Graph sketching-based Massive Data Clustering

Anne Morvan [*1,2], Krzysztof Choromanski[3], Cédric Gouy-Pailler[1], and Jamal Atif[2]

[1]CEA, LIST, 91191 Gif-sur-Yvette, France
[2]Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France
[3]Google Brain Robotics, New York, USA

February 24, 2017

Abstract

In this paper, we address the problem of recovering arbitrary-shaped data clusters from massive datasets. We present DBMSTClu a new density-based non-parametric method working on a limited number of linear measurements i.e. a *sketched* version of the similarity graph G between the N objects to cluster. Unlike k-means, k-medians or k-medoids algorithms, it does not fail at distinguishing clusters with particular structures. No input parameter is needed contrarily to DBSCAN or the Spectral Clustering method. DBMSTClu as a graph-based technique relies on the similarity graph G which costs theoretically $O(N^2)$ in memory. However, our algorithm follows the dynamic semi-streaming model by handling G as a stream of edge weight updates and sketches it in one pass over the data into a compact structure requiring $O(\text{poly} \log(N))$ space. Thanks to the property of the Minimum Spanning Tree (MST) for expressing the underlying structure of a graph, our algorithm successfully detects the right number of non-convex clusters by recovering an approximate MST from the graph sketch of G. We provide theoretical guarantees on the quality of the clustering partition and also demonstrate its advantage over the existing state-of-the-art on several datasets.

1   Introduction

Clustering is one of the principal data mining tasks consisting in grouping related objects in an unsupervised manner. It is expected that objects belonging to the same cluster are more similar to each other than to objects belonging to different clusters. There exists a variety of algorithms performing this task. Methods like k-means Lloyd [1982], k-medians Jain and Dubes [1988] or k-medoids Kaufman and Rousseeuw [1987] are useful unless the number and the shape of clusters are unknown which is unfortunately often the case in real-world applications. They are typically unable to find clusters with a non-convex shape. Although DBSCAN Ester et al. [1996] does not have these disadvantages, its resulting clustering still depends on the chosen parameter values.

One of the successful approaches relies on a graph representation of the data. Given a set of N data points $\{x_1, ..., x_N\}$, a graph can be built based on the similarity of data where points of the dataset are the vertices and weighted edges express distances between these objects. Besides, the dataset can be already a graph G modeling a network in many fields, such as bioinformatics - where gene-activation dependencies are described through a network - or social, computer, information, transportation network analysis. The clustering task consequently aims at detecting clusters as groups of nodes that are densely connected with each other and sparsely connected to vertices of other groups. In this context, Spectral Clustering

---

Nascimento and de Carvalho [2011] is a popular tool to recover clusters with particular structures for which classical k-means algorithm fails. When dealing with large scale datasets, a main bottleneck of the technique is to perform the partial eigendecomposition of the associated graph Laplacian matrix, though. An other inherent difficulty is to handle the huge number of nodes and edges of the induced similarity graph: storing all edges can cost up to $O(N^2)$ where N is the number of nodes. Over the last decade, it has been established that the dynamic streaming model Muthukrishnan [2005] associated with linear sketching techniques Ahn et al. [2012a] - also suitable for distributed processing -, is a good way for tackling this last issue.

Contributions We present here a new clustering algorithm DBMSTClu providing a solution to the following issues: 1) detecting arbitrary-shaped data clusters, 2) with no parameter, 3) in a space-efficient manner by working on a limited number of linear measurements, a *sketched* version of the similarity graph G. To deal with the space constraints, our algorithm follows the dynamic semi-streaming model by handling G as a stream of edge weight updates and sketches it in only one pass over the data into a compact structure taking $O(\text{poly} \log(N))$ space. Thanks to the property of the Minimum Spanning Tree (MST) for expressing the underlying structure of a graph, our algorithm automatically identifies the right number of non-convex clusters by recovering an approximate MST from the graph sketch of G.

The remaining of this paper is organized as follows. In Section 2 we describe the related work about graph clustering and sketching. DBMSTClu (DB for Density-Based), our MST-based algorithm for clustering is then explained in Section 3. Section 4 presents the experimental results comparing our proposed clustering algorithm to other existing methods. Finally, section 5 concludes our work and discusses future directions.

2 Related work

2.1 General graph clustering

The approach of graph representation of the data leads to the extensive literature over graph clustering - related to graph partitioning and community detection - Schaeffer [2007]. A widely used method to detect communities is the Edge Betweenness Clustering Girvan and Newman [2002] which is a hierarchical divisive algorithm removing iteratively edges with the highest edge betweenness. This method can take into account the edge weights but has a high time complexity: $O(|E|^2|V|)$. DenGraph Falkowski et al. [2007] proposes a graph version of DBSCAN dealing with noise. Work from Ailon et al. [2013] focuses on the problem of recovering clusters with considerably dissimilar sizes. Recent works include also approaches from convex optimization using low-rank decomposition of the adjacency matrix Oymak and Hassibi [2011], Chen et al. [2012a,b, 2014a,b]. These methods bring theoretically guarantees about the exact recovery of the ground-truth clustering for the Stochastic Block Model Holland et al. [1983], Condon and Karp [2001], Rohe et al. [2011] but demand to compute the eigendecomposition of a $N \times N$ matrix (resp. $O(N^3)$ and $O(N^2)$ for time and space complexity). Moreover they are restricted to unweighted graphs - weights in the work from Chen et al. [2014b] are about uncertainty of existence of an edge, not a distance between points -.

