@delab.csd.auth.gr

Abstract

Performing data mining tasks in streaming data is considered a challenging research direction, due to the continuous data evolution. In this work, we focus on the problem of clustering streaming time series, based on the sliding window paradigm. More specifically, we use the concept of α-clusters in each time instance separately. A subspace α-cluster consists of a set of streams, whose value difference is less than α in a consecutive number of time instances (dimensions). The clusters can be continuously and incrementally updated as the streaming time series evolve. The proposed technique is based on a careful examination of pair-wise stream similarities for a subset of dimensions and then, it is generalized for more streams per cluster. Performance evaluation results show that the proposed pruning criteria are important for search space reduction, and that the cost of incremental cluster monitoring is computationally more efficient than reclustering.

1 Introduction

The study of query processing and data mining techniques for data stream processing has recently attracted the interest of the research community [4, 6, 10], due to the fact that many applications manage data that change very frequently with respect to time. Examples of such emerging applications are network monitoring, financial data analysis, sensor networks to name a few. The most important property of data streams is that new values are continuously arriving, and therefore efficient storage and processing techniques are required to cope with the high update rates.

Due to the highly dynamic nature of data streams, random access is prohibitive. Therefore, each data stream is possible to be read only once (or a limited number of times). This feature poses additional difficulties for query processing and data mining tasks.

Clustering is an important data mining task [9] and significant results have been reported for several data types. The challenge in a set of streaming time series is to update the clustering information as time progresses, avoiding the computationally intensive reclustering process.

Given a set of streaming time series, clustering can be applied to all available values within a specified length, known as the *sliding window*. The sliding window size defines the dimensionality of each streaming time series. For example, a sliding window of size 256 means that each time series is a 256-dimensional vector. Each dimension corresponds to a time instance. Searching for clusters in a large number of dimensions may result to failure, because as the size of the sliding window increases the probability that two streams will belong to the same cluster decreases. In many cases, although two or more streams do not belong to the same cluster for the whole sliding window, they do so by considering a subset of dimensions.

Figure 1 illustrates three streaming time series A, B and C with a sliding window of size 17. We assume that two streams belong to the same cluster if the difference of the values in the corresponding dimensions is less than or equal to 2. By inspecting Figure 1, it is evident that these streams can not belong to the same cluster, since the difference of values in several dimensions is more than 2. For example, the value difference of A and B in the second dimension is

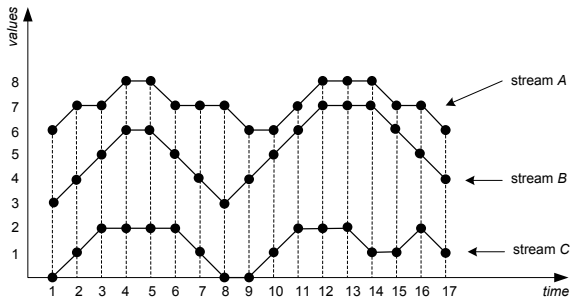


Figure 1. Example of subspace clustering.

$7 - 4 = 3$. However, by considering subsets of dimensions, streams A and B belong to the same cluster for the dimension intervals $[d_3, d_6]$, which contains d_3, d_4, d_5, d_6 and $[d_9, d_{17}]$, which contains d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{17}. It is evident that the value difference of streams A and B in each of these dimensions is less than or equal to 2.

The basic requirements for the generation of subspace clusters is that each cluster should contain a sufficient number of streams, in a sufficient number of consecutive dimensions. The generated subspace clusters contribute to the discovery of useful knowledge, since they reveal a high degree of similarity among streams participating in the same cluster.

The paper presents a methodology to attack the continuous subspace clustering problem by proposing the subspace α-cluster. We study effective algorithms towards efficient subspace α-cluster generation for a set of streaming time series. Towards this direction, we propose a method to update the clusters when new stream values become available, avoiding the process of reclustering. The generated α-clusters are defined only on consecutive dimensions. In summary, the contributions of this work is as follows:

(i) the study of the subspace clustering problem in streaming time series,

(ii) the study of continuous subspace clustering taking into account the time series evolution, and

(iii) the performance evaluation of the proposed method based on real-life and synthetic data sets.

The rest of the work is organized as follows. In Section 2 we discuss the appropriate related work. Section 3 studies in detail the proposed method for continuous clustering of streaming time series. Section 4 presents performance evaluation results, whereas Section 5 concludes the work.

