SNCStream: A Social Network-based Data Stream Clustering Algorithm

Jean Paul Barddal
Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
jean.barddal@ppgia.pucpr.br

Heitor Murilo Gomes
Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
hmgomes@ppgia.pucpr.br

Fabrício Enembreck
Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
fabricio@ppgia.pucpr.br

ABSTRACT

Data Stream Clustering is an active area of research which requires efficient algorithms capable of finding and updating clusters incrementally. On top of that, due to the inherent evolving nature of data streams, it is expected that these algorithms manage to quickly adapt to both concept drifts and the appearance and disappearance of clusters. Nevertheless, many of the developed two-step algorithms are only capable of finding hyper-spherical clusters and are highly dependant on parametrization. In this paper we introduce SNCStream, a one-step online clustering algorithm based on Social Networks Theory, which uses homophily to find non-hyper-spherical clusters. Our empirical studies show that SNCStream is able to surpass density-based algorithms in cluster quality and requires feasible amount of resources (time and memory) when compared to other algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms

Keywords

Data Stream Clustering, Concept Drift, Novelty Detection, Social Network Analysis

1. INTRODUCTION

Let $\mathcal{S}$ be a data stream providing instances $\vec{x}_i$ intermittently every t units of time, where $\vec{x}_i$ is a d-dimensional data object arriving at a timestamp i. It is assumed that $\mathcal{S}$ is unbounded, i.e. $|\mathcal{S}| \rightarrow \infty$, thus, it is not feasible to store all instances in memory before processing. This characteristic enforce algorithms to either process data in limited size chunks or to incrementally process instances. Firstly, every instance $\vec{x}_i$ must be processed before an instance $\vec{x}_{i+1}$ becomes available, otherwise instances start to accumulate and the algorithm may have to discard them. Secondly, there is an inherent temporal aspect associated with a stream process, where the data distribution may change over time, namely concept drift. Therefore, algorithms must also be able to detect and adapt to drifts, updating the algorithm's model. Also, ideally, clustering algorithms must be able to detect the appearance and disappearance of clusters, which may also happen with time, thus, discerning between seeds of new clusters and noisy data.

Generally, data stream clustering can be described as the act of grouping streaming data in meaningful clusters [5]. As any other task in data streams, clustering must be performed within limited time, memory and deal with data stream peculiarities, i.e. concept drift and clusters appearance and disappearance, namely concept evolution. Apart from the time and memory space constraints, two requirements are long-awaited for data stream clustering algorithms: no assumptions about the number of clusters and the discovery of clusters with arbitrary shapes, i.e. not only hyper-spherical.

In this paper we introduce SNCStream, a one-step online clustering algorithm capable of finding non-hyper-spherical clusters. SNCStream, in opposition to other data stream clustering algorithms, uses only one step processing to find clusters by using a social network generation and evolution model, which is based on homophily.

The remainder of this work is organized as follows: Section 2 surveys related work for data stream clustering. Section 3 presents basic concepts of social networks. In Section 4 we introduce the SNCStream algorithm. In Section 5, we present an empirical evaluation and discuss about parametrization sensibility. Section 6 concludes this paper and presents future work.

2. RELATED WORK

A variety of data stream clustering algorithms were developed in last years aiming two-step clustering (online and offline steps), such as CluStream [2], ClusTree [18] and DenStream [8].

During the online step, algorithms incrementally update specific data structures to deal with the evolving nature of data streams without compromising time and memory con-

SAC'15 April 13-17, 2015, Salamanca, Spain.

http://dx.doi.org/10.1145/2695664.2695674.

straints. One of the most widely used data structure is feature vectors, a triplet $CF = \langle LS, SS, N \rangle$, where LS stands for the sum of the objects summarized, SS is the squared sum of these objects and N is the amount of objects [21]. Feature vectors are able to represent hyper-spherical clusters incrementally due to its incremental and additive properties. A feature vector CF_j can be incremented by an instance $\vec{x}_i$ as follows: $LS_j \leftarrow LS_j + \vec{x}_i$, $SS_j \leftarrow SS_j + (\vec{x}_i)^2$ and $N_j \leftarrow N_j + 1$. As for the additive property, two feature vectors CF_i and CF_j can be merged in a third CF_l as follows: $LS_l \leftarrow LS_i + LS_j$, $SS_l \leftarrow SS_i + SS_j$ and $N_l \leftarrow N_i + N_j$. In addition, in order to provide recently retrieved instances higher importance in clustering, a variety of window models featuring sliding, damped and landmark were developed [21].