2.2 MST-based graph clustering

The Minimum Spanning Tree (MST) is known to help recognizing clusters with arbitrary shapes. Clustering algorithms from this family identify clusters by performing suitable cuts among the MST edges. The first one Standard Euclidean MST (SEMST) is from Zahn [1971] and consists in deleting the heaviest edges from the Euclidean MST of the considered graph but this completely fails when the intra-cluster distance is lower than the inter-clusters one. For decades since MST-based clustering methods Asano et al. [1988], Grygorash et al. [2006] have been developed and can be classified into the group of density-based methods. MSDR Grygorash et al. [2006] relies on the mean and the standard deviation of edge weights

within clusters but will encourage clusters with points far from each other as soon as they are equally "far". Moreover, it does not handle clusters with less than three points. In practice, MST-based clustering algorithms have been successfully applied in bioinformatics Xu et al. [2002] and image color segmentation Grygorash et al. [2006].

2.3 Graph sketching

The work in Ahn et al. [2012a] is at the origin of graph sketches study by giving dynamic semi-streaming methods for recovering different properties of graphs (MST, connectivity, bipartiteness, etc.) and has lead to numerous works since, with among others Ahn et al. [2012b, 2013], Bhattacharya et al. [2015], Huang and Peng [2016], Bandyopadhyay et al. [2016].

3 Our streaming clustering algorithm DBMSTClu

3.1 Context and notations

Consider a dataset with N points. Either the underlying network already exists, or it is assumed that a similarity graph G between points can be built where points of the dataset are the vertices and weighted edges express distances between these objects. For instance, this can be the Euclidean distance. In both cases, the graph on which DBMSTClu is applied should follow this definition:

Definition 3.1 (graph G = (V, E)) *A graph $G = (V, E)$ consists in a set of vertices or nodes V and a set of edges $E \subseteq V \times V$. No attributes are assigned to nodes or edges. The graph is undirected but weighted. The weight w on an edge between node i and j - if this edge exists - corresponds to the normalized predefined distance between i and j, s.t. $0 < w \leq 1$.*

In the remaining part of the paper we denote $[N] = \{1, ..., N\}$. $|V|$ and $|E|$ stand respectively for the cardinality of sets V and E. In short, $|V| = N$ and $|E| = M$. For a dense graph, $M = N(N-1)/2$. $E = \{e_1, ..., e_M\}$ and for all edge e_i is assigned a weight w_i corresponding to a distance between two vertices. $E(G)$ is used to describe the set of edges of a graph G.

Our streaming clustering algorithm is divided into two main steps: 1) From the stream of edge weights, we build a sketch of the graph and then compute an approximate Minimum Spanning Tree (MST) with no more information than the previously obtained sketch. 2) The nodes clustering phase is performed from the approximate MST without requiring any parameter.

3.2 Streaming graph sketching

Processing data in the dynamic streaming model Muthukrishnan [2005] implies the following: 1) The graph should be handled as a stream s of edge weight updates: $s = (a_1, ... a_j, ...)$ where a_j is the j-th update in the stream corresponding to the tuple $a_j = (i, w_{old,i}, \Delta w_i)$ with i denoting the index of the edge to update, $w_{old,i}$ its previous weight and Δw_i the update to perform. Consequently, after reading a_j in the stream, the i-th edge is assigned the new weight $w_i = w_{old,i} + \Delta w_i$. 2) The algorithm should make only one pass over this stream - or at most few passes, but in this paper only one pass is allowed -. 3) Edges can be both inserted or deleted (turnstile model). So weights change regularly, as in social networks where individuals can be friends for some time then not anymore.

As a new update is seen only once in the stream of updates, we use graph sketches from Ahn et al. [2012a] relying on the principle of ℓ_0-sampling Cormode and Firmani [2014] to produce a limited number of linear measurements of the graph. The basic idea is to compress information on edge weights through few linear measurements. These are updated dynamically as new data a_j from stream s is seen only once. This compact data structure enables us to draw almost uniformly at random a nonzero weighted edge (in short, a nonzero edge) at any time thanks to ℓ_0-sampling Cormode and Firmani [2014]:

Definition 3.2 (ℓ_0-sampling) *An (ϵ, δ) ℓ_p-sampler for a nonzero vector $x \in \mathbb{R}^n$ fails with a probability at most δ or returns some $i \in [n]$ with probability*

$$(1 \pm \epsilon) \frac{|x_i|^p}{||x||_p^p} \tag{1}$$

where $||x||_p^p = (\sum_{i \in [n]} |x_i|^p)^{1/p}$ is the p-norm of x. In particular, if $p = 0$, in case of no failure, it returns some i with probability

$$(1 \pm \epsilon) \frac{1}{|\operatorname{supp} x|} \tag{2}$$

where $\operatorname{supp} x = \{i \in [n] \mid x_i \neq 0\}$.

The sketch requires $O(\operatorname{poly} \log(N))$ space. It follows that our algorithm is *semi-streamed* but in practice the space cost is significantly lower than the theoretical $O(N^2)$ bound and the algorithm needs only one pass over the data. This sketch is the first one to support both insertion and deletion of edges. In this context, the number of nodes is known while the edges, whose weights can be increased or decreased (but have always to stay positive) are summarized into those sketches.

3.3 Recovery of an approximate MST from the graph sketch

The work from Ahn et al. [2012a] proposed an algorithm to compute approximately in a single-pass the weight of a MST by appropriate samplings from the sketch. The spatial complexity of the sketch is more precisely $O(N \log^3 N)$, the recovery time for MST is $\tilde{O}(N)$ (the notation $\tilde{O}(N)$ is used to hide $\operatorname{poly} \log(N)$ factors) while the time for each update of the sketch is $\operatorname{poly} \log(N)$. Here the extended method is applied (based on private conversation with Mario Lucic) for obtaining rather an approximate MST - and not simply its weight - by registering edges as they are sampled. Referring to the proof of Lemma 3.4 in Ahn et al. [2012a], the approach is simply justified by applying Kruskal's algorithm where edges with lower weights are first sampled.