2 Related Work

Clustering is a well studied research field in diverse disciplines with significant research contributions. In [5] it has been demonstrated that similarity search and clustering are *meaningless* for spaces that are embedded in a very large number of dimensions. This observation has lead a significant number of researchers to study alternative clustering methodologies. One research direction that has been followed is subspace clustering.

In [3], the authors have studied the problem of subspace clustering in a high-dimensional space and they have proposed CLIQUE, which is a grid-based bottom-up algorithm to discover density-based clusters. CLIQUE determines dense cells and merges them to create clusters in a high-dimensional space. In [7], the concept of entropy is used to determine a dense cell. The aforementioned methods apply to static data sets, and their adaptation to the streaming case is not obvious.

In [1, 2] the authors have proposed top-down algorithms for subspace cluster discovery. The basic drawback of these methods is the usage of parameter k, which is the number of subspace clusters that each method should report. In many real applications, this value is not known a priori.

Several research contributions have used δ-clusters to discover subspace clusters [8, 12, 13, 14]. However, the concept of δ-clusters is treated differently. In [8], δ-biclusters have been proposed to find subspace clusters in a set of genes and conditions of DNA microarrays. In [13], the pScore metric has been proposed to measure the coherence of a cluster. The method determines object and attribute pair-wise clusters and utilizes a prefix-tree to generate clusters in a high-dimensional space. The same metric has been used in [12] to find pair-wise clusters, along with a depth-first-search algorithm to prune redundant non-maximal clusters. In [14], it has been shown that the above methods do not scale well in large data sets, and therefore the authors have proposed the Counting Tree data structure, that provides a compact summary of the dense patterns. The above methods operate on non-evolving data sets. It is not straightforward to apply these methods for the streaming case, since they rely on algorithms that can not be easily adapted.

Recently, the problem of data stream clustering has attracted the research interest [6, 10]. The majority of these contributions apply the k-median clustering technique. The fundamental characteristic of the proposed methods is that they attack the problem of incremental clustering for the values of only one data stream. However, this is quite restrictive, taking into account that modern applications require the management of a large number of data streams. Moreover, in [11], the authors show that the clustering of the values of streaming time series is meaningless. Notice that our method groups incrementally steaming time series, that produced by different data streams, in clusters by using their values and doesn't group their values.

To the best of the authors' knowledge, this is the first attempt to solve the incremental subspace clustering problem in streaming time series.

3 Incremental Clustering

Table 1 summarizes the basic symbols and the corresponding definitions that are used throughout the study.

We begin our exploration with a number of basic definitions that are used for the rest of the work.

Definition 1 (*simple α-cluster*)
A *simple α-cluster* contains a number of streams with

Symbol	Description
s, s_i	a streaming time series
$s[i]$	the value of s in the i-th dimension
N	the number of streams
W	the size of the sliding window
C_i	a maximal subspace α-cluster
$c_{i,j}$	the j-th simple α-cluster of the i-th dimension
c, c'	simple α-clusters
m	number of streams in a cluster
G, G_i	a group of candidate α-clusters
$minRows$	minimum number of streams contained in a subspace α-cluster
$minCols$	minimum number of consecutive dimensions contained in a subspace α-cluster
α	maximum distance between any two streams for a given dimension

Table 1. Basic symbols used throughout the study.

pair-wise distances at most α in a single dimension. There is no restriction applied to the number of streams contained in each cluster. □

The j-th simple α-cluster in the i-th dimension is represented as $c_{i,j}$. The previous definition does not take into consideration possible restrictions applied to the number of streams in each cluster and the number of consecutive dimensions. By forcing each cluster to contain at least $minRows$ streams and at least $minCols$ dimensions we have:

Definition 2 (*subspace α-cluster*)
A *subspace α-cluster* contains at least $minRows$ streams, for which the maximum value difference is at most α in at least $minCols$ consecutive dimensions. □

In the example illustrated in Figure 1, assuming that $minRows$ = 2, $minCols$ = 3 and α = 2, we have two generated subspace α-clusters containing streams A and B, defined by the dimensions $[d_3, d_6]$ and $[d_9, d_{17}]$. However, assuming that $minCols = 5$, we have only one subspace α-cluster defined by the dimensions $[d_9, d_{17}]$.