On the offline step, conventional clustering algorithms such as k-means [20] and DBSCAN [12] are feasible, since they do not handle the massive amount of instances, treating only these summarizing structures such as CFs.

In the next sections we present some literature algorithms based on online and offline steps.

2.1 CluStream

The CluStream adopts the landmark windowing technique, thus, treats the stream based on data chunks of size $\mathcal{H}$ [2]. CluStream assumes a number q of CFs that should be maintained at any instant of the stream. Initial CFs are computed with an amount of instances $\mathcal{N}$, also determined by the user. Instances obtained from the stream should be merged with existing CFs, or initiate new CFs. CluStream calculates an Euclidean distance for $\vec{x}_i$ to each CF, then, determines whether the distance to the closest CF_j is less or equal to its radius. In the positive case, $\vec{x}_i$ is merged within CF_j. Conversely, $\vec{x}_i$ starts a new CF_k. If the amount of CFs is above q, the two closest CFs are merged. When $\mathcal{H}$ is reached, all q CFs are recalculated with the next N instances obtained from the stream.

On the offline step, CluStream uses a modification of the k-means or DBSCAN algorithms, in order to obtain clusters based on the q CFs computed during the online step. In this paper, we compare the DBSCAN version, since k-means is highly dependant of the user-given parameter of ground-truth clusters K.

2.2 ClusTree

ClusTree [18] maintains CFs in a R-Tree [14]. R-Trees are data structures similar to B-Trees, yet, are used for multi-dimensional access, in this case, CFs. ClusTree creates a hierarchy of CFs in different granularity levels. Depending on how much time is available to process each instance, ClusTree performs a search in the R-Tree in order to find the most similar CF. Accordingly to user-given thresholds, it is determined whether this instance should or not be merged. In the negative case, a new CF is then created and added to the R-Tree. ClusTree also copes with noisy data by using outlier-buffers, which are CFs with low density.

In order to assign more importance to recent instances, ClusTree adopts an exponential decay function to assign weights for CFs. Since the R-Tree maintains in its nodes feature vectors, its components N, SS and LS are updated accordingly to an exponential decay function.

On the offline step, algorithms such as k-means and DBSCAN are used in order to detect clusters, where CFs centers are treated as centroids. As in CluStream, the compared version of ClusTree uses DBSCAN at the offline step to find clusters.

2.3 DenStream

DenStream is based on the DBSCAN algorithm, which guarantees the union of the ϵ-neighborhood of clusters which covers all dense areas of the attribute space. Therefore, a core object is an object which ϵ-neighborhood has at least ψ neighbors and a dense area is the union of all ϵ-neighborhoods of all core objects. DenStream defines the concept of a core-micro-cluster in a time instant t, which is a temporal extension to a CF, as $CMC(w, c, r)$ to a group of near instances $\vec{x}_i, \vec{x}_{i+1}, \ldots, \vec{x}_n$ where w is its weight, c its center and r its radius.

In evolutionary data streams, the role of clusters and noisy data may exchange, therefore, micro-clusters are incremented and updated also during concept drifts and evolutions. Thus, there are two types of micro-clusters: potential micro-cluster and outlier micro-cluster. A potential micro-cluster is similar to a core-micro-cluster, with the difference on the weight restriction, where $w \geq \beta\psi$ and $0 \leq \beta \leq 1$ is a threshold parameter to detect outliers. At the other hand, an outlier micro-cluster occurs when $w < \beta\psi$.

The online step of DenStream has the objective of maintaining a group of potential and outlier micro-clusters. At the arrival of each instance $\vec{x}_i$, DenStream tries to aggregate $\vec{x}_i$ to the closest potential micro-cluster accordingly to the weight restrictions. In the negative case, the same occurs for outlier micro-clusters. If $\vec{x}_i$ was aggregated in an outlier micro-cluster, the weight restriction is checked to determine whether this micro-cluster should be promoted to a potential micro-cluster. Conversely, if $\vec{x}_i$ was not merged with any micro-cluster at all, it starts a new outlier micro-cluster.