Note that the term MST is kept in the whole paper for simplicity, but the sketching technique so our algorithm enables also to recover a Minimum Spanning Forest if the initial graph is disconnected.

3.4 DBMSTClu: MST-based Graph clustering

3.5 Preliminaries

Consider a dataset with N points. After the sketching phase, an approximate MST further named T has been obtained with $N - 1$ edges s.t. $\forall i \in [N-1]$, $0 < w_i \leq 1$. Our density-based clustering algorithm DBMSTClu will exclusively rely on this object by performing some cuts among the edges of the tree s.t. $K - 1$ cuts result in K clusters. At most $N - 1$ cuts are therefore possible. After a cut, obtained clusters can be seen as subtrees of the initial T and the analysis of their qualities is only based on edges contained in those subtrees. For all clusters C_i, $i \in [K]$, we denote the corresponding subtree of T, S_i. In the sequel, for instance the maximal edge of a cluster will refer to the edge with the maximum weight from the subtree corresponding to the current cluster.

Our algorithm is a non-parametric divisive top-down procedure (as opposed to an agglomerative clustering algorithm): it starts from one cluster containing the whole dataset and at each iteration, a cut corresponding to the one maximizing a certain criteria is performed.

The criterion used for identifying the best cut to do (if any should be made) at a given stage is a measure of the *validity* of the obtained clustering partition. This is a function of two positive quantities defined below: *Dispersion* and *Separation* of one cluster. The quality of a given cluster is then measured from Dispersion and Separation while the quality of the clustering partition results from the weighted average of all cluster validity indices. Finally, all those measures are based on the value of edge weights and the two latter ones lie between -1 and 1.
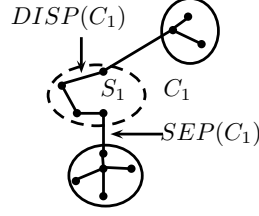
Figure 1: Illustration of the introduced definitions with $N = 12$, $K = 3$ for cluster C_1 represented by the subtree S_1 of T (dashed-circled set in the middle).

Definition 3.3 (Cluster Dispersion) *The Dispersion of a cluster C_i (DISP) represented by the subtree S_i of MST T is defined as the maximum edge weight of S_i. If the cluster is a singleton (i.e. contains only one node), the corresponding Dispersion is set to 0. More formally, with S_j the subtree of T corresponding to C_i:*

$$\forall i \in [K], \ DISP(C_i) = \begin{cases} \max\limits_{j, \ e_j \in S_j} w_j & if \ \ |E(S_j)| \neq 0 \\ 0 & otherwise. \end{cases} \tag{3}$$

Definition 3.4 (Cluster Separation) *The Separation of a cluster C_i (SEP) is defined as the minimum distance between the nodes of C_i and the ones of all other clusters $C_j, i \neq j, 1 \leq i, j, \leq K, K \neq 1$ where K is the total number of clusters. In practice, it corresponds to the minimum weight among all already cut edges from T incident on a node from C_i. If $K = 1$, the Separation is set to 1. More formally, with S_j the subtree of T corresponding to C_i,*

$$\forall i \in [K], \ SEP(C_i) = \begin{cases} \min\limits_{j, \ e_j \in Cuts(C_i)} w_j & if \ \ K \neq 1 \\ 1 & otherwise. \end{cases} \tag{4}$$

Figure 1 sums up all the definitions above.

High values for Separation means that the cluster is well separated from the other clusters while low values suggest that the cluster is well connected to other clusters.

Definition 3.5 (Validity Index of a Cluster) *The Validity Index of a cluster C_i, $1 \leq i \leq K$ is defined as:*

$$V_C(C_i) = \frac{SEP(C_i) - DISP(C_i)}{\max(SEP(C_i), DISP(C_i))} \tag{5}$$

The Validity Index of a Cluster is defined s.t. $-1 \leq V_C(C_i) \leq 1$ where 1 corresponds to an optimal validity index and -1 to the worst. Also notice that a division by zero (i.e. $\max(DISP(C_i), SEP(C_i)) = 0$) can never happen because Separation and Dispersion are both positive, either because they correspond to an existing edge which by definition have a nonzero positive weight or because a singleton cluster leads to a Dispersion equal to 0.

Notice that when Dispersion is higher than Separation, $-1 < V_C(C_i) < 0$. On the contrary, when Separation is higher than Dispersion, $0 < V_C(C_i) < 1$. So our clustering algorithm will naturally encourage clusters with a high Separation over those with a high Dispersion.

Definition 3.6 (Validity Index of a Clustering Partition) *The Density-Based Validity Index of a Clustering partition $\Pi = \{C_i\}, 1 \leq i \leq K$, $DBCVI(\Pi)$ is defined as the weighted average of the Validity*
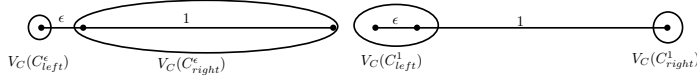
Figure 2: Illustration of the Validity Index of a Cluster with $N = 3$ points. For a given small ϵ, two cuts are considered: edge with weight ϵ (top) and 1 (bottom). (left) $V_C(C_{left}^{\epsilon}) = 1$ and $V_C(C_{right}^{\epsilon}) = \epsilon - 1 < 0$. (right) $V_C(C_{left}^{1}) = 1 - \epsilon > 0$ and $V_C(C_{right}^{1}) = 1$. DBMSTClu will naturally privilege the bottom partition for which validity indices of each cluster are positive.

