Stream Clustering with Dynamic Estimation of Emerging Local Densities

Ziyin Wang
Department of Computer and Information Science
Indiana University-Purdue University Indianapolis
Indianapolis, IN 46202, USA
Email: wang2457@purdue.edu

Gavriil Tsechpenakis
Department of Computer and Information Science
Indiana University-Purdue University Indianapolis
Indianapolis, IN 46202, USA
Email: gtsechpe@indiana.edu

***Abstract*—We present a method for clustering data streams incrementally, designed to discover all valid density peaks in a single pass, in a non-parametric fashion. It detects emerging clusters along the stream by dynamically locating kernels in the most promising areas and performing a Stochastic Mean Shift procedure to find clustering centers. We present a density estimation approach for dynamic initialization, considering every sub-stream that follows 'emerging data' as a sample set and applying Hypothesis Testing (p-value approach) to estimate its local density. The sub-stream size and the p-value are determined in a way that provides provable accuracy guarantee. We compare our method with the state-of-the-art, on realistic and complex datasets. We show that it outperforms not only stream algorithms but also their more complex, non-stream foundational paradigms.**

I. Introduction

In stream clustering, data are processed sequentially (e.g., during acquisition) with two main constraints, namely computational efficiency and the use of limited memory, that give rise to single-pass algorithms. The lack of prior knowledge, such as the number of clusters and data size, renders the problem even more challenging. The use of stream clustering is well motivated by a wide range of application domains such as onboard processing in unmanned aerial vehicles, online user behavior, commercial transaction chains, etc.

Clustering data streams by approximating popular non-stream algorithms has been a popular trend in the state-of-the-art, with tight and proven convergence boundaries [26][1][12][15], providing much faster processing at the expense of clustering quality. Considering that the foundational algorithms themselves already sacrifice accuracy for reasonable speed, there is still much left to be desired regarding the trade-off between efficiency and clustering quality. Our objective in this work was to develop an algorithm that achieves the highest possible accuracy in the center-based clustering domain, while maintaining top-tier speed for processing data streams.

Here we consider the Density Peak clustering paradigm: given a neighborhood size θ and a minimum density ratio $0 < p_f < 0.5$, find all local density peaks $C = \{c_1, c_2, ..., c_K\}$ such that $p_k^\theta = \frac{N_k}{N} \geq p_f, \forall k = 1, \ldots, K$, where p_k^θ is the density ratio of θ-neighborhood around c_k, N_k is the number of data points covered by this θ-neighborhood, and N is the data size. We consider this formulation for data streams because (a) the number of clusters is *a priori* unknown and new clusters are likely to emerge continually, and (b) a density peak inherently represents its neighborhood. In our method, neighborhood size θ and density threshold p_f are the only two user-specified parameters, and no other prior information is required.

A. Related Work

The data stream clustering problem gave rise to a number of seminal works [30] [21] [2] [8]. The recent state-of-the-art focuses more on approximating well-known algorithms [26][25][1][12] [15][5]. [25] describes a heuristic method for online K-means but without approximation boundary guarantees. [1] presents a sampling method along with map-reduce to construct a weighted subset (coreset), such that running K-means++ on coreset can be a proven approximation of the original algorithm. The coreset idea was also adopted in [12] and [15]. The recently work in [19] presents an online approximation of K-means that instantiates a new cluster when an incoming data point is far away from existing clusters. In [5] Markov Chain Monte Carlo sampling is used as a seeding approximation of K-means++ algorithm. [16] describes a hierarchical clustering method along with tree rotation to approximate K-means with tractable solution when the number of clusters is large.

A key attribute of data streams is that they can potentially be very large, which calls for designing scalable methods, such as [11]. [10] describes a parallel method to scale the K-means++ initialization process. In [7], a Map-Reduce model is used to minimize dependency between iterations in K-means for efficient scaling. [6] shows an salable solution in generating coreset for K-means and K-median problems.

