Sampling in Dirichlet Process Mixture Models for Clustering Streaming Data

Or Dinari and Oren Freifeld

The Department of Computer Science, Ben-Gurion University, Israel

Streaming-data Clustering: Challenges

- Possibly-infinite data stream.
- New data arriving rapidly.
- Need to be able to provide an estimate of the model at any given time.
- Data statistics are usually non-stationary:
  - Clusters may appear/disappear.
  - Cluster properties (*e.g.*, centers) can change with time.
  - Cluster weights can change with time.

Streaming-data Clustering: Example

Figure 1: Video segmentation (example frames). Results shown for MiniBatch-Kmeans (denoted as MBK) with several different K values, as well as for ScStream (which inferred 80 clusters).

- Frames arrive rapidly.
- Each frame is a batch, consisting of 410K samples, each of which is a 5D vector (*RGBXY*).
- Cluster statistics change over time (*e.g.* the surfer location).
- Need consistent labeling across frames.

Can the Dirichlet Process Mixture Model (DPMM) be used for Clustering Streaming Data?

Pros:

- Can instantiate new clusters as the stream evolves.
- Highly flexible, can handle different data types (*e.g.* components can be Gaussians, multinomials, *etc.*).

Cons:

- Cannot handle concept drifts very well.
- Cannot forget old data.
- Even in SOTA methods (*e.g.*, [Dinari *et al.*, HPML 2019]), inference is too slow rapid data streams.

References

[1] Marcel R Ackermann et al. "Streamkm++ a clustering algorithm for data streams". In: *Journal of Experimental Algorithmics* (2012).

[2] Charu C Aggarwal et al. "A framework for clustering evolving data streams". In: *Proceedings 2003 the Very Large Data Bases conference*. Elsevier. 2003.

[3] Jason Chang and John W Fisher III. "Parallel sampling of DP mixture models using sub-cluster splits". In: *NeurIPS*. 2013.

[4] Yixin Chen and Li Tu. "Density-based clustering for real-time stream data". In: *ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007.

[5] Or Dinari et al. "Distributed MCMC inference in Dirichlet process mixture models using Julia". In: *IEEE CCGRID Workshop on High Performance Machine Learning*. 2019.

[6] Michael Hahsler and Matthew Bolaños. "Clustering data streams based on shared density between micro-clusters". In: *IEEE Transactions on Knowledge and Data Engineering* (2016).

[7] Matthew D Hoffman et al. "Stochastic variational inference." In: *Journal of Machine Learning Research* (2013).

[8] Yisroel Mirsky et al. "pcstream: A stream clustering algorithm for dynamically detecting and managing temporal contexts". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2015.

[9] David Sculley. "Web-scale k-means clustering". In: *WWW*. 2010.

[10] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM sigmod record* (1996).

The Proposed Solution: ScStream

- Based in part on a SOTA DPMM sampler [3] and its highly-efficient distributed implementation [5].
- Uses weighted batched sufficient statistics for calculating the posterior.
- Combines the iterative sampling with an additional iteration that uses a deterministic subroutine based on the predictive posterior.

ScStream Satisfies the Following Desiderata

- Fast.
- Does not need to revisit previously-processed data.
- Can modify the number of clusters as needed.
- Supports non-stationary cluster statistics.
- No label switching.
- Efficient memory use.

Weighted Batched Sufficient Statistics

Consider Gaussian components with a Normal Inverse Wishart prior, $\mathrm{NIW}(\kappa, \boldsymbol{m}, \nu, \boldsymbol{\Psi})$. In the DPMM, the posterior for cluster k, $\mathrm{NIW}(\kappa_k^*, \boldsymbol{m}_k^*, \nu_k^*, \boldsymbol{\Psi}_k^*)$. is calculated via

$$\kappa_k^* = \kappa + N_k \qquad \nu_k^* = \nu + N_k \qquad \boldsymbol{m}_k^* = \frac{1}{\kappa_k^*}\kappa\boldsymbol{m} + \sum\nolimits_{i=1}^{N} \boldsymbol{x}_i \mathbb{1}_{z_i=k} \qquad \boldsymbol{\Psi}_k^* = \frac{1}{\nu_k^*}\nu\boldsymbol{\Psi} + \sum\nolimits_{i=1}^{N} \boldsymbol{x}_i\boldsymbol{x}_i^T \mathbb{1}_{z_i=k} \tag{1}$$