Indices of all clusters in the partition. We denote N as the number of points in the dataset.

$$DBCVI(\Pi) = \sum_{i=1}^{K} \frac{|C_i|}{N} V_C(C_i) \tag{6}$$

The Validity Index of Clustering lies also between -1 and 1 where 1 corresponds to an optimal density-based clustering partition while -1 is for the worst one.

Notice that our quantities are significantly distinct from the *separation* and *sparseness* defined in Moulavi et al. [2014]. Indeed, firstly, their quantities are not well defined for special cases when clusters have less than four nodes or a partition containing only one cluster. Secondly differentiating internal and external nodes or edges does not enable to properly recover easy clusters like the three convex blobs from Figure 5. Moreover, our DBCVI differs from the Silhouette Coefficient from Rousseeuw [1987]. Although this is based on close concepts like *tightness* and also *separation*, then the global coefficient is based on the average values of Silhouette coefficients of each point, while our computation of DBCVI begins at the cluster level.

Our algorithm DBMSTClu is summarized in Algorithm 1. As previously said, we start from a partition with one cluster containing the whole dataset. The DBCVI of this initial partition is set to the worst possible value: -1. While there exists a cut which makes the DBCVI greater from the one of the current partition, we greedily choose such a cut which maximizes the obtained DBCVI among all the possible cuts. When no direct improvement is possible, the algorithm stops. It is guaranteed that we cut the edge maximizing the DBCVI at each iteration since by construction, the *greedy* algorithm will test each possible cut. Moreover, the non-parametric characteristic helps obtaining stable partitions.

Since Dispersion for a singleton cluster C_i is 0, we have $V_C(C_i) = 1$. One could argue that in this case, an optimal partition would be given by all points in singleton clusters. In practice, the algorithm stops after a reasonable number of cuts, getting trapped in a local maximum corresponding to a meaningful cluster partition.

In Algorithm 1, $performCut$ is the routine performing the cut corresponding to the edge in parameter with respect to already formed clusters. $getDBCVI$ is the function computing the Validation Index of the Clustering partition in parameter.

3.6 Quality of clusters

An analysis of the algorithm and the quality of the obtained clusters is given in this section. Our main results are the following: 1) DBMSTClu differs significantly from the naive approach of SEMST by preferring cuts which are not necessarily corresponding to the heaviest edge (Propositions 1 and 2). 2) While the current partition contains at least one cluster with a negative validity index, DBMSTClu will find a cut improving the global index (Proposition 3).

Propositions 1 and 2 rely on the three basic lemmas:

Algorithm 1 Clustering algorithm

 1: **Input:** T, the MST
 2: $splitDBCVI \leftarrow -1.0$
 3: $cut\_candidate\_list \leftarrow [\ edges(T)\ ]$
 4: $clusters = [\ ]$
 5: **while** $splitDBCVI < 1.0$ **do**
 6:   $optimal\_cut \leftarrow None$
 7:   **for** each cut in $cut\_candidate\_list$ **do**
 8:     $newClusters \leftarrow performCut(clusters, cut)$
 9:     $newDBCVI \leftarrow getDBCVI(newClusters, T)$
10:     **if** $newDBCVI \geq splitDBCVI$ **then**
11:       $optimal\_cut \leftarrow cut$
12:     **end if**
13:   **end for**
14:   **if** $optimal\_cut \neq None$ **then**
15:     $clusters \leftarrow performCut(clusters, optimal\_cut)$
16:     $splitDBCVI \leftarrow getDBCVI(clusters, T)$
17:     $remove(cut\_candidate\_list, optimal\_cut)$
18:   **else**
19:     **Output:** clusters
20:   **end if**
21: **end while**
22: **Output:** clusters

Lemma 1 (Case of the edge with the highest weight) *Let T be the MST corresponding to the similarity graph. If the first cut performed by DBMSTClu among the edges of T corresponds to the edge with the heaviest weight then the produced DBCVI will be nonnegative.*

Proof 1 *First notice that when performing the first cut in the partition, the separation for both obtained clusters is the same and equal to be the weight of the considered edge we cut. In our case, the considered edge for cut is the heaviest one. Consequently, for both resulting clusters, respective dispersions will be lower than the separation leading to two nonnegative cluster validation indices. Obviously, the final DBCVI as a convex sum of two nonnegative quantities will be nonnegative.*

Lemma 2 (Case of the edge with the lowest weight) *Let T be the MST corresponding to the similarity graph. If the first cut performed by DBMSTClu among the edges of T corresponds to the edge with the lowest weight then the produced DBCVI will be less than zero.*

Proof 2 *Same reasoning in the opposite case s.t. $SEP(C_i) - DISP(C_i) \leq 0$ for $i \in \{1, 2\}$.*

Lemma 3 *Let T be the MST of the data. Let us consider a cluster C and denote its separation by s. Then, among the edges of C, DBMSTClu will never perform a cut on an edge e of weight w lower than s if both resulting clusters have at least one edge weight greater than w.*

Proof 3 *As a first cut, if edge e is cut, it is clear that separation between both possible clusters is w. Let us assume that in both subtrees, we have an edge with a weight higher than w. This means that for both clusters, corresponding dispersions are higher than separation. So the cluster validation indices are negative. Lemma 1 gives us that cutting the heaviest edge gives two nonnegative cluster validation indices hence leading to a higher final DBCVI than cutting edge e. As a cut is done on the edge improving at most the DBCVI from the current state, e would not get cut at this step of the algorithm. At any other stage of the algorithm, the same reasoning holds as soon as $w \leq s$ since with respect to the possible cut of edge e, the connected component behaves like an isolated graph (separation for the two resulting clusters is w).*

Proposition 1 *Let N be the number of nodes in the initial MST T. Let us consider the following specific case: all edges have a weight equal to w except two edges e_1 and e_2 respectively with weight w_1 and w_2 s.t. $w_1 > w_2 > w > 0$. In this case, the algorithm will not cut any edge with weight w and e_2 will be chosen for getting cut instead of e_1 iff:*

$$w_2 > \frac{2n_2 w_1 - n_1 + \sqrt{n_1^2 + 4w_1(n_2^2 w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)}. \tag{7}$$

where n_1 (resp. n_2) denotes the number of nodes in the first cluster resulting from the cut of e_1 (resp. e_2). Otherwise, e_1 will get cut.