A subspace α-cluster C is represented as a pair ($\mathcal{S}$,[d_i,d_j]), where $\mathcal{S}$ is a set of streams and [d_i,d_j] is an interval of $j - i + 1$ consecutive dimensions (time instances), where $i \leq j$. Evidently, the cardinality of $\mathcal{S}$ must be at least $minRows$, whereas the number of consecutive dimensions must be at least $minCols$. We assume that the streams contained in $\mathcal{S}$ are represented by their corresponding IDs. Furthermore, we assume that stream IDs are stored in $\mathcal{S}$ in a non-decreasing order.

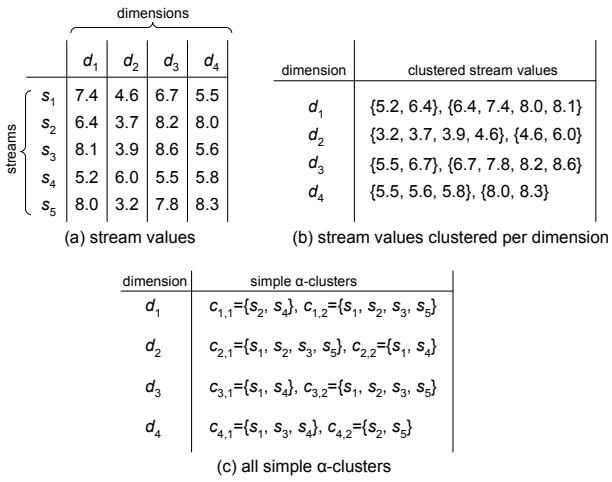
Definition 3 (*maximal subspace α-cluster*)
A subspace α-cluster ($\mathcal{S}$,[d_i,d_j]) is *maximal*, if a) we can not find another α-cluster ($\mathcal{S}$,[d_k,d_l]) such that $k \leq i$ and $l \geq j$ and b) we can not find another α-cluster ($\mathcal{T}$,[d_i,d_j]) such that $\mathcal{S} \subset \mathcal{T}$. □

We proceed with the detailed description of the proposed methodology, which attacks the following problem: **Given** a set of streaming time series, a maximum value difference α, a sliding window size W and two integer numbers $minRows$ and $minCols$, **determine** all maximal subspace α-clusters continuously, **where** each cluster contains at least $minRows$ streams, and the value difference is less than or equal to α, in at least $minCols$ consecutive dimensions.

The proposed methodology comprises the following phases: (i) the initialization phase, which determines an initial set of maximal subspace α-clusters, and (ii) a series of update phases which incrementally maintain the clusters when new stream values become available.

3.1 Cluster Initialization

The purpose of the cluster initialization (CI) is to determine an initial set of maximal subspace α-clusters, based on the last W values of each streaming time series. The CI process comprises a series of steps. In the first step, each time instance (dimension) is inspected separately to determine simple α-clusters (which are defined in one dimension only). Next, all clusters containing $m = 2$ streams in the maximum possible number of dimensions are generated. In each subsequent step the algorithm tries to increase the number of streams per cluster ($m = m + 1$), until all possible maximal subspace α-clusters are generated, according to the values of α, $minRows$ and $minCols$. Clusters that contain less than $minCols$ dimensions are discarded permanently in each step of the algorithm, since they can not contribute to the final answer.



	d_1	d_2	d_3	d_4
s_1	7.4	4.6	6.7	5.5
s_2	6.4	3.7	8.2	8.0
s_3	8.1	3.9	8.6	5.6
s_4	5.2	6.0	5.5	5.8
s_5	8.0	3.2	7.8	8.3

dimension	clustered stream values
d_1	{5.2, 6.4}, {6.4, 7.4, 8.0, 8.1}
d_2	{3.2, 3.7, 3.9, 4.6}, {4.6, 6.0}
d_3	{5.5, 6.7}, {6.7, 7.8, 8.2, 8.6}
d_4	{5.5, 5.6, 5.8}, {8.0, 8.3}

dimension	simple α-clusters
d_1	$c_{1,1}$={s_2, s_4}, $c_{1,2}$={s_1, s_2, s_3, s_5}
d_2	$c_{2,1}$={s_1, s_2, s_3, s_5}, $c_{2,2}$={s_1, s_4}
d_3	$c_{3,1}$={s_1, s_4}, $c_{3,2}$={s_1, s_2, s_3, s_5}
d_4	$c_{4,1}$={s_1, s_3, s_4}, $c_{4,2}$={s_2, s_5}

Figure 2. Cluster initialization.