Most algorithms that approximate foundational non-stream methods usually provide limited clustering quality, as explained above. Some other accurate solutions that have been approximated in streaming problems have higher complexity, and their streaming counterparts are more focused on maintaining clustering quality than achieving the best possible speed. For instance, [28] that uses a Dirichlet Process mixture model is a popular Bayesian nonparametric model for small, low-dimensional data, with $O(iNd^3)$ complexity, where i is the number of Gibbs Sampling iterations, N is the data size and d is the data dimension. [17][4] are faster approximations that reduce the cubic complexity and overcome the bottleneck of Gibbs Sampling. [13] maintains three $N \times N$ matrices to discover clusters in small datasets, with time and space complexities $O(iN^2)$ and $O(N^2)$ respectively. [22] maps the original data set into a 2-D density map with complexity $O(N^2)$ in time and space, such that few density peaks can be manually selected. Nowadays, even moderate-sized datasets are far beyond the capacity of most of these methods.

B. *Our contribution*

We present a novel approach for clustering data streams that is able to find all natural clusters accurately in a single pass, while detecting emerging clusters over time. To support this functionality, we developed a dynamic initialization framework based on Hypothesis Testing (p-value approach). We postulate the problem in way that it can be elegantly solved by this Hypothesis Testing and the accuracy can be proved. We use the Stochastic Mean Shift paradigm to seek nearest density peaks in a single pass, which is drastically benefited by our dynamic initialization approach. In our comparisons with existing popular methods we show that our approach greatly improves the trade-off between accuracy in speed: it outperforms the stream clustering competition in terms of efficiency, and achieves similar level accuracy compared to more complex, non-stream algorithms. We also report results showing that our approach is more accurate compared to some popular iterative methods (K-means, GMM, and Meanshift) that many state-of-the-art stream algorithms approximate.

II. Method

Following the Density Peaks [22] and Mean-shift [9] paradigms, we assume that each clustering center is located at the density peak of a local neighborhood. In every step, Mean-shift computes the mean of data inside a kernel, and moves the kernel to the mean location. If we take a random sample from data inside the kernel, its probabilistic expectation happen to be the mean location, namely $\sum_i p(i)x_i = \frac{1}{N}\sum_i x_i$, where $p(i)$ is the probability that x_i is chosen. This property allows us to seek local density peaks within a single pass.

Consider a toy dataset, in Fig. 1, generated by a Gaussian distribution. Suppose we initially place a kernel with bandwidth θ at Point A. We randomly sample a data point $\boldsymbol{x_i}$ from this dataset (without replacement), and compute its distance to the center. If it is smaller than θ we call it a match, and we update the center as,

$$\boldsymbol{c_k}^{(new)} = \frac{n_k \boldsymbol{c}_k + \boldsymbol{x}_i}{n_k + 1} \tag{1}$$

where $\boldsymbol{c_k}$ is the center of the kernel and n_k is the number of samples it has been matched with so far. Note that inside the kernel, the matched sample is more likely to come from a dense dataset region (yellow-highlighted in Fig. 1). Therefore, each update will problematically move the center towards a denser location. If we apply this "sampling, matching, and updating" process throughout the entire dataset, the center will keep moving towards the nearest density peak, as shown with the red cross path in Fig.1. The resulting path is quite similar to the Mean-shift procedure, however the entire process here is stochastic and merely requires a single pass. Note that since a kernel always moves towards a density peak, n_k can be reset into a small value if the kernel gets trapped into a sparse region (where θ is too small to capture sufficient population). Such case is unlikely to happen with appropriate similarity values θ or when initialization is close to the density peak (point B).

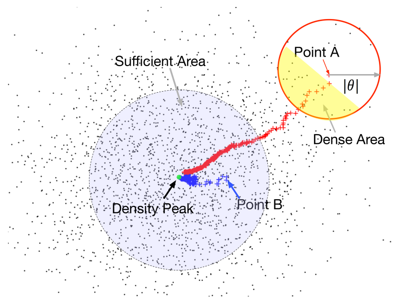

Figure 1. Updating a center: two candidate cluster initializations, at Point A and Point B. The paths shown in red and blue colored cross points indicate convergence towards the density peak of the natural cluster during the 'sampling, matching, and updating' process. The denser region (in yellow) inside the θ-defined kernel denotes where the next matched sample is more likely to be. The blue shadow shows a "sufficient area" where any kernel inside this area is sufficiently dense. The entire process requires a single pass through the dataset, with no iterative operation involved.

A. *Dynamic Initialization: Density Estimation*

To initialize the kernels with respect to their number (*how many?*) and locations (*where?*), our objective is to find at least one kernel that can be shifted towards each density peak. We achieve this by estimating the local density around each input data point that does not fall into any existing kernel, and initializing a new kernel at that location.