Since we work with batches, we replace these expressions with

$$\kappa_k^* = \kappa + N_k^B \qquad \nu_k^* = \nu + N_k^B \qquad \boldsymbol{m}_k^* = \frac{1}{\kappa_k^*}\left(\kappa\boldsymbol{m} + \sum\nolimits_{b=q}^{B}\left[\mathcal{K}(B,b)\sum\nolimits_{\boldsymbol{x}_i\in X_b}\boldsymbol{x}_i\mathbb{1}_{z_i=k}\right]\right) \qquad \boldsymbol{\Psi}_k^* = \frac{1}{\nu_k^*}\left(\nu\boldsymbol{\Psi} + \sum\nolimits_{b=q}^{B}\left[\mathcal{K}(B,b)\sum\nolimits_{\boldsymbol{x}_i\in X_b}\boldsymbol{x}_i\boldsymbol{x}_i^T\mathbb{1}_{z_i=k}\right]\right) \tag{2}$$

where $\mathcal{K}(B,b)$ is a weighting function (the older the batch, the lower its weight), B is the index of the current batch and b is the index of an older batch. Other exponential families (*e.g.*, multinomials) are handled similarly.

The Algorithm

Algorithm 1: ScStream

Input: H, α, $\mathcal{K}$, ϵ,T
Data: Stream $\boldsymbol{X}$

- $X_1 \leftarrow \boldsymbol{X}.next$
- $C_1 \leftarrow X_1$
- $K \leftarrow 1$
- Randomly partition C_1 into subclusters $C_{1,1}$ and $C_{1,2}$
- $q \leftarrow 1$
- Extract $h_1^{1:1} = (s_1^1, n_1^1)$, $\bar{h}_{1,1}^{1:1} = (\bar{s}_{1,1}^1, \bar{n}_{1,1}^1)$ and $\bar{h}_{1,2}^{1:1} = (\bar{s}_{1,2}^1, \bar{n}_{1,2}^1)$ from $(C_1, C_{1,1}, C_{1,2})$
- $\mathcal{M} \leftarrow (h_1^{1:1}, \bar{h}_{1,1}^{1:1}, \bar{h}_{1,2}^{1:1})$
- **while** *Not Converged* **do**
  - $K, \mathcal{M} \leftarrow$algorithm 2$(X_1; H, \alpha, K, \mathcal{K}, \infty, _, q, B, \mathcal{M})$
- **while** $X_B \leftarrow \boldsymbol{X}.next$ **do**
  - $(h_k^{q:(B-1)}, \bar{h}_{k,1}^{q:(B-1)}, \bar{h}_{k,2}^{q:(B-1)})_{k=1}^K \leftarrow \mathcal{M}$
  - $q \leftarrow \min\{b : b \in \{1,\dots,B\}, \mathcal{K}(B,b) > \epsilon\}$
  - $\mathcal{M} \leftarrow (h_k^{q:B-1}, \bar{h}_{k,1}^{q:B-1}, \bar{h}_{k,2}^{q:B-1})_{k=1}^K$
  - **for** $t = 1 : T+1$ **do**
    - $K, \mathcal{M} \leftarrow$algorithm 2$(X_B; H, \alpha, \dots, t, q, B, \mathcal{M})$
  - Yield $\mathcal{M}$

Algorithm 2: Iteration of the Modified DPMM Sampler

Input: H, α, K, $\mathcal{K}$,T,t, q,B, $\mathcal{M} = (h_k^{q:B}, (\bar{h}_{k,j}^{q:B})_{j\in\{1,2\}})_{k=1}^K$
Output: K',$\mathcal{M}'$
Data: X_B