We illustrate the CI process by means of an example,

which is depicted in Figures 2, 3 and 4. Assume that there are $N = 5$ streaming time series with a sliding window of size $W = 4$. Moreover, let $\alpha = 2$, $minRows = 4$ and $minCols = 3$. Figure 2(a) shows the value of each stream in every dimension, Figure 2(b) shows subsets of values that satisfy the α constraint, whereas Figure 2(c) shows the generated simple α-clusters for $\alpha = 2$.

To determine the simple α-clusters for each dimension we proceed as follows. The values in each dimension are sorted in a non-decreasing order. The produced sorted sequence S is processed by means of two pointers p_{left} and p_{right}. Initially, p_{left} and p_{right} are placed on the first element of the sorted sequence. The pointer p_{right} is continually increased until it reaches an element where $|S[p_{left}] - S[p_{right}]| > \alpha$. If this happens, then all elements $S[p_{left}]$, $S[p_{left}+1]$, ..., $S[p_{right}-1]$ form a cluster in the corresponding dimension. Then, the pointer p_{left} is increased by one, and the same process is applied until p_{right} reaches the end of the sorted sequence. If two clusters end at the same element, the one containing the minimum number of elements is discarded.

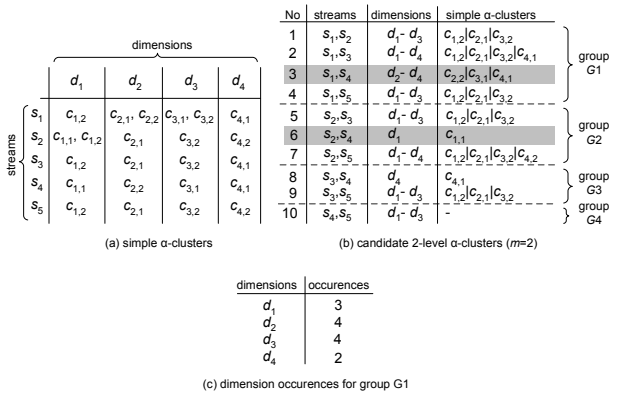


Figure 3. Cluster initialization (continued).

Following the generation of the initial set of simple α-clusters, the next step considers pairs of streams and determines if there are any simple α-clusters with two streams ($m = 2$). Figure 3(a) depicts the generated simple α-clusters for each dimension, whereas Figure 3(b) shows all possible 2-level clusters that are generated. Each 2-level cluster is formed by combining two streams that have common simple α-clusters, in each one of at least $minCols$ consecutive dimensions. The common simple α-clusters are illustrated in the fourth column of Figure 3(b). The candidate 2-level clusters are separated in four different groups, as it is indicated by the dashed lines in Figure 3(b). All candidate clusters in each group must share $m - 1$ streams and can differ in only the last one. Each group is treated separately, and therefore, we begin with the first group which is composed of candidate clusters containing stream s_1. Some of these clusters will be rejected, whereas the others will be used to form candidate 3-level clusters.

Proposition 1 (*cluster pruning criterion*)
If the number of candidate m-level clusters contained in a group is less than $minRows - m + 1$ then all the clusters in this group can be safely discarded from further consideration[1]. □

Evidently, all candidate clusters in the first group survive the cluster pruning criterion. At a first glance, it seems that all four clusters qualify, since each pair of streams contain at least three dimensions. However, with a more careful look we can see that dimension d_4 must be rejected. The following proposition explains.

Proposition 2 (*dimension pruning criterion*)
If each candidate α-cluster in a group G contains exactly m streams and the number of occurrences of a dimension in G is less than $minRows - m + 1$, then this dimension can not contribute to the generation of subspace α-clusters. □

If dimension pruning affects an existing cluster, either the cluster will be rejected, if the number of dimensions is less than $minCols$, or will shrink, if the number of dimensions is at least $minCols$. Applying the dimension pruning criterion to our case, it is evident that dimension d_4 has only two occurrences, (see Figure 3(c) and therefore must be rejected from further consideration. This means that cluster no.3 contains streams $\{s_1, s_4\}$ and dimensions d_2, d_3. However, since $minCols = 3$ this cluster is rejected (Figure 3(b)).