Recalling the Density Peak problem statement given in section I, we define a kernel as ***sufficient*** if its density ratio is greater than p_f, and ***insufficient*** otherwise. Around any valid density peak, there exists a dense region such that any kernel inside this region is sufficient (as illustrated by the blue circle in Fig.1). Therefore, as long as we correctly estimate the density around at least one emerging data point within a 'sufficiently' populated region, we are guaranteed to find the local density peak following the stochastic Mean Shift procedure above. Thus, in our dynamic kernel generation principle:

(i) every data point that cannot be matched to any existing kernel is temporarily set as a location of a new kernel;
(ii) for every new kernel, we perform an efficient density test (we estimate its density using the p-value approach), and discard it as noise if it cannot pass the test;
(iii) we use the kernels that pass the density test to seek local density peaks.

We consider each data point of the stream as a random sample from the entire dataset. For every input point that does not belong to an existing kernel, the subsequent r data construct a sample set of size r that can be used for density estimation following the p-value approach. Specifically, given a statistical confidence α (here we consider $\alpha = 1\%$), the following two conditions must be satisfied:

Condition A: If a kernel is known to be sufficient, we have $1-\alpha$ confidence that it will not fail density test.

Condition B: If a kernel passes the density test, we have $1-\alpha$ confidence that it is indeed sufficient.

Proposition 1 (Satisfying Condition A): For any r random samples from the dataset, with

$$r = \frac{ln\alpha}{ln(1-p_f)}, \tag{2}$$

the probability that there are no data points covered by a sufficient kernel k is less than or equal to α.

Proof: Consider the density $p_k = \frac{N_k}{N}$ of a sufficient kernel c_k, $p_k \geq p_f$, with N_k being the set of points covered by c_k and N being the size of the dataset. The probability that a single sample is not covered by kernel c_k is $1-p_k$. Thus, the probability that none of the r samples are covered by the kernel will be,

$$\begin{aligned}(1-p_k)^r = (1-p_k)^{\frac{\ln\alpha}{\ln(1-p_f)}} &= (1-p_k)^{\log_{1-p_f}\alpha} \\ &\leq (1-p_f)^{\log_{1-p_f}\alpha} = \alpha\end{aligned} \tag{3}$$

□

Proposition 2 (Satisfying Condition B): Consider any r samples from the entire dataset and a sufficient kernel c_k. Let the stochastic variable X denote the number of data points among the r samples that fall into kernel c_k. For

$$X_f = rp_f + \Phi^{-1}(\alpha)\sqrt{rp_f\Big(1-p_f\Big)}, \tag{4}$$

we have

$$P\{X > X_f\} \geq 1-\alpha, \tag{5}$$

where $\Phi^{-1}(.)$ is the inverse Cumulative Density Function of Standard Normal distribution. In other words, we have $1-\alpha$ confidence that if the kernel passes the density test, it is indeed sufficient.

Proof: X follows Binomial Distribution $X \sim B(r, p_k)$. According to the Central Limit Theorem, when r is large and $rp_k(1-p_k) \geq 10$, we can approximate binomial distribution $B(r, p_k)$ with normal distribution $\mathcal{N}(rp_k, \sqrt{rp_k(1-p_k)})$. If we consider the statistical variable, $Y = \frac{X - rp_k}{\sqrt{rp_k(1-p_k)}} \sim \mathcal{N}(0,1)$,

$$\begin{aligned}P\{X > X_f\} &\geq P\left\{\frac{X-rp_k}{\sqrt{rp_k(1-p_k)}} > \frac{X_f - rp_k}{\sqrt{rp_k(1-p_k)}}\right\} \\ &\geq P\left\{Y > \frac{X_f - rp_f}{\sqrt{rp_f(1-p_f)}}\right\} \\ &= P\{Y > \Phi^{-1}(\alpha)\} = 1-\alpha,\end{aligned} \tag{6}$$

given $p_k \geq p_f$, $p_f \leq 0.5$ (more than one density peak). □

In Fig.1, the dense region covers a large number of data points; this implies that we are able to detect the density peak from at least one point from this region that passes the density test. Suppose we test L data points from such a dense region.