- **if** $t < T+1$ **then**
  - $(h_1^{q:B}, \bar{h}_{1,1}^{q:B}, \bar{h}_{1,2}^{q:B}) \leftarrow \mathcal{M}$
  - Compute $(S_k^B)_{k=1}^K$ and $(N_k^B)_{k=1}^K$
  - 1 iteration of the restricted sampler using $(S_k^B)_{k=1}^K$ and $(N_k^B)_{k=1}^K$
- **else**
  - $\boldsymbol{\pi} \leftarrow \left(\frac{N_1^B}{\sum_{k=1}^K N_k^B+\alpha}, \dots, \frac{N_K^B}{\sum_{k=1}^K N_k^B+\alpha}, \frac{\alpha}{\sum_{k=1}^K N_k^B+\alpha}\right)$
  - **for** $k \in \{1,\dots,K\}$ **do**
    - $\bar{\boldsymbol{\pi}}_k \leftarrow \left(\frac{\frac{\alpha}{2}+\bar{N}_{k,1}^B}{\alpha+\sum_{s=\{1,2\}}\bar{N}_{k,s}^B}, \frac{\frac{\alpha}{2}+\bar{N}_{k,2}^B}{\alpha+\sum_{s=\{1,2\}}\bar{N}_{k,s}^B}\right)$
  - **for** $x_i \in X_B$ **do**
    - $z_i \leftarrow \underset{k\in\{1,\dots,K\}}{\arg\max}\, \pi_k p(z_i = k|\boldsymbol{x}_i, H, S_k^B, N_k^B)$
    - $\bar{z}_i \leftarrow \underset{j\in\{1,2\}}{\arg\max}\, \bar{\pi}_{z_i} p(\bar{z}_i = j|\boldsymbol{x}_i, H, \bar{S}_{z_i,j}^B, \bar{N}_{z_i,j}^B)$
- **for** $k \in \{1,\dots,K\}$ **do**
  - Extract (s_k^B, n_k^B), $(\bar{s}_{k,1}^B, \bar{n}_{k,1}^B)$ and $(\bar{s}_{k,2}^B, \bar{n}_{k,2}^B)$ (from C_k, $\bar{C}_{k,1}$ and $\bar{C}_{k,2}$, respectively) and update $(h_k^{q:B}, \bar{h}_{k,1}^{q:B}, \bar{h}_{k,2}^{q:B})$ accordingly
- **for** $k \in \{1,\dots,K\}$ **do**
  - Propose splitting C_k to its subsclusters and accept the split with probability $\min(1, H_{\text{split}})$
- **for** $k, k' \in \{1,\dots,K\}$ **do**
  - Propose merging C_k and $C_{k'}$ and accept the merge with probability $\min(1, H_{\text{merge}})$
- $\mathcal{M}' \leftarrow (h_k^{q:B}, (\bar{h}_{k,j}^{q:B})_{j\in\{1,2\}})_{k=1}^{K'}$ where K' is the new number of clusters

Our ScStream Code is Publicly Available with Support for either Julia or Python

- **Julia:** github.com/BGU-CS-VIL/DPMMSubClustersStreaming.jl
- **Python:** github.com/BGU-CS-VIL/dpmmpythonStreaming