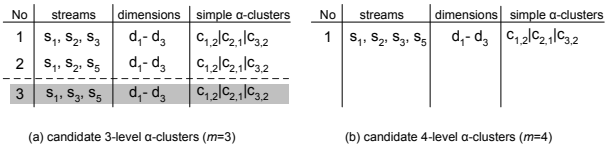


Figure 4. Cluster initialization (continued).

Next, the method tries to merge the clusters that survived the previous step, towards the generation of clusters containing $m + 1$ streams. Therefore we try to combine the clusters no.1 with no.2, no.1 with no.4 and no.2 with no.4 (recall that no.3 has been rejected). These combinations are depicted in tabular form in Figure 4(a). The clusters are categorized in two different groups. Each group should contain clusters that share all stream IDs, except the last. For example, candidate clusters no.1 and no.2 are contained in the first group since they differ in the last stream only, and they have two streams in common s_1 and s_2. Again, at a first

[1]The proofs of the propositions are omitted due to lack of space.

glance all three candidate clusters of Figure 4(a) qualify. However, cluster no.3 can be safely rejected, as it is suggested by the cluster pruning criterion (Proposition 1). This is illustrated by the shaded row in Figure 4(a).

By inspecting clusters no.1 and no.2 in the first group it is evident that both clusters survive both pruning criteria. Therefore, these two clusters can be combined towards the generation of a single 4-level cluster, which is illustrated in Figure 4(b). Recall, that $minRows = 4$ and $minCols = 3$. Therefore, this cluster is recorded as an answer, since it contains four streams and these streams form a subspace α-cluster in three dimensions.

Let us now check the second group of clusters depicted in Figure 3(b). The candidate cluster no.6 would never be created by the algorithm, since it does not satisfy the $minCols$ restriction. It is shown here only for demonstration purposes. This means that there are now only two candidate clusters in this group. According to the cluster pruning criterion these clusters should be discarded without any further consideration.

Up to this point, we have checked all candidate clusters of streams s_1 and s_2. Is it necessary to check the clusters for streams s_3, s_4 and s_5? Since there are three remaining streams it is impossible to generate a 4-level cluster ($minRows = 4$), as it is illustrated by the Proposition 3.

Proposition 3 (*stream pruning criterion*)
If the number of remaining streams is less than $minRows$ then all groups of candidate clusters generated by these streams can be safely discarded since it is impossible to give subspace α-clusters. □

Algorithm CI stops at this point and reports as an answer the cluster illustrated in Figure 4(b). Proposition 4 shows that it is not possible to miss any cluster.

Proposition 4 (*correctness of CI algorithm*)
By treating each group of candidate clusters separately, it is impossible to miss a maximal subspace α-cluster. □

Algorithm CI computes all maximal subspace α-clusters, by considering only candidate α-clusters which belong to the same group. This way, it is impossible to discover the same cluster more than once and therefore, less computational effort is required. The outline of the CI algorithm is depicted in Figure 5.

3.2 Cluster Maintenance

The purpose of the cluster maintenance (CM) phase is to keep the answers up to date. This phase is executed when new values for all the streams become available. Since processing is based on the sliding window paradigm, the left-most dimension should be discarded and a new one should be included. An example is illustrated in Figure 6(a), where stream values in dimension d_1 should be rejected, whereas stream values in dimension d_5 should be taken into consideration to update the clustering information. This requires the deletion of all simple α-clusters of dimension d_1 and the determination of all simple α-clusters for dimension d_5. These clusters are illustrated in Figure 6(b).

Algorithm CI ($\mathcal{S}$,α, $minRows$, $minCols$, W)
Input
$\mathcal{S}$: set of streams,
α: max value difference for a dimension in a cluster,
$minRows$: min number of streams per cluster,
$minCols$: min number of dimensions per cluster,
W: sliding window size
Output
$\mathcal{A}$: set of maximal subspace α-clusters

1. **for** i=1 **to** W
2. compute all simple α-clusters for dimension d_i;
3. **end for**
4. **for** i=1 **to** $N - minRows + 1$
5. set $m = 2$;
6. generate m-level candidate α-clusters for stream i;
7. apply cluster pruning;
8. apply dimension pruning;
9. **while** there exist m-level candidates **do**
10. generate $m + 1$-level candidate α-clusters that contain $minCols$ or more dimensions;
11. increase m;
12. **if** $m \geq minRows$ **and**
13. C is maximal subspace α-cluster **then**
14. update $\mathcal{A}$;
15. **end if**
16. apply cluster pruning;
17. apply dimension pruning;
18. **end while**
19. **end for**
20. report $\mathcal{A}$;

Figure 5. Outline of CI algorithm.