Algorithm 1: Stochastic_Mean_Shift(D, θ, p_f)

```
1  $Dictionary, Memory \leftarrow \emptyset$, r←eq.(2), $\phi \leftarrow eq.(7)$
2  for each $x \in D$ do
3      $k \leftarrow$ nearest kernel to $x$ in $Dictionary$
4      if similarity (x, k) $\geq \theta$ then
5          update $Dictionary$ kernel $k$ by eq.(1)
6      else
7          $k \leftarrow$ nearest kernel to $x$ in $Memory$
8          if similarity (x, k) $\geq \theta$ then
9              update $Memory$ kernel $k$ by eq.(1)
10             $k.activity = k.activity + r$
11             if $k.activity \geq \phi$ then
12                 trasfer $k$ to $Dictionary$
13         else
14             initial a kernel $k_{new}$ at $x$
15             $k_{new}.activity = r$
16             $Memory.add(k_{new})$
17     Decrease activity of each kernel in $Memory$ by 1
18     Remove kernels with negative activity
19 return $Dictionary$
```

According to Proposition 1, the probability of a single failure is $\alpha = 1\%$. Therefore, the probability of failure for all L tests will be α^L, i.e. it will be almost impossible to fail detecting the density peak. Additionally, according to Proposition 2, if any kernel from such dense area passes the test, we have $1-\alpha$ confidence that it is indeed sufficient.

B. *Single-Pass Algorithm*

Algorithm 1 describes our single pass clustering algorithm. We introduce two data structures, *Dictionary* that stores permanently kernels, and *Memory* where every newly emerged kernel is examined according to the density test and is transferred to Dictionary if it passes, or is discarded as noise if it fails. Both Dictionary and Memory are empty at the beginning, and then enriched while parsing the data.

Proposition 1 and 2 can be concisely implemented by maintaining a variable *activity* for every kernel in Memory. Activity is set to r when a new kernel is created and decreases by 1 after processing every data point in the stream. When a Memory kernel is matched, its activity value is increased by r; when this value exceeds a threshold ϕ, the corresponding kernel is transferred to the Dictionary as a permanent kernel. Therefore, a new kernel is maintained in Memory for least r parsed data points after initialization, and diminishes as outlier if it is not matched with any of the r input points, or it moves to Dictionary when it is matched frequently. Following the notations of Proposition 2, the (upper) threshold ϕ is,

$$\phi = rX_f \tag{7}$$

A different way to perceive the role of the activity counter is that it accounts for frequent matching with sufficient and rare matching with insufficient kernels, in dense and sparse areas respectively. So far we considered independence among the data, i.e., cluster assignments are evenly distributed along the stream, which is the worst case scenario: all kernels have on average similar, limited number of matches within a sub-sequence of size r, and their activity counters increase slower. If the cluster assignments are unevenly distributed due to temporal data dependencies of some nature, the kernels are matched more frequently within different sub-streams, and thus are more likely to be kept as permanent kernels faster, since their activities increase fast within short parsing periods.

C. *Subset size for successful clustering*

Consider a sub-stream formed by the first N^* data points of the input stream. Our goal is to find a sufficient kernel for every valid density peak by processing only these N^* samples, so that Dictionary will have enough subsequent data to update itself to the density peaks.

Consider a sufficient kernel c_k that covers N_k data points, with $\frac{N_k}{N} = p_k > p_f$. Let N_k^* be the number of instances, among the first N^* random samples, covered by kernel c_k. Then N_k^* follows Binomial Distribution $N_k^* \sim B(N^*, p_k)$. Using again the Central Limit Theorem, let $M = (N_k^* - N^* p_k)/\sqrt{N^* p_k q_k}$, where $q_k = 1 - p_k$ and M follows Standard Normal distribution, $M \sim \mathcal{N}(0,1)$. We hope that in the first N^* random samples, the instances covered by kernel c_k are more than 0.9 times of the expectation, $N_k^* > 0.9 N^* p_k$; the probability is,

$$\begin{aligned} & P\left\{N_k^* > 0.9N^*p_k\right\} \\ & = P\left\{\frac{N_k^* - N^*p_k}{\sqrt{N^*p_kq_k}} > \frac{0.9N^*p_k - N^*p_k}{\sqrt{N^*p_kq_k}}\right\} \\ & = P\left\{M > -\frac{0.1N^*p_k}{\sqrt{N^*p_kq_k}}\right\} = \Phi\left(\frac{0.1N^*p_k}{\sqrt{N^*p_kq_k}}\right), \end{aligned} \tag{8}$$