Proof 4 *Let $DBCVI_1$ (resp. $DBCVI_2$) be the DBCVI after cut of e_1 (resp. e_2).*

As w corresponds to the minimum weight, Lemma 2 and Lemma 3 state that the algorithm will not cut at edge e since the resulting DBCVI would be negative while DBCVI is guaranteed to be nonnegative if we cut e_1. So, the choice will be between e_1 and e_2. Choosing edge e_2 corresponds to $DBCVI_2 > DBCVI_1$. The remaining part of the proof is fastidious calculation showing that:

$$DBCVI_2 > DBCVI_1 \iff$$

$$w_2 > \frac{2n_2 w_1 - n_1 + \sqrt{n_1^2 + 4w_1(n_2^2 w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)}.$$

This proposition emphasizes that the algorithm is cleverer than simply cutting the heaviest edge first. Indeed, although $w_2 < w_1$, there exists a condition when e_2 could get cut instead of e_1. Moreover, an edge with weight w s.t. $w < w_2 < w_1$ can never be cut at the first iteration. This proposition can seem very particular but in practice it really happens as we are using an approximate MST and rounded weights.

Proposition 2 *Let N be the number of nodes in the initial MST T. Let us consider the following specific case: all edges have a weight equal to w except two edges e_1 and e_2 respectively with weight w_1 and w_2 s.t. $w_1 > w_2 > w > 0$. In the particular case where edge e_1 with maximal weight w_1 is between two subtrees with the same number of points, i.e. $n_1 = \frac{N}{2}$, e_1 will be always preferred over e_2 as the first optimal cut.*

Proof 5 *We make a reductio ad absurdum by showing that cutting edge e_2 i.e. $DBCVI_2 > DBCVI_1$ leads to the contradiction $w_1/w < 1$. This is again long calculus.*

Remark 1 *Let us consider the initial MST T before the first cut has been performed, with e_1 the edge with the maximal weight w_1 s.t. $n_1 = N/2$. In a more general case than for Proposition 2, we have respectively e_2 with the maximal weight w_2 of the left subtree generated by a virtual cut of e_1, e_3 with weight e_3, maximal weight of the right tree. So it is not true to claim that e_1 will be the best possible first cut. Here is a counter-example (see Figure 3): consider the following flat MST with $N = 8$, $w_1 = 1$, $w_2 = w_3 = 1 - \epsilon$, and other weights are ϵ.*
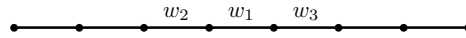

Figure 3: Counter-example for Remark 1 : A flat MST with $N = 8$, $w_1 = 1$, $w_2 = w_3 = 1 - \epsilon$, and other weights are ϵ.

This is clear that we prefer not to cut in the middle, i.e. edge e_1 because for $\epsilon = 0.1$, $DBCVI_2 \approx 0.27 > DBCVI_1 = \epsilon = 0.1$.
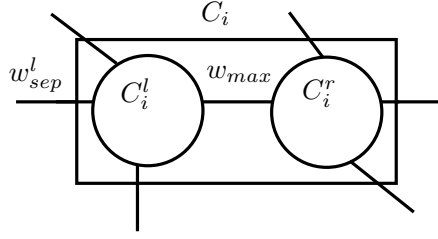
Figure 4: Generic example for proof of Proposition 3

This last proposition holds for every iteration in the algorithm.

Proposition 3 (Fate of clusters with a negative validity index) *Let $K = t + 1$ be the number of clusters in the clustering partition at iteration t. If some $i \in [K]$ exists s.t. $V_C(C_i) < 0$, then DBMSTClu will cut an edge at this stage.*

Proof 6 *Let be $i \in [K]$ s.t. $V_C(C_i) < 0$ i.e. $SEP(C_i) < DISP(C_i)$. We denote w_{sep}^l and w_{max} respectively the minimal weight outing the cluster C_i and the maximal weight in the subtree S_i of C_i. By definition, $SEP(C_i) = w_{sep}^l$ and $DISP(C_i) = w_{max}$. By cutting the cluster C_i on the edge giving the dispersion, i.e. the edge with weight w_{max}, we define C_i^l and C_i^r resp. the left and right resulting clusters. Arbitrarily, we decide that the left part of C_i gives the separation value (w_{sep}^l), but the same reasoning applies on the right side. Figure 4 gives a visual illustration of the case. After a cut on the edge with weight w_{max}, separation of the left resulting cluster C_i^l remains the same since $w_{sep}^l < w_{max}$, i.e. $SEP(C_i^l) = SEP(C_i)$. But, clearly, $DISP(C_i^l) < SEP(C_i)$ (there is equality if there exists an other edge in C_i^l with a weight equal to w_{max}). So, $V_C(C_i^l) > V_C(C_i)$.*

For the right side, $SEP(C_i^r) > SEP(C_i)$ (equality if there is an outing edge from C_i with weight equal to w_{sep}^l). Moreover, $DISP(C_i^r) < DISP(C_i)$ (equality if there is another edge with weight w_{max}) and $V_C(C_i^r) > V_C(C_i)$. Consequently, in both cases, the validation indices of both clusters are higher than $V_C(C_i)$. So, with this cut, DBCVI is improved. Finally, if we do not fall into the special cases i.e. we have strict inequalities, we improve the global DBCVI by cutting at least the heaviest edge in a cluster with a negative index. In this case, no cluster with a negative index will stay uncut.