Experiments and Results

		BIRCH	CluStream†	D-Stream	DBSTREAM	StreamKM++†	Mini Batch K-Means†	pcStream	SoVB	ScStream (Ours)	DPMM Sampler
2D Gaussians	ARI:	.81 ± .12	.86 ± .11	.88 ± .16	.90 ± .11	.53 ± .11	.82 ± .09	.60 ± .12	.58 ± .11	**.93 ± .08**	.92 ± .14
	NMI:	.89 ± .04	.94 ± .03	.94 ± .05	.94 ± .04	.71 ± .05	.89 ± .03	.76 ± .07	.75 ± .05	**.95 ± .03**	.94 ± .10
	Purity:	.83 ± .06	**.94 ± .03**	.91 ± .09	.91 ± .06	.57 ± .05	.83 ± .05	.70 ± .08	.68 ± .06	.92 ± .05	.91 ± .10
	F-Measure:	.84 ± .10	.88 ± .09	.90 ± .13	.91 ± .09	.61 ± .08	.85 ± .08	.66 ± .10	.65 ± .09	**.94 ± .07**	.93 ± .11
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	.48 ± .00	.37	.52	**.68 ± .01**	N/A
CoverType	ARI:	.07 ± .08	.10 ± .07	.07 ± .11	.10 ± .13	.09 ± .09	.07 ± .06	.03 ± .02	.10 ± .09	**.15 ± .11**	.10 ± .11
	NMI:	.14 ± .09	.19 ± .09	.19 ± .11	.18 ± .15	.15 ± .08	.13 ± .06	.20 ± .07	.13 ± .10	**.21 ± .14**	.16 ± .12
	Purity:	.66 ± .10	.71 ± .11	.70 ± .10	.68 ± .11	.68 ± .11	.66 ± .12	**.79 ± .08**	.66 ± .13	.71 ± .11	.67 ± .12
	F-Measure:	.44 ± .10	.33 ± .05	.58 ± .14	**.60 ± .13**	.42 ± .10	.37 ± .06	.11 ± .05	.48 ± .08	.47 ± .08	.48 ± .09
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	.06 ± .01	.08	.01	**.13 ± .01**	N/A
ImageNet100	ARI:	.21 ± .11	.30 ± .13	N/A	.13 ± .15	.55 ± .15	.49 ± .17	.20 ± .09	.31 ± .18	**.63 ± .19**	.64 ± .28
	NMI:	.35 ± .11	.45 ± .09	N/A	.22 ± .17	.62 ± .09	.58 ± .12	.33 ± .08	.45 ± .20	**.69 ± .15**	.72 ± .24
	Purity:	.64 ± .12	.75 ± .12	N/A	.43 ± .13	**.91 ± .06**	.87 ± .09	.66 ± .10	.49 ± .13	.78 ± .14	.74 ± .22
	F-Measure:	.39 ± .08	.44 ± .10	N/A	.43 ± .09	.62 ± .14	.57 ± .15	.33 ± .09	.55 ± .11	**.73 ± .12**	.76 ± .17
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	**.57 ± .02**	.26	.23	.48 ± .01	N/A
ImageNet1K	ARI:	N/A	.30 ± .14	N/A	.30 ± .16	N/A	.45 ± .12	.19 ± .07	.00 ± .02	**.62 ± .17**	N/A
	NMI:	N/A	.45 ± .10	N/A	.40 ± .14	N/A	.59 ± .07	.38 ± .06	.00 ± .02	**.68 ± .13**	N/A
	Purity:	N/A	.74 ± .14	N/A	.62 ± .13	N/A	**.97 ± .03**	.76 ± .08	.25 ± .04	.78 ± .13	N/A
	F-Measure:	N/A	.44 ± .09	N/A	.48 ± .11	N/A	.51 ± .12	.28 ± .09	.38 ± .04	**.72 ± .12**	N/A
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	**.63 ± .01**	.30	.00	.41 ± .02	N/A
100D Multinomials	ARI:	N/A	.00 ± .01	N/A	.00 ± .00	.34 ± .24	.41 ± .24	N/A	.21 ± .14	**.78 ± .24**	.45 ± .22
	NMI:	N/A	.11 ± .05	N/A	.00 ± .00	.65 ± .16	.69 ± .16	N/A	.52 ± .14	**.89 ± .12**	.62 ± .30
	Purity:	N/A	.09 ± .03	N/A	.03 ± .00	.53 ± .25	.61 ± .25	N/A	.31 ± .15	**.84 ± .20**	.53 ± .25
	F-Measure:	N/A	.04 ± .01	N/A	.04 ± .01	.35 ± .24	.42 ± .24	N/A	.23 ± .13	**.78 ± .24**	.46 ± .22
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	.54 ± .01	N/A	.27	**.72 ± .01**	N/A
20NewsGroup	ARI:	N/A	.00 ± .00	N/A	N/A	.01 ± .00	.01 ± .00	N/A	.06 ± .01	**.13 ± .01**	.12 ± .01
	NMI:	N/A	.12 ± .02	N/A	N/A	.07 ± .01	.09 ± .01	N/A	.20 ± .02	**.36 ± .03**	.33 ± .02
	Purity:	N/A	.13 ± .02	N/A	N/A	.11 ± .01	.12 ± .01	N/A	.13 ± .01	**.28 ± .02**	.24 ± .02
	F-Measure:	N/A	.10 ± .00	N/A	N/A	.10 ± .00	.09 ± .00	N/A	.14 ± .01	**.20 ± .01**	.19 ± .01
	Full-NMI:	N/A	N/A	N/A	N/A	N/A	.05 ± .01	N/A	.17	**.32 ± 0.03**	N/A