where Φ is the Cumulative Density Function of Standard Normal distribution. We expect $P\{N_k^* > 0.9N^*p_k\} > 1-\alpha$; if $\alpha = 0.01$, then from eq. (8), we have[1],

$$N^* > 497.3\frac{1-p_k}{p_k} > 497.3\frac{1-p_f}{p_f} \tag{9}$$

This equation indicates that our method does not 'prefer' small datasets, especially when the cluster population is very small. However, simple post-processing, such as data duplication or interpolation, can resolve this issue.

III. Experiments

In this section, we compare our method against state-of-the-art center-based clustering algorithms on realistic datasets. As mentioned in section I, our goal is to achieve top level performance in both speed and accuracy. Therefore, we consider not only stream clustering method but also accuracy-dedicated algorithms. All clustering algorithms considered in this paper are listed in Table I; in the complexity notations, N denotes data size, d is the data dimensionality, i is the number of iterations (usually $30 < i \leq 100$ except IGMM that usually takes hundreds of iterations for Gibbs sampling), K is the number of clusters, and C is the Coreset size in StreamKM++ and BICO. We also list the ability to discover the number of

[1]If $\Phi(x) = .99$ then $x = 2.23$

Table I
All the clustering algorithms in comparison

Algorithm	complexity	clusters	stream	#para
Our Method	$O(Nd)$	variant	yes	2
Density Peak[22]	$O(N^2d)$	variant	no	2
AP [13]	$O(iN^2d)$	variant	no	2
Infinite GMM[28]	$O(iNd^3)$	variant	no	5
K-means++[3]	$O(iKNd)$	fixed	no	1
GMM[20]	$O(iKNd)$	fixed	no	1
Mean Shift[9]	$O(iKNd)$	variant	no	2
StreamKM++[1]	$O(Nd + iKCd)$	fixed	yes	2
BICO [12]	$O(Nd + iKCd)$	fixed	yes	3
Pirch[16]	hierarchical	variant	yes	3
Birch [30]	hierarchical	variant	yes	5

Table II
Basic statistics of all datasets

dataset	clusters	points	dimension
Olivetti[24]	40	400	4096
Segmentation [29]	8	2310	18
LaSat [29]	6	6435	36
Letters [29]	26	20K	16
MNIST [18]	10	70K	784
Covertype [29]	7	581012	54
ALOI [14]	1000	108000	128
ILSVRC12 [23]	1000	1.3M	2048

clusters (indicated as 'variant' in the 'clusters' column), along with the number of user-specified parameters (#para). Among these algorithms, Density Peak, Affinity Propagation (AP) and Infinite GMM are accuracy-dedicated methods that are only suitable for relatively small datasets. K-means++, GMM and Mean Shift are popular algorithms applied in diverse domains and are also regarded as paradigms for approximation in many state-of-the-art stream clustering algorithms. StreamKM++ and BICO are two popular approximation algorithms of K-means++ that are able to cluster very large datasets with proven approximation accuracy. Pirch is a recent approach carefully designed for 'extreme clustering' [16], where the number of clusters is very large. Finally, Birch [30] provides a baseline for comparison.

We evaluate the above algorithms on 8 datasets of different domains, dimensionality, and sizes. Characteristics of these datasets are shown in Table II. 'Lasat' satellite data, 'Letters', 'Segmentation', and 'Covertype' are provided as benchmark datasets by UCI Machine Learning Repository [29]. 'ILSVRC' dataset consists 1.3 million of web images from 1000 categories, and we use the next-to-last layer of Inception Network [27] as the image description.

Here we consider two metrics for evaluating clustering accuracy: F1 score and mean accuracy. F1 score balances the impact between precision and recall. Mean accuracy only considers number of points correctly assigned to ground-truth clusters, which penalizes numbers of clusters higher than the ground-truth. We also report running times for all compared algorithms on a 2.7GHz Intel-i7 processor.