As a consequence, at the end of the clustering algorithm, every cluster will have a positive cluster validation index. So, at each step of the algorithm, we have the following bound for the final DBCVI:

$$DBCVI \geq \sum_{i=1}^{K} \frac{|C_i|}{N} \max(V_C(C_i), 0) \tag{8}$$

4 Experiments

Experiments were conducted using Python and scikit-learn library Pedregosa et al. [2011]. All figures should be read in color. Experiments were performed on three classic datasets from the Euclidean space: three blobs, noisy circles and noisy moons. Each dataset contains 1000 data points in 20 dimensions: the first two dimensions are randomly drawn from predefined 2D-clusters, as shown in the following figures, while the other 18 dimensions are random Gaussian noise. We compare the results of DBMSTClu with
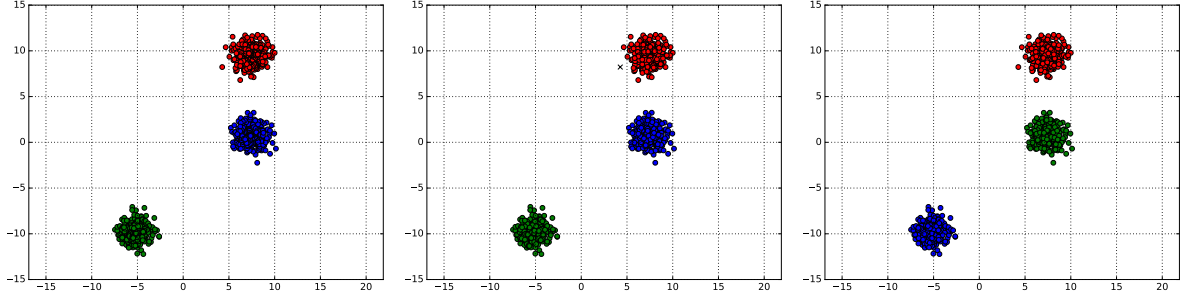
Figure 5: Three blobs: SEMST, DBSCAN ($\epsilon = 1.0$, $minPts = 5$), DBMSTClu with an approximate MST
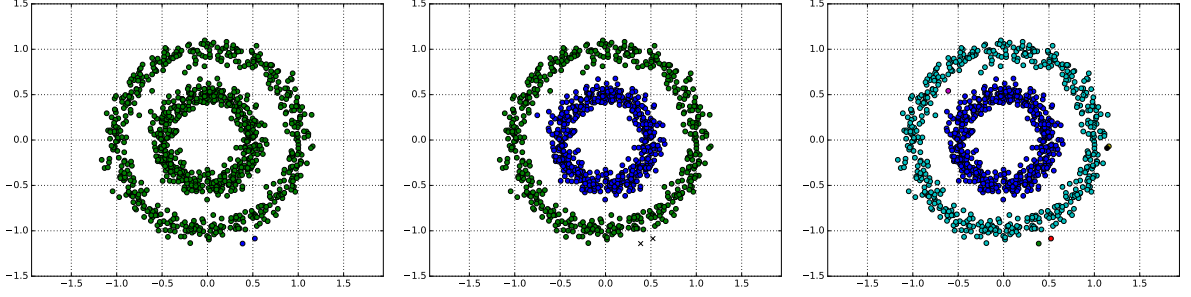

Figure 6: Noisy circles: SEMST, DBSCAN ($\epsilon = 0.15$, $minPts = 5$), DBMSTClu with an approximate MST

two other algorithms: DBSCAN Ester et al. [1996] and a naive approach of a MST-based algorithm. The latter works as the following and is due to Zahn [1971]: as input parameters, the standard Euclidean MST and the target number of clusters are given; then the $K - 1$ heaviest edges of the MST are cut. We call it Standard Euclidean Minimum Spanning Tree (SEMST). For DBMSTClu, we build the k-NN graph from computing the Euclidean distance between data points with $k = 550$ for noisy circles and moons and $k = 350$ for the three blobs. Then this k-NN graph is passed into the sketch phase to produce an approximate version of the exact MST.

Figures 5, 6 and 7 show the results while providing to the clustering algorithm an approximate MST obtained from the sketch phase. They were produced with a noise level such that SEMST fails and DBSCAN does not perform well without parameters optimization. For the three blobs, each method perform well. For DBSCAN all the cross points correspond to noise. For the concentric circles, SEMST
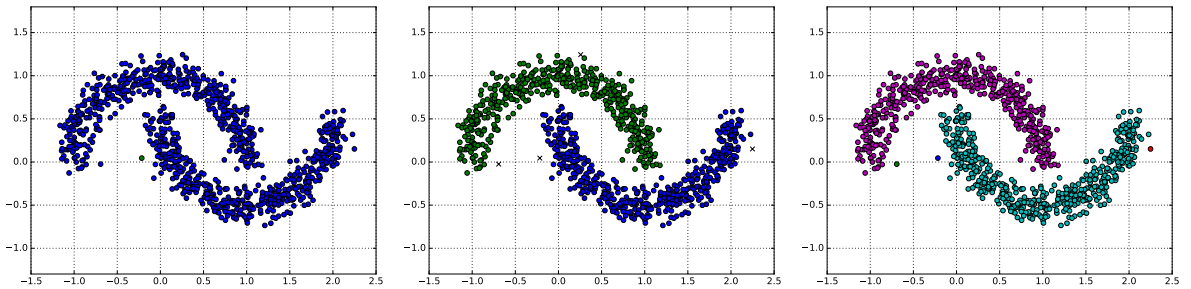