The offline step of DenStream uses the DBSCAN algorithm to find clusters based on potential micro-clusters computed during the online step.

3. SOCIAL NETWORKS

Social Networks Theory has been applied in many research fields, from computer science to sociology, due to its formal description of structural variables based on graph theory.

Even though Social Network Analysis focus on subjective topics such as an individual behavior in society, its building blocks (nodes and edges) are represented computationally as a graph $G = (V, E, W)$ where V is the set of nodes, E the set of edges between nodes and W is the set of tuples which associates to each edge in E a weight (Euclidian distance).

Different Social Network models were developed over the years, with the objective of modeling both generation and evolution of networks, where we emphasize: Random [11], Small-World [23] and Scale-free [4].

The Random generation model is based on the hypothesis that the existence of a connection between a pair of nodes is given by a probability p.

The Small-world model incorporates attributes of both random and regular networks. Consequently, this topology presents a high clustering coefficient, inherited from the regular networks, and a small average path length, as random networks [23].

The objective of the Scale-free model is to represent the dynamics of real networks, where connections between nodes can be replaced with time such as the World Wide Web

and Cellular Networks [4]. Thus, authors in [4] developed generation and evolution elements. A Scale–free network starts with a diminished network size n. For every time unit t, a new node is added to the network establishing k connections with already existing nodes. Also, on the Scale-free model, when choosing the nodes which this new node will establish its connections, it is assumed a probability $\prod(d_i) = \frac{d_i}{\sum_j d_j}$, where d represents the degree of a node i.

As a consequence of the preferential attachment process, Scale–free networks are "dominated" by a few vertices denominated hubs [10]. Thus, Scale–free network degree distribution follows the distribution $p(k) \sim k^{(-\lambda)}$ where $p(k)$ is the probability of a random node being attached to k other nodes and $\{\lambda \in \mathbb{R} \mid 2 \leq \lambda \leq 3\}$ for many real networks [4]. In addition, at each time unit t, besides the addition of new nodes, a rewiring process exists. The rewiring component is based on the homophily definition [22], where nodes tend to eradicate connections with dissimilar nodes, replacing them by new connections with more similar ones with a probability θ. Since homophily tends to establish connections between similar nodes, this process is useful for the task of clustering.

Algorithm 1: Insertion Procedure Pseudocode.

```
input : a node (instance or CF) v_i, the network G and the
        amount of connections each micro-cluster will
        establish at its insertion in the network k.
neighbors ← the k closest microclusters in V to v_i;
V ← V ∪ {v_i};
foreach p_i ∈ neighbors do
    newEdge ← ⟨v_i, p_i⟩;
    E ← E ∪ {newEdge};
    W ← W ∪ {⟨newEdge, d(v_i, p_i)⟩};
/* rewiring procedure                                  */
foreach p_i ∈ V do
    k_i ← degree(p_i);
    newNeighbors ← the k_i closest nodes in V to p_i using a
    2-hop neighborhood;
    Remove all edges connecting p_i from E and its
    correspondent weights from W;
    foreach p_j ∈ newNeighbors do
        /* Establishes new edges                       */
        newEdge ← ⟨p_i, p_j⟩;
        E ← E ∪ {newEdge};
        W ← W ∪ {⟨newEdge, d(p_i, p_j)⟩};
```

4. SNCSTREAM

Social Network Clusterer Stream (SNCStream) is based on the hypothesis that intra-cluster data are related due to diminished dissimilarity and inter-cluster data are not related, due to higher dissimilarity. SNCStream models this problem as a social network, where the set of nodes V are instances or micro-clusters, edges E represent connections between these nodes, and subgroups in this network represent clusters. In order to keep track of clusters during the stream, SNCStream uses an adaptation of the Scale-free model rewiring process where nodes rewirings are based on homophily.

Initially, SNCStream starts with an empty network $G = (V, E, W)$, which is constructed with instances $\vec{x}_i$ obtained from $\mathcal{S}$. For each instance $\vec{x}_i$, SNCStream finds its k closest instances by computing Euclidian distances, where k is a user-given parameter.