We ran all experiments 100 times, except Affinity Propagation (AP) and Density Peak, which are deterministic. To visualize more effectively accuracy, running time, and their

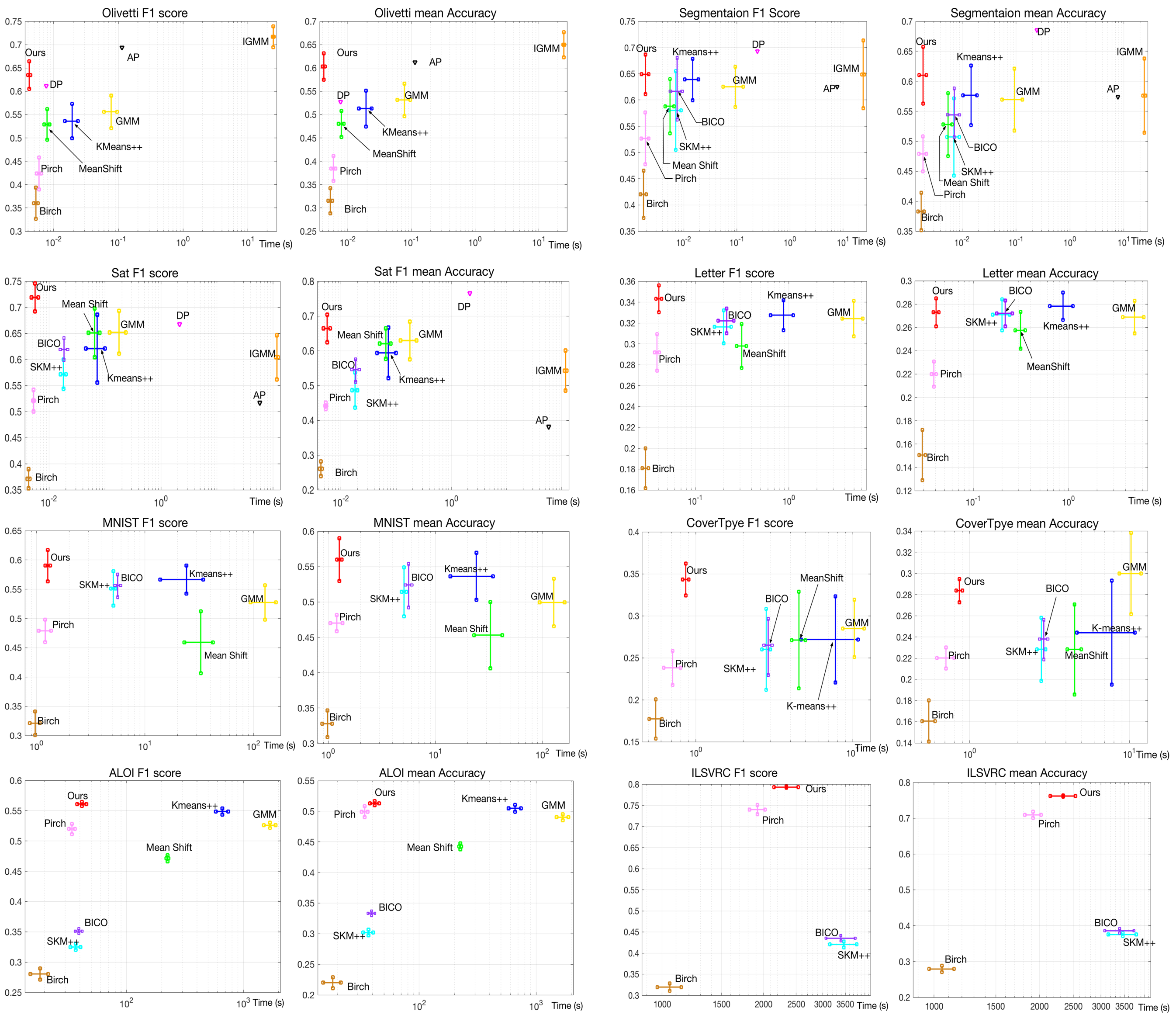

Figure 2. F1 scores, mean Accuracy, and execution times for all compared methods, for all employed datasets. Each method is denoted with bars of the same color across all the datasets (see text).

standard deviations, we use the 2-D plots shown in Fig 2, per dataset and per evaluation metric. The x-axis is log-scale running time (since some methods have much higher complexity), and y-axis is the evaluation metric. We use two bars for each method to show running time and accuracy, along with their standard deviations (bar lengths) over the 100 repetitions. The intersection point of each horizontal-vertical bar pair corresponds to the average values of the employed accuracy metric and the running time. Obviously, the best method resides at the top-left part of each plot.