Figure 7: Noisy moons: SEMST, DBSCAN ($\epsilon = 0.15$, $minPts = 5$), DBMSTClu with an approximate MST

does not cut on the consistent edges, hence leads to an isolated cluster with two points. DBSCAN classifies the same points as noise while recovering the two circles well. Finally, DBMSTClu finds the two main clusters and also creates five singleton clusters understood as noise. For noisy moons, whereas DBSCAN considers three outliers, DBMSTClu puts them into singletons. As theoretically proved above, experiments valid the fact that our algorithm is much more subtle than simply cutting the heaviest edges as the failure of SEMST shows. Moreover our algorithm exhibits an ability to detect outliers. Another decisive advantage of our algorithm is the absence of any required parameters.

5   Conclusion

In this paper we introduced a novel semi-streaming Density-Based Clustering algorithm which we call DBMSTClu. Unlike most existing clustering methods, DBMSTClu is non-parametric. Its robustness has been assessed by using as input a sketch of the Minimum Spanning Tree (MST) of the induced similarity graph deduced from the dataset rather than the MST itself. This sketch is computed dynamically on the fly as new edge weight updates are received and requires only one pass over the data. This allows us to perform clustering in a space-efficient way. Our approach shows promising results, as evidenced by the experimental part. Further work would be to use this algorithm in privacy issues, as the information we lose when sketching might be sufficient to ensure data privacy. Moreover, as it is already the case for the graph sketching, it would be interesting to adapt both the MST recovery and DBMSTClu to the fully online setting, i.e. to be able to modify appropriately current MST and clustering partition as a new edge weight update from the stream is seen.

Acknowledgements

References

K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467, Philadelphia, PA, USA, 2012a. Society for Industrial and Applied Mathematics.

K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 5–14, New York, NY, USA, 2012b. ACM.

K. J. Ahn, S. Guha, and A. McGregor. *Spectral Sparsification in Dynamic Graph Streams*, pages 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

N. Ailon, Y. Chen, and H. Xu. Breaking the small cluster barrier of graph clustering. *CoRR*, abs/1302.4549, 2013.

T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88, pages 252–257, New York, NY, USA, 1988. ACM.

B. Bandyopadhyay, D. Fuhry, A. Chakrabarti, and S. Parthasarathy. Topological graph sketching for incremental and scalable analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1231–1240, New York, NY, USA, 2016. ACM.

S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 173–182, New York, NY, USA, 2015. ACM.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012a.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012b.

Y. Chen, A. Jalali, S. Sanghavi, and H. Xu. Clustering partially observed graphs via convex optimization. *J. Mach. Learn. Res.*, 15(1):2213–2238, Jan. 2014a.

Y. Chen, S. H. Lim, and H. Xu. Weighted graph clustering with non-uniform uncertainties. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1566–II–1574. JMLR.org, 2014b.

A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, Mar. 2001.

G. Cormode and D. Firmani. A unifying framework for l_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014. Special issue on Data Summarization on Big Data.

M. Ester, H. peter Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

T. Falkowski, A. Barth, and M. Spiliopoulou. Dengraph: A density-based community detection algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 112–115, Washington, DC, USA, 2007. IEEE Computer Society.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002.

O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 73–81, 2006.

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5 (2):109–137, June 1983.

Z. Huang and P. Peng. Dynamic graph stream algorithms in \$o(n)\$ space. *CoRR*, abs/1605.00089, 2016.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L_1–Norm and Related Methods*, pages 405–416, 1987.

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

D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 839–847, 2014.

S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1 (2):117–236, Aug. 2005.

M. C. Nascimento and A. C. de Carvalho. Spectral methods for graph clustering - a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

S. Oymak and B. Hassibi. Finding dense clusters via "low rank + sparse" decomposition. *CoRR*, abs/1104.5186, 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.*, 39(4):1878–1915, 08 2011.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

S. E. Schaeffer. Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64, Aug. 2007.

Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. 18(4):536–545, 2002.

C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, Jan. 1971.

6 APPENDIX

In this section we prove propositions 1 and 2 presented in the main body of the paper for which only a sketch of the proof has been given.

Proposition 1 Let N be the number of nodes in the initial MST T. Let us consider the following specific case: all edges have a weight equal to w except two edges e_1 and e_2 respectively with weight w_1 and w_2 s.t. $w_1 > w_2 > w > 0$. In this case, the algorithm will not cut any edge with weight w and e_2 will be chosen for getting cut instead of e_1 iff:

$$w_2 > \frac{2n_2 w_1 - n_1 + \sqrt{n_1^2 + 4w_1(n_2^2 w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)}. \qquad (9)$$

where n_1 (resp. n_2) denotes the number of nodes in the first cluster resulting from the cut of e_1 (resp. e_2). Otherwise, e_1 will get cut.

Proof 7 *Let $DBCVI_1$ (resp. $DBCVI_2$) be the DBCVI after cut of e_1 (resp. e_2).*

As w corresponds to the minimum weight, Lemma 2 and Lemma 3 say that the algorithm will not cut at edge e since the resulting DBCVI would be negative while DBCVI is guaranteed to be positive if we cut e_1. So, the choice will be between e_1 and e_2.