Afterwards, $\vec{x}_i$ is added to V and edges and weights with the k nearest neighbors are added to its correspondent sets E and W accordingly to Algorithm 1. After $\vec{x}_i$ addition to the network, all nodes $v_i \in V$ perform rewirings based on homophily, thus v_i replaces edges with higher dissimilarities w with edges with closest neighbors, which account for lower dissimilarity. Every v_i then computes Euclidian distances with all of its 2-hop neighbors. A 2-hop neighborhood is assumed since potential closest nodes are likely to be neighbors of the current neighbors. This 2-hop neighborhood is an approximation in order to prevent distance computation between all nodes, which would be computationally costly, thus, may fail in isolated scenarios. With the results of these Euclidian distances, v_i replaces edges with the most dissimilar instances with some similar ones, yet, maintaining its degree k_i.

Due to the rewiring process, communities of instances tend to appear naturally since the amount of intra-clusters edges between similar instances grows and of dissimilar instances (between clusters), shrinks. Figure 1 presents the evolution of a network as instances arrive, where one can see that the rewiring procedure enlarges the amount of intra-cluster edges and diminishes the amount of inter-clusters connections. This procedure is repeated until, in Figure 1n, two clusters emerge.

Nonetheless, this method alone, besides composing and keeping track of clusters in online fashion, is not practical for data streams due to the massive amount of data. Consequently, SNCStream encompasses a window N which determines the initial amount of instances to be treated before the network of instances is replaced by a potential micro-cluster network. Potential micro-clusters and outlier micro-clusters are defined as in [8].

When the network size $|V|$ reaches N, all instances are transformed into potential micro-clusters and an outlier micro-cluster buffer *omc* is initialized. At this point, instances $\vec{x}_i$ retrieved from $\mathcal{S}$ are treated by SNCStream similarly to DenStream. Firstly, SNCStream tries to merge $\vec{x}_i$ into an existent micro-cluster. SNCStream computes an Euclidian distance with each $v_j \in V$, determining the closest micro-cluster to $\vec{x}_i$: *cp*. If the radius of $cp + \vec{x}_i$ is below ϵ, $\vec{x}_i$ is added to *cp*, otherwise, SNCStream repeats the above steps for the outlier micro-clusters in an outlier buffer *omc*. In the negative case, i.e. $\vec{x}_i$ is not merged within any existing micro-clusters, $\vec{x}_i$ initiates a new outlier micro-cluster in *omc*.

When an outlier micro-cluster *oc* is promoted to a potential micro-cluster (weight of *oc* is greater or equal $\beta\psi$), it is removed from *omc* and inserted in the network G using Algorithm 1.

In order to provide recently retrieved data higher weights, SNCStream uses a damped window model, where all CFs' components LS, SS and N are updated accordingly to an exponential decay function.

As in DenStream [8], the weight $w(v_i)$ decreases exponentially over time using a function $f(t) = 2^{-\alpha \times t}$, where $\alpha > 0$ is the decaying factor. When the weight $w(v_i)$ of a micro-cluster v_i is below $\beta\psi$, it is removed from the network (if v_i is a potential micro-cluster), or from *omc* (if v_i is an outlier micro-cluster). In the first case, all neighbors v_j of v_i are allowed to rewire in order to maintain their degree k_j after the v_i's removal. This early rewiring procedure is performed