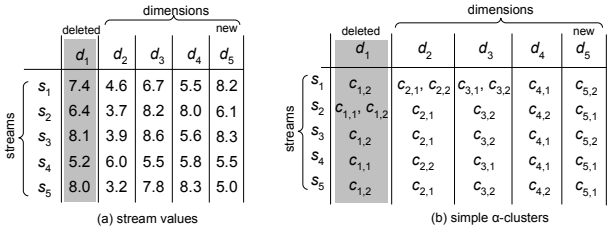


Figure 6. Arrival of dimension d_5.

Algorithm CM-UPALL ($\mathcal{S}$, α, $minRows$, $minCols$, W)
Input
$\mathcal{S}$: set of streams,
α: max value difference for a dimension in a cluster,
$minRows$: min number of streams per cluster,
$minCols$: min number of dimensions per cluster,
W: sliding window size
Output
$\mathcal{A}$: set of maximal subspace α-clusters

1. delete all the simple α-clusters of the first dimension;
2. find all the simple α-clusters for the new dimension;
3. update existing maximal subspace α-clusters;
4. delete the clusters that have less than $minCols$ dimensions;
5. **for** i=1 **to** $N - minRows + 1$
6. set $m = 2$;
7. generate m-level candidate α-clusters of stream i only for the last $minCols$ dimensions;
8. apply cluster pruning;
9. **while** there exist m-level candidates **do**
10. generate $m + 1$-level candidate α-clusters that contain $minCols$ dimensions;
11. increase m;
12. **if** $m \geq minRows$ **and**
13. C is maximal subspace α-cluster **then**
14. update $\mathcal{A}$;
15. **end if**
16. apply cluster pruning;
17. **end while**
18. **end for**
19. report $\mathcal{A}$;

Figure 7. Outline of CM-UPALL algorithm.

The cluster maintenance algorithm CM-UPALL, which is depicted in Figure 7, operates in two steps.

1. In the first step, existing maximal subspace α-clusters are checked, since some of them may be rejected due to the deletion of dimension d_1. Moreover, some of the existing clusters may be expanded by the inclusion of the newly created dimension d_5.

2. In the second step, the algorithm searches for new maximal subspace α-clusters that may be generated due to the arrival of the new dimension d_5.

Initially, each cluster containing d_4 as its right-most dimension is checked for possible expansion by adding dimension d_5. If the cluster can be expanded, it is included in the answer. Next, dimension d_1 is deleted from all clusters that contain it. If by deleting d_1 a cluster is left with less than $minCols$ dimensions, then it is deleted. Finally, the other clusters that are not affected by the deletion of d_1 and the inclusion of d_5 are considered part of the new answer. To search for new clusters that may have been formed due to the inclusion of dimension d_5, the algorithm inspects only the last $minCols$ dimensions (Proposition 5).

Proposition 5 (*correctness of CM-UPALL algorithm*)
Let d_{new} be the newly created dimension. To search for new clusters it is sufficient to study the last $minCols$ dimensions (i.e., $d_{new-minCols+1}$, $d_{new-minCols+2}$, ..., d_{new}). □

4 Performance Evaluation

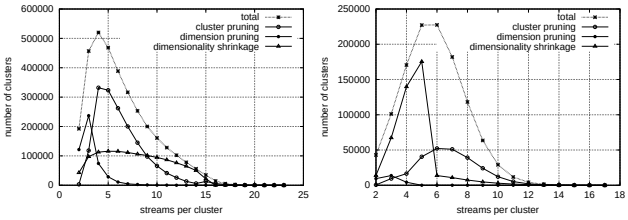
The proposed method has been implemented in C++ and all experiments have been conducted on a Pentium IV at 3.6 GHz, with 1 GB RAM, running Windows XP. In the sequel, we present the data sets that have been used in our experiments and the experimental results obtained by the performance study. The performance evaluation is based on the following data sets:

SYNTHETIC: The parameter values used (if not otherwise specified) are: the number of streams (N) is 5000, the sliding window (W) is 100, $\alpha = 0.0$, the number of embedded maximal subspace α-clusters is 100, and each one contains 50 streams in 10 dimensions.