The DBCVI while cutting e_1, respectively e_2 are:

$$DBCVI_1 = \frac{n_1}{N}(1 - \frac{w}{w_1}) + (1 - \frac{n_1}{N})(1 - \frac{w_2}{w_1}) \qquad (10)$$

$$DBCVI_2 = \frac{n_2}{N}(\frac{w_2}{w_1} - 1) + (1 - \frac{n_2}{N})(1 - \frac{w}{w_2}) \qquad (11)$$

Cutting on edge e_2 implies $DBCVI_2 > DBCVI_1 > 0$. To simplify the reasoning, without loss of generality, we scale the weights by w as follows: $w \leftarrow 1$, $w_1 \leftarrow w_1/w$, $w_2 \leftarrow w_2/w$. So we have: $w_1 > w_2 > 1$. Then,

$$DBCVI_2 > DBCVI_1 \iff$$

$$\frac{n_2}{N}(\frac{w_2}{w_1} - 1) + (1 - \frac{n_2}{N})(1 - \frac{1}{w_2})$$

$$- \frac{n_1}{N}(1 - \frac{1}{w_1}) + (1 - \frac{n_1}{N})(1 - \frac{w_2}{w_1}) > 0 \quad (12)$$

After some calculations and simplifications,

$$DBCVI_2 > DBCVI_1 \iff$$

$$w_2^2 \underbrace{(N + n_2 - n_1)}_{a} + w_2 \underbrace{(n_1 - 2n_2 w_1)}_{b} + \underbrace{(n_2 - N)w_1}_{c<0} > 0. \quad (13)$$

This is a second degree polynomial equation in w_2 with coefficients a, b and c. Clearly, $\Delta = b^2 - 4ac$ is positive and the product of the solutions c/a is negative. But $w_2 > 0$, so by computing the positive solution $\frac{-b+\sqrt{b^2-4ac}}{2a}$ and after some simplifications we have the final result:

$$DBCVI_2 > DBCVI_1 \iff$$

$$w_2 > \frac{2n_2 w_1 - n_1 + \sqrt{n_1^2 + 4w_1(n_2^2 w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)}$$

Proposition 2 Let N be the number of nodes in the initial MST T. Let us consider the following specific case: all edges have a weight equal to w except two edges e_1 and e_2 respectively with weight w_1 and w_2 s.t. $w_1 > w_2 > w > 0$. In the particular case where edge e_1 with maximal weight w_1 is between two subtrees with the same number of points, i.e. $n_1 = \frac{N}{2}$, e_1 will be always preferred over e_2 as the first optimal cut.

Proof 8 *We make a reductio ad absurdum by showing that cutting edge e_2 will lead to a contradiction. After the scaling process, $DBCVI_1$ and $DBCVI_2$ are rewritten:*

$$DBCVI_1 = \frac{1}{2}(1 - \frac{1}{w_1}) + \frac{1}{2}(1 - \frac{w_2}{w_1})$$
$$= 1 - \frac{1}{2w_1} - \frac{w_2}{2w_1} \tag{14}$$
$$DBCVI_2 = \frac{n_2}{N}(\frac{w_2}{w_1} - 1) + (1 - \frac{n_2}{N})(1 - \frac{1}{w_2})$$
$$= 1 - \frac{1}{w_2} + \frac{n_2}{N}\underbrace{(\frac{w_2}{w_1} + \frac{1}{w_2} - 2)}_{=A} \tag{15}$$

We have $A < 0$, since $\frac{w_2}{w_1} < 1$ because $w_2 < w_1$ and $\frac{1}{w_2} < 1$ ($w_2 > w_1$). Let now consider without loss of generality that edge e_2 is on the "right side" (right cluster/subtree) of e_1 (similar proof if e2 is on the left side of e_1). Consequently, it is clear that for maximizing $DBCVI_2$ as a function of n_2, we need $n_2 = n_1 + 1$. Then,

$$DBCVI_2 > DBCVI_1$$
$$\iff -\frac{1}{w_2} + (\frac{1}{2} + \frac{1}{N})(\frac{w_2}{w_1} - 2 + \frac{1}{w_2}) > -\frac{1}{w_1} - \frac{w_2}{w_1}$$
$$\iff (\frac{1}{2w_1} + \frac{1}{Nw_1} + \frac{1}{2w_1})w_2 - 1 - \frac{2}{N} + \frac{1}{2w_1} +$$
$$(-1 + \frac{1}{2} + \frac{1}{N})\frac{1}{w_2} > 0$$
$$\iff \underbrace{(1 + \frac{1}{N})}_{a>0} w_2^2 + w_2 \underbrace{(\frac{1}{2} - w_1(1 + \frac{2}{N}))}_{b<0} + w_1 \underbrace{(\frac{1}{N} - \frac{1}{2})}_{c<0} > 0$$

As $c/a < 0$ and $w_2 > 0$, we have:

$$w_2 > \frac{N}{2(N+1)} \left[w_1(1 + \frac{2}{N}) - \frac{1}{2} + \sqrt{\Delta} \right]$$

with $\Delta = (w_1(1 + \frac{2}{N}) - \frac{1}{2})^2 + 4(1 + \frac{1}{N})(\frac{1}{2} - \frac{1}{N})w_1$ We will show now that this inequality is incompatible with $w_1 > w_2$.

$$w_1 > w_2 \iff w_1 > \frac{N}{2(N+1)} \left[w_1(1 + \frac{2}{N}) - \frac{1}{2} + \sqrt{\Delta} \right]$$
$$\iff w_1 + \frac{1}{2} > \sqrt{\Delta}$$
$$\iff \frac{4}{N} w_1^2 (1 + \frac{1}{N}) + \frac{4}{N}w_1(-1 - \frac{1}{N}) < 0$$
$$\iff w_1 < 1 \; : ILLICIT$$

Indeed, after the scaling process, $w_1 < 1$ is not possible because it means $w_1/w < 1$. Finally, it is not allowed to cut edge e_2, the only remaining possible edge to cut is e_1.