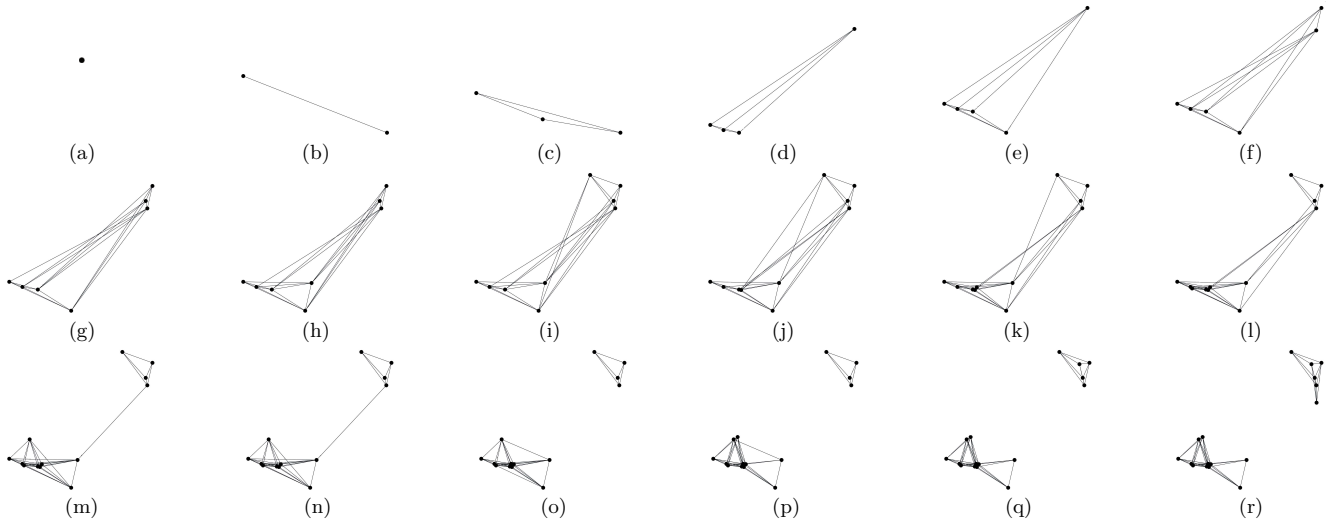

Figure 1: Example of network evolution obtained from the RBF_2 experiment from instances 1 through 18.

Algorithm 2: Rewiring for Removal Procedure Pseudocode.

input : a node v_j to rewire, a node v_i which is about to be removed and the network G.

`/*` v_j `is rewired to its` k_j `closest nodes, with the exception of` v_i`, using a 2-hop neighborhood.` `*/`

1 $k_j \leftarrow$ degree(v_j);
2 *newNeighbors* $\leftarrow$ the k_j closest micro-clusters in V to v_j with the exception of v_i in a 2-hop neighborhood;
3 Remove all edges connecting v_j from E and its correspondent weights from W;
4 Repeat steps 11 through 14 from Algorithm 1.

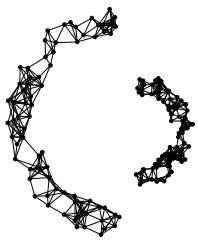

Figure 2: Snapshot of the network taken from the banana dataset toy problem.

in order to keep the network with a higher amount of edges. Algorithm 2 presents the pseudocode for the early removal rewiring procedure.

Again, as a result of the rewiring process, each CF will establish connections with the most similar CFs in G, thus, clusters are formed at the online step without any batch clustering algorithm such as k-means or DBSCAN. When a user requests for cluster retrieval, SNCStream returns the subgroups of the network. Also, since clusters are formed with no hyper-spherical constraints, SNCStream is able to find non-hyper-spherical clusters such as presented in Figure 2 where one can see two non-hyper-spherical clusters.

One could argue about the effects of the parameter k on the generation and evolution of the network, therefore, in Section 5.2, we discuss about the parameter sensitivity and show that $k = 4$ is an appropriate value for tested domains.

Experiment	Instances	Attributes	Reference
Airlines	539,383	8	[16]
Electricity	45,312	8	[15]
Forest Covertype	581,012	54	[17]
KDD'98	95,412	56	[18]
KDD'99	4,898,431	38	[3]

Table 1: Real datasets used for empirical evaluation.

Experiment	CMM			
	CluStream	ClusTree	DenStream	SNCStream
RBF_2	0.84	0.91	0.81	**0.98**
RBF_5	0.67	0.85	0.73	**0.98**
RBF_{10}	0.55	0.88	0.71	**0.97**
RBF_2^*	0.83	0.91	0.82	**0.97**
RBF_5^*	0.67	0.85	0.71	**0.94**
RBF_{10}^*	0.57	0.85	0.70	**0.89**
Airlines	0.67	0.76	0.47	**0.96**
Electricity	0.41	0.44	0.49	**0.96**
Forest Covertype	0.50	0.49	0.69	**0.90**
KDD'98	0.37	0.37	0.37	**0.38**
KDD'99	0.50	0.49	0.69	**0.90**
Average Ranking	*3.45*	*2.36*	*3.18*	*1.00*

Table 2: CMM (Cluster Mapping Measure) obtained from experiments.