In general, our algorithm always achieves high levels of accuracy across all datasets with the fastest running time. 'Olivetti', 'Sementation', and 'Sat' are small datasets, and thus we were able to test more complex algorithms (AP, DP, and IGMM). Every complex algorithm showed increased accuracy in at least one dataset, however no complex algorithm was able to perform well on all three small datasets. Our algorithm was not always the most accurate, yet it showed very consistent clustering quality. More importantly, these results reveal a promising alternative for high quality clustering: our method, AP, DP, and IGMM are all able to detect the number of clusters and require at least 2 user-specified parameters, with our method achieving high accuracy with the fastest running times. We should still note though that each algorithm has a preferred application area where our method may not always outperform the competition.

'Letter', 'MNIST', 'Covertype', and 'ALOI' are three medium-size datasets, where the superiority of our algorithm in terms of F1 score and speed can be clearly seen. However, considering mean accuracy, K-means++ showed better results in the 'Letter' dataset; the reason may be that incorrect assignments to fixed number of clusters were not penalized.

Similar observation can be made in the case of GMM in the 'CoverType' dataset. Despite these two peculiar outcomes of K-means and GMM, our algorithm consistently generates top quality regardless of data distribution, dataset size, and evaluation metric.

We evaluated our method on the 'ALOI' and 'ILSVRC' datasets that consist of very large numbers of clusters, and observed that it performed StreamKM++ and BICO with respect to all three evaluation criteria. The reason for the poor performance of the competition is that each cluster in 'ALOI' contains only 100 data points. If we maintain a coreset with 10% of the original dataset, each cluster in coreset contains merely 10 data points, which are too few to make K-means++ work successfully. However, increasing the coreset size will sacrifice efficiency. In this dataset, even the original K-means++ showed no better accuracy than our algorithm, with 15.8 times longer running time. 'ILVRC' was the largest dataset and we were only able to run stream algorithms. Again, our method shows superiority in terms of both F1 score and mean accuracy.

Finally, from the plots in Fig. 2 we observe that our method maintains overall smaller standard deviations of execution times compared to the other non-deterministic clustering algorithms. In general, among the competing methods, we observed that StreamKM++ and BICO produce well balanced performance between speed and accuracy. Birch is always the fastest but with limited accuracy (left-bottom in the plots). Pirch does not outperform StreamKM++ or BICO when the number of clusters is small but achieves major performance jump when the number of clusters is large (and the cluster sizes are relatively small).

IV. Conclusion

We showed that accuracy does not necessarily need to be sacrificed for faster speeds in clustering data streams. Samples of the original dataset can provide sufficient, rich information that can be used for saving computation. In this paper, we considered a window of the data stream that is used as a sample set to estimate emerging kernels which in turn are updated towards density peaks in a single pass. We evaluated the performance of our method and competing clustering algorithms, and the results show that our approach is an excellent alternative that improves the trade-off between accuracy and efficiency.

References

[1] M.R. Ackermann, M. Martens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, 'StreamKM++: A clustering algorithm for data streams,' *J of Experimental Algorithmics* 17:2.4, 2012. 1, 4

[2] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, 'A framework for clustering evolving data streams,' *in Proc. Int'l Conf. on Very Large Databases*, pp.81-92, 2003. 1

[3] D. Arthur and S. Vassilvitskii, 'k-means++: the advantages of careful seeding,' *in Proc. ACM-SIAM Symp. on Discrete Algorithms*, pp. 1027-1035, 2007. 4

[4] O. Bachem, M. Lucic, and A. Krause, 'Coresets for nonparametric estimation-the case of DP-means,' *in Proc. Int'l Conf. on Machine Learning*, pp. 209-217, 2015. 2

[5] O. Bachem, M. Lucic, S.H. Hassani, and A. Krause, 'Approximate K-Means++ in Sublinear Time,' *in Proc. AAAI Conf. on Artificial Intelligence*, pp. 1459-1467, 2016. 1

[6] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, 'Scalable k-means++,' *in Proc. VLDB Endowment*, pp. 622-633, 2012. 1