STOCKS: The data set consists of a number of series denoting the closing prices of stocks and can be obtained from http://finance.yahoo.com. Each stock has been subdivided to a number of subseries of length 200, to obtain a total of 2313 different time series.

ECG: The data set contains electrocardiograms of two-channel recordings and can be obtained from the MIT-BIH Arrhythmia Database (http: //www.physionet.org/physiobank/database/mitdb/). Each channel was digitized at 360 samples per second. We chose an electrocardiogram of a sixty nine years old male, containing 650000 samples. To form the data set, we picked 30000 of out of the 650000 points randomly and each time series is formed from the consecutive 200 values of the selected point.

Initially, we examine the efficiency of the proposed pruning criteria. Recall that to generate m-level α-clusters, the $(m-1)$-level clusters are required. It can be shown that the total number of possible clusters that can be generated is $2^N - 1$, where N is the number of streaming time series. However, the application of the pruning criteria man-



(a) $\alpha = 0.2, minRows = 15, minCols = 3, W = 100$
(b) $\alpha = 0.2, minRows = 5, minCols = 6, W = 100$

Figure 8. Pruning power for STOCKS.

ages to reduce drastically the number of generated clusters. This effect is demonstrated in Figure 8, which depicts (1) the total number of clusters in each level, (2) the number of pruned clusters due to cluster pruning, (3) the number of pruned clusters due to dimension pruning and (4) the number of affected clusters by the dimensionality shrinkage. It is evident, that the majority of the candidate α-clusters is discarded. Cluster pruning is more significant when it happens in the first levels, since more clusters are pruned subsequently.

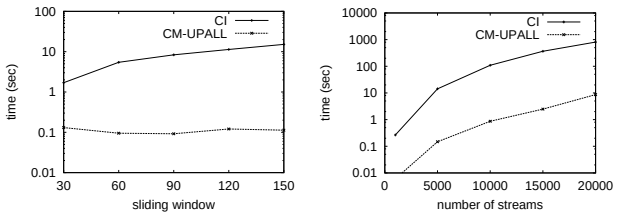


Figure 9. Response time vs (a) sliding window size and (b) number of streams.

Next, we study the scalability of the method with respect to the size of the sliding window (W) and the number of streams (N). The corresponding results are depicted in Figure 9. The graph has logarithmic scale along the y coordinate. We give both the initialization and the update time. Figure 9(a) illustrates the scalability of the method with respect to the sliding window size. Figure 9(b) depicts the scalability of the method with respect to the number of streams. In order to have a similar set up, we generated different synthetic data sets of 1000 to 20000 streams. In each data set, we embedded 100 maximal subspace α-clusters, but we varied the $minRows$ parameter so that the number of values used in the clusters to be proportional to the total number of values. In both cases, the cost of CI is more significant than that of CM-UPALL. It is evident that the incremental subspace clustering using the CM-UPALL procedure outperforms the reclustering process by applying the CI procedure every time an update occurs.

Next, we examine the relationship among the parameters $minRows$, $minCols$ and α. The sliding window size is set to 100. Figure 10(a) shows that the cost decreases as the parameters $minRows$ and $minCols$ increase their values. In Figure 10(b), it is shown that the cost decreases as $minRows$ increases and α decreases. A small α gives a large number of simple α-clusters and therefore, the probability that two streams will belong to the same simple α-cluster is reduced. Thus, the number of maximal subspace α-clusters decreases.

Finally, Table 2 shows the number of maximal subspace α-clusters for the ECG data set. The sliding window size is set to 100. The table depicts the number of clusters, the cost of cluster initialization phase, the number of clusters when some update operations have been performed and the average update time.

Figure 11 illustrates some of the clusters identified by the proposed algorithm in the ECG ($\alpha = 2$) data set. It is evident, that there is a high degree of similarity among streams belonging to the same cluster for the specific dimensions.

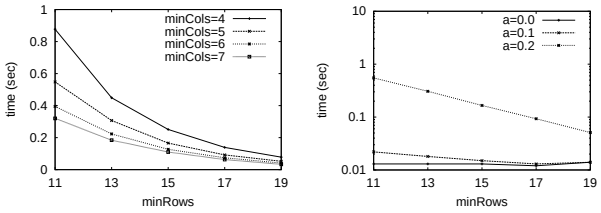


Figure 10. Response time vs (a) $minRows$ and $minCols$ (α fixed at 0.2) and (b) $minRows$ and α ($minCols$ fixed at 5) for STOCKS.