5. EMPIRICAL EVALUATION

In order to compare SNCStream with other algorithms, we developed a validation environment using both real and synthetic data. All algorithms are compared in terms of CMM, CPU Time and RAM-Hours. CMM is an external clustering evaluation metric that accounts for non-associated, misassociated instances and noisy data inclusion [19]. Also, CMM considers recently retrieved instances with more weight than older ones by using an exponential decay function inside evaluation windows. RAM-Hours is a measure of RAM usage, where each Gigabyte of RAM dispended for one hour

Experiment	CPU Time (s)			
	CluStream	ClusTree	DenStream	SNCStream
RBF_2	288.54	**175.69**	660.37	694.95
RBF_5	295.45	**147.39**	629.6	373.2
RBF_{10}	440.37	**187.99**	959.03	438.57
RBF_2^*	300.13	**185.62**	645.02	661.12
RBF_5^*	297.84	**150.29**	598.98	317.09
RBF_{10}^*	441.99	**192.93**	902.51	422.15
Airlines	278.55	**137.89**	246.84	167.68
Electricity	33.32	**13.31**	47.55	51.85
Forest Covertype	598.54	327.59	418.67	**300.17**
KDD'98	1783.24	1261.51	**1104.74**	1188.12
KDD'99	908.58	**265.72**	7944.89	2450.1
Average Ranking	*2.72*	*1.27*	*3.27*	*2.72*

Table 3: CPU Time (s) obtained from experiments.

Experiment	RAM-Hours			
	CluStream	ClusTree	DenStream	SNCStream
RBF_2	$\mathbf{1.70 \times 10^{-6}}$	7.60×10^{-5}	1.82×10^{-5}	3.54×10^{-5}
RBF_5	$\mathbf{2.10 \times 10^{-6}}$	6.65×10^{-5}	1.96×10^{-5}	1.51×10^{-5}
RBF_{10}	$\mathbf{4.05 \times 10^{-6}}$	9.84×10^{-5}	3.97×10^{-5}	1.59×10^{-5}
RBF_2^*	$\mathbf{1.76 \times 10^{-6}}$	5.93×10^{-5}	1.67×10^{-5}	3.71×10^{-5}
RBF_5^*	$\mathbf{2.12 \times 10^{-6}}$	5.74×10^{-5}	1.87×10^{-5}	1.34×10^{-5}
RBF_{10}^*	$\mathbf{4.06 \times 10^{-6}}$	1.01×10^{-4}	3.60×10^{-5}	1.65×10^{-5}
Airlines	$\mathbf{1.75 \times 10^{-6}}$	6.38×10^{-5}	7.23×10^{-6}	6.18×10^{-6}
Electricity	$\mathbf{2.65 \times 10^{-7}}$	6.84×10^{-6}	1.60×10^{-6}	1.71×10^{-6}
Forest Covertype	$\mathbf{5.50 \times 10^{-6}}$	2.22×10^{-4}	1.75×10^{-5}	1.21×10^{-5}
KDD'98	3.14×10^{-4}	1.06×10^{-2}	9.87×10^{-4}	$\mathbf{1.57 \times 10^{-4}}$
KDD'99	$\mathbf{1.74 \times 10^{-5}}$	2.21×10^{-4}	6.88×10^{-4}	8.82×10^{-5}
Average Ranking	*1.09*	*3.91*	*2.82*	*2.18*

Table 4: RAM-Hours obtained from experiments.

equals on RAM-Hour [6]. All experiments were performed on a Intel Xeon CPU E5649 @ 2.53GHz ×8 based computer running CentOS with 16GB of memory.