[7] M.F. Balcan, S. Ehrlich, and Y. Liang, 'Distributed k-means and k-median Clustering on General Topologies,' *in Proc. NIPS*, pp. 1995-2003, 2013. 1

[8] F. Cao, M. Estert, W. Qian, and A. Zhou, 'Density-based clustering over an evolving data stream with noise,' *in Proc. SIAM Int'l Conf. on Data Mining*, pp. 326-337, 2006. 1

[9] D. Comaniciu and P. Meer, 'Mean shift: A robust approach toward feature space analysis,' *IEEE Trans. PAMI*, 24(5):603-619, 2002. 2, 4

[10] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, 'Optimized big data K-means clustering using MapReduce,' *The Journal of Supercomputing*, 70(3):1249-1259, 2014. 1

[11] A. Ene, S. Im, and B. Moseley, 'Fast clustering using MapReduce,' *in Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 681-689, 2011. 1

[12] H. Fichtenberger, M. Gille, M. Schmidt, C. Schwiegelshohn, and C. Sohler, 'BICO: BIRCH meets coresets for k-means clustering,' *in Proc. European Symp. on Algorithms*, pp. 481-492, 2013. 1, 4

[13] B.J. Frey and D. Dueck, 'Clustering by passing messages between data points,' *Science*, 315(5814):972-976, 2007. 2, 4

[14] J.M. Geusebroek, G.J. Burghouts, A.W.M. Smeulders, 'The Amsterdam library of object images,' *Int'l J. of Computer Vision*, 61(1):103-112, 2005. 4

[15] S. Guha and N. Mishra, 'Clustering data streams,' Data Stream Management, Springer, pp. 169-187, 2016. 1

[16] A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum, 'A Hierarchical Algorithm for Extreme Clustering,' *in Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 255-264, 2017. 1, 4

[17] B. Kulis and M.I. Jordan, 'Revisiting k-means: New algorithms via Bayesian nonparametrics,' *in Proc. Int'l Conf. on Machine Learning*, 2011. 2

[18] Y. LeCun, C. Cortes, and C.J.C. Burges, The MNIST database of handwritten digits: http://yann.lecun.com/exdb/mnist/. 4

[19] E. Liberty, R. Sriharsha, and M. Sviridenko, 'An algorithm for online k-means clustering,' *in Proc. Workshop on Algorithm Engineering and Experiments*, pp. 81-89, 2016. 1

[20] T.K. Moon, 'The expectation-maximization algorithm,' *IEEE Signal Proc. Magazine*, 13(6):47-60, 1996. 4

[21] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, 'Streaming-data algorithms for high-quality clustering,' *in Proc. Int'l Conf. on Data Engineering*, pp. 685:694, 2002. 1

[22] A. Rodriguez and A. Laio, 'Clustering by fast search and find of density peaks,' *Science*, 344(6191):1492-1496, 2014. 2, 4

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, 'Imagenet large scale visual recognition challenge,' *Int'l J. of Computer Vision*, 115(3):211-252, 2015. 4

[24] F.S. Samaria, and A.C. Harter, 'Parameterisation of a stochastic model for human face identification,' *in Proc. IEEE Workshop on Applications of Computer Vision*, 1994. 4

[25] D. Sculley, 'Web-scale k-means clustering,' *in Proc. ACM Int'l conf. on World Wide Web*, pp. 1177-1178, 2010. 1

[26] J.A. Silva, E.R. Faria, R.C. Barros, E.R. Hruschka, A.C.P.L.F. de Carvalho, and J. Gama, 'Data stream clustering: A survey,' *ACM Computing Surveys*, 46(1):13, 2013. 1

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 'Rethinking the inception architecture for computer vision,' *in Proc. CVPR*, pp. 2818-2826, 2016. 4

[28] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei, 'Sharing clusters among related groups: Hierarchical Dirichlet processes,' *in Proc. NIPS*, pp. 1385-1392, 2005. 1, 4

[29] UCI Machine Learning Repository: http://archive.ics.uci.edu/ml 4

[30] T. Zhang, R. Ramakrishnan, and M. Livny, 'BIRCH: an efficient data clustering method for very large databases,' *in Proc. ACM SIGMOD*, pp. 103-114. 1996. 1, 4