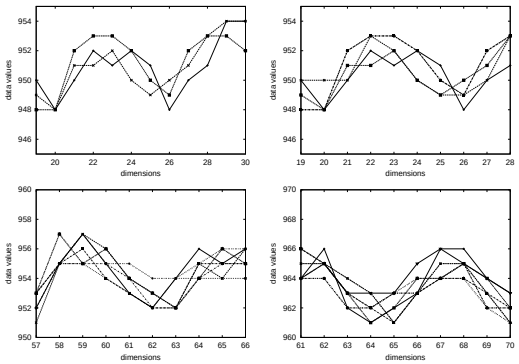


Figure 11. Examples of maximal subspace α-clusters for ECG data set.

			number of clusters and average update time after									
α	$minRows$	$minCols$	initialization		5 updates		10 updates		15 updates		20 updates	
0.0	30	3	984	2695.75	980	65.08	984	68.41	979	68.15	986	68.90
0.0	150	2	76	8211.28	75	65.96	72	64.60	71	86.40	69	76.88
1.0	10	9	335	1362.11	321	73.67	314	75.27	310	76.06	311	77.31
1.0	35	5	220	9879.22	209	219.32	203	231.61	201	231.60	194	237.57

Table 2. Number of clusters and average update time for ECG.

5 Conclusions

We have studied the problem of continuous subspace clustering in streaming time series data. More specifically, a novel method has been proposed towards efficient cluster generation and maintenance. Each cluster is composed of a number of streaming time series, where the pair-wise value difference inside a cluster is at most α, subject to the restrictions that the minimum number of streams is $minRows$ and the minimum number of dimensions is $minCols$. It has been demonstrated that by using the proposed pruning criteria, significant search space reduction is achieved.

Acknowledgments

Research supported by the Research Program PENED 2003, funded by the General Secretariat of Research and Technology (GSRT), Ministry of Development, Greece.

References

[1] C.C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, J.S. Park, "Fast Algorithms for Projected Clustering", *ACM International Conference on Management of Data*, pp.61-72, 1999.

[2] C.C. Aggarwal, P.S. Yu, "Finding Generalized Projected Clusters in High Dimensional Spaces", *ACM International Conference on Management of Data*, pp.70-81, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopoulos, P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Application", *ACM International Conference on Management of Data*, pp.94-105, 1998.

[4] S. Babu, J. Widom: "Continuous Queries over Data Streams", *ACM SIGMOD Record*, Vol.30, No.3, pp.109-120, 2001.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, "When is nearest neighbors meaningful", *International Conference on Database Theory*, pp.217235, 1999.

[6] M. Charikar, L. O'Callaghan, R. Panigrahy, "Better Streaming Algorithms for Clustering Problems", *Symposium on the Theory of Computing*, pp.30-39, 2003.

[7] C. Cheng, A.W. Fu, Y. Zhang, "Entropy-based Subspace Clustering for Mining Numerical Data", *ACM International Conference on Knowledge Discovery and Data Mining*, pp.84-93, 1999.

[8] Y. Cheng, G.M. Church, "Biclustering of Expression Data", *International Conference on Intelligent Systems for Molecular Biology*, pp.93-103, 2000.

[9] M.H. Dunham: "*Data Mining: Introductory and Advanced Topics*", Prentice Hall, 2002.

[10] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, "Clustering Data Streams: Theory and Practice", *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No.3, pp.515-528, 2003.

[11] J. Lin, E. Keogh and W. Truppel: "Clustering of Streaming Time Series is Meaningless", *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.

[12] J. Pei, X. Zhang, M. Cho, H. Wang, P.S. Yu, "MaPle: A Fast Algorithm for Maximal Pattern-based Clustering", *IEEE International Conference on Data Mining*, pp.259-266, 2003.

[13] H. Wang, W. Wang, J. Yang, P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets", *ACM International Conference on Management of Data*, pp.394-405, 2002.

[14] H. Wang, F. Chu, W. Fan, P.S. Yu, J. Pei, "A Fast Algorithm for Subspace Clustering by Pattern Similarity", *International Conference on Statistical and Scientific Database Management*, pp.51-60, 2004.