All algorithms were parametrized accordingly to their original papers. CluStream parameters are: Horizon $\mathcal{H} = 1,000$ and $q = 1,000$ [3]. ClusTree parameters are: a Horizon $\mathcal{H} = 1,000$ and a maximum tree height = 8 [18]. DenStream parameters are: $\psi = 1$, $\mathcal{N} = 1,000$, $\lambda = 0.25$, $\epsilon = 0.02$ e $\beta = 0.2$ [8]. All of the above cited algorithms run a DBSCAN at the offline step, using the following parameters: $\psi = 1$, $\mathcal{N} = 1,000$, $\lambda = 0.25$, $\epsilon = 0.02$, $\beta = 0.2$ and a offline multiplier $\eta = 2$. Finally, SNCStream parameters are: $\psi = 1$, $\mathcal{N} = 100$, $\lambda = 0.25$, $\epsilon = 0.02$, $\beta = 0.2$ and $k = 4$. The initial window size $\mathcal{N}$ differs from other algorithms since SNCStream finds its initial micro-clusters based on $\mathcal{N}$ while others do not.

Synthetic data streams were generated using the Radial Basis Function (RBF) generator available at MOA framework [7]. The RBF generator creates a user-given amount of drifting centroids, which are defined by a label, center, weight and standard deviation accordingly to a Gaussian distribution. Another possibility of the RBF generator is the appearance or disappearance of clusters.

For this evaluation we created 6 synthetic data streams with 1,000,000 instances varying the dimensionality of instances $d = \{2, 5, 10\}$, namely RBF_2, RBF_5 and RBF_{10} where the amount of clusters is fixed in 5; and RBF_2^*, RBF_5^* and RBF_{10}^* for experiments where the amount of clusters vary in the $[2; 8]$ domain.

Jutting synthetic data streams, algorithms were evaluated using real datasets where clusters are known to be non-hyper-spherical. These datasets are commonly used for supervised learning and we assume classes as ground-truth clusters. These datasets are listed at Table 1. All experiments were evaluated every 1,000 instances.

5.1 Discussion

Since all evaluated algorithms are deterministic, its results do not fulfill the requirements for parametric hypothesis testing. Therefore, a combination of non-parametric test was used combining Friedman's [13] and Bonferroni-Dunn's post-hoc $1 \times N$ pivotal test [1], using a confidence level $\alpha = 0.05$. The combination of these two tests are due its corrections provided to diminish type II errors [9].

Table 2 presents results for CMM, where one can see that SNCStream outperformed all algorithms for the tested experiments. In order to determine whether there is significant statistical difference between algorithms' CMM, we used Friedman's test. Thus, we were able to determine that there is statistical difference between algorithms in terms of CMM. Therefore, we used Bonferroni-Dunn's $1 \times N$ test pivoting SNCStream and determined that it is superior to all others with a 95% confidence level.

In Table 3 one can see results for CPU Time in seconds where ClusTree shows better average results due to its tree structure, where comparisons are made in $\mathcal{O}(\log n)$. Nevertheless, Friedman's test pointed that there is statistical difference between algorithms and Bonferroni-Dunn's procedure determined that ClusTree outperforms DenStream but not CluStream and SNCStream.

Finally, Table 4 presents results for RAM-Hours, where one can see that CluStream presents better results, except on the KDD'98 experiment. Again, by using Friedman's test and Bonferroni-Dunn's procedure, we were able to determine that CluStream outperforms ClusTree and DenStream in terms of RAM-Hours, yet, not SNCStream.

5.2 Parameter Sensibility

SNCStream reckons on one parameter to find clusters: the amount of connections established at the arrival of each instance, k; while other parameters are used only to form micro-clusters. Therefore, to determine whether different values of k affect results directly, we ran all experiments varying k in the $[1; 10]$ interval. Table 5 presents results obtained for SNCStream varying the value of k, where one can see that SNCStream presents good results within the whole range, yet, $k = 4$ has better average results. In order to determine whether $k = 4$ is superior that other configurations, we ran the same combination of non-parametric tests. Thus, we were able to determine that $k = 4$ has no statistical difference between other values of k, with the exception where $k = 1$. Based on these results, we assume $k = 4$ as default value, although optimization for other domains can be achieved by varying this parameter.

6. CONCLUSION

In this paper we presented the SNCStream, a one-step data stream clustering algorithm capable of finding non-hyper-spherical clusters. SNCStream uses a scale-free-like homophily procedure to track the evolution of clusters during data streams. SNCStream was evaluated along with other clustering algorithms in a variety of real and synthetic datasets, which showed that SNCStream achieved higher CMM while maintaining suitable CPU Time and RAM-Hours when compared to other algorithms with a lower amount of parameters. In addition, SNCStream is not bounded to a user-given amount of ground-truth clusters to be found.