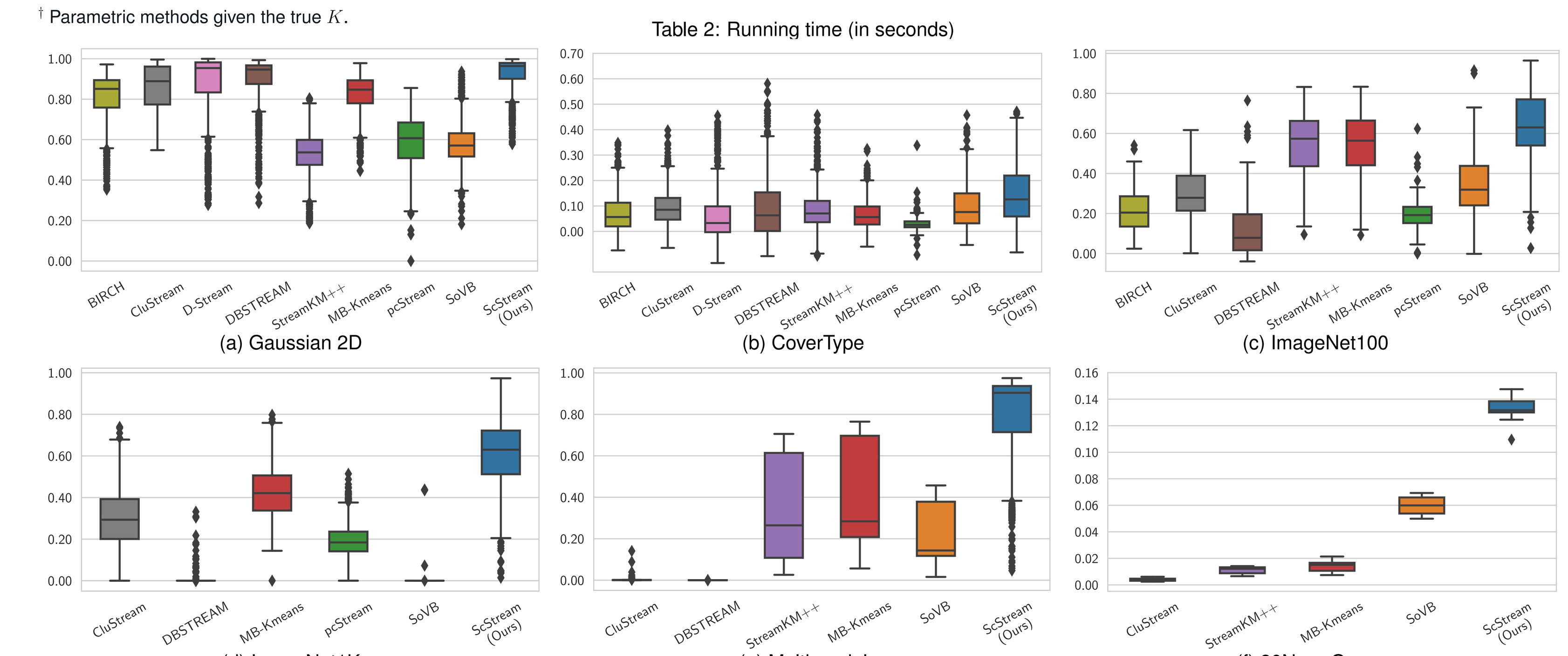
† Parametric methods given the true K.

Table 1: Comparing our method (ScStream) with BIRCH [10], CluStream [2], D-Stream [4], DBSTREAM [6], StreamKM++ [1], Mini Batch K-Means [9], pcStream [8], SoVB [7]. Also included is DPMM sampler [5]. N/A indicates that a method did not scale enough or lacks support for the data type.

	BIRCH	CluStream†	D-Stream	DBSTREAM	StreamKM++†	Mini Batch K-Means†	pcStream	SoVB	ScStream (Ours)	DPMM Sampler
2D Gaussians	112.5	31.3	24.7	17.0	15.4	1.4	1020.7	53.3	22.9	589.5
CoverType	95.8	45.9	1723.8	12.1	25.5	0.8	1610.3	115.6	6.1	254.8
ImageNet100	57.9	66.7	N/A	65.2	242.7	12.0	15.7	100.5	23.1	1039.9
ImageNet1K	N/A	1454	N/A	814	N/A	148	195	9219	1005	N/A
100D Multinomials	N/A	44.7	N/A	12.9	25.5	0.8	N/A	115.6	23.5	254.8
20NewsGroup	N/A	71.9	N/A	N/A	61.1	0.2	N/A	3.1	12.7	122.6

† Parametric methods given the true K.

Table 2: Running time (in seconds)



(a) Gaussian 2D (b) CoverType (c) ImageNet100 (d) ImageNet1K (e) Multinomial (f) 20NewsGroups

Figure 2: Box plots of the ARI metric for each of the experiments.