Experiment	CMM									
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
RBF_2	0.96	0.96	**0.98**	**0.98**	0.97	0.96	0.96	0.95	0.94	0.94
RBF_5	0.93	0.95	0.97	**0.98**	0.97	0.95	0.94	0.90	0.86	0.81
RBF_{10}	0.93	0.95	**0.97**	**0.97**	0.95	0.93	0.90	0.87	0.82	0.75
RBF_2^*	0.95	0.95	**0.98**	0.97	0.96	0.95	0.94	0.93	0.92	0.90
RBF_5^*	0.92	0.94	**0.96**	0.94	0.92	0.89	0.85	0.80	0.75	0.70
RBF_{10}^*	0.91	0.91	**0.92**	0.89	0.85	0.81	0.76	0.70	0.66	0.63
Airlines	0.77	0.89	0.94	0.96	0.97	0.97	**0.98**	**0.98**	**0.98**	**0.98**
Electricity	0.69	0.85	0.93	0.96	0.96	0.97	0.98	0.98	**0.99**	**0.99**
Forest Covertype	0.53	0.75	0.84	0.89	0.91	0.93	0.94	0.94	0.95	**0.96**
KDD'98	0.37	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**
KDD'99	0.80	0.82	0.88	0.90	0.92	0.94	0.94	0.94	**0.95**	**0.95**
Average Ranking	*7.54*	*6.00*	*4.18*	*4.09*	*4.82*	*4.82*	*5.18*	*5.81*	*6.36*	*6.18*

Table 5: CMM obtained by varying k for SNCStream algorithm.

In future works we plan on verifying the impact of other distance metrics for higher dimensionality data, perform a further discussing regarding other parameters and network aspects such as topology and centrality metrics. Also, we expect to use archive programming to optimize distance computation and develop a specific graph implementation to reduce processing time and memory usage. In addition, further discussion with other evaluation metrics and algorithms is also envisioned.

7. REFERENCES

[1] H. Abdi. Bonferroni and Sidak corrections for multiple comparisons. In N. J. Salkind, editor, *Encyclopedia of Measurement and Statistics*, pages 103–107. Sage, Thousand Oaks, CA, 2007.

[2] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 575–586, New York, NY, USA, 2003. ACM.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003.

[4] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. In *Reviews of Modern Physics*, pages 139–148. The American Physical Society, January 2002.

[5] A. Amini and T. Y. Wah. On density-based data streams clustering algorithms: A survey. *Journal of Computer Science and Technology*, 29(1):116–141, 2014.

[6] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[7] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[8] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, pages 328–339, 2006.

[9] G. Corder and D. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, 2011.

[10] C. D. Correa, T. Crnovrsanin, and K.-L. Ma. Visual reasoning about social networks using centrality sensitivity. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):106–120, 2012.

[11] P. Erdos and A. Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *KDD*, pages 226–231. AAAI Press, 1996.

[13] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, Dec. 1937.

[14] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, pages 47–57, New York, NY, USA, 1984. ACM.

[15] M. Harries and N. S. Wales. Splice-2 comparative evaluation: Electricity pricing, 1999.

[16] E. Ikonomovska, J. Gama, B. Zenko, and S. Dzeroski. Speeding-up hoeffding-based regression trees with options. In *ICML*, pages 537–544, 2011.

[17] P. Kosina and J. a. Gama. Very fast decision rules for multi-class problems. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 795–800, New York, NY, USA, 2012. ACM.

[18] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: Indexing micro-clusters for anytime stream mining. *Knowl. Inf. Syst.*, 29(2):249–272, Nov. 2011.

[19] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer. An effective evaluation measure for clustering on evolving data streams. In *Proc. of the 17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD 2011), San Diego, CA, USA*, pages 868–876, New York, NY, USA, 2011. ACM.

[20] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, Sept. 1982.

[21] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. a. Gama. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013.

[22] M. Van Steen. *Graph Theory and Complex Networks: An Introduction*. Maarten Van Steen, 2010.

[23] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.