Coresets for Time Series Clustering

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of constructing coresets for clustering problems with time series data. This problem has gained importance across many fields including biology, medicine, and economics due to the proliferation of sensors for real-time measurement and rapid drop in storage costs. In particular, we consider the setting where the time series data on N entities is generated from a Gaussian mixture model with autocorrelations over k clusters in $\mathbb{R}^d$. Our main contribution is an algorithm to construct coresets for the maximum likelihood objective for this mixture model. Our algorithm is efficient, and, under a mild assumption on the covariance matrices of the Gaussians, the size of the coreset is independent of the number of entities N and the number of observations for each entity, and depends only polynomially on k, d and $1/\varepsilon$, where ε is the error parameter. We empirically assess the performance of our coresets with synthetic data.

1 Introduction

A multivariate time series dataset, represented as $\mathcal{X} = \{X_i = (x_{i1}, \ldots, x_{i,T_i}) \subset \mathbb{R}^{T_i \times d} \mid i \in [N]\}$, where N is the number of entities, T_i is the number of time periods corresponding to entity i and d is the number of features, tracks features of a cross-section of entities longitudinally over time. Such data is also referred to as panel data [8] and has seen rapid growth due to proliferation of sensors, IOT and wearables that facilitate real time measurement of various features associated with entities and the rapid drop in storage costs. Specific examples of time series data include biomedical measurements (e.g., blood pressure and electrocardiogram), epidemiology and diffusion through social networks, weather, consumer search, content browsing and purchase behaviors, mobility through cell phone and GPS locations, stock prices and exchange rates in finance [2].

Computational problems of interest with time series data include pattern discovery [46], regression/prediction [36], forecasting [9], and clustering [44, 28, 2] which arises in applications such as anomaly detection [43]. Though clustering is a central and well-studied problem in unsupervised learning, most standard treatments of clustering tend to be on static data with one observation per entity [32, 11]. Time series clustering introduces a number of additional challenges and is an active area of research, with many types of clustering methods proposed [15]; direct methods on raw data, indirect methods based on features generated from the raw data, and model based clustering where the data are assumed to be generated from a model. For surveys of time series clustering, see [44, 2].

We focus on model-based time series clustering using a likelihood framework that naturally extends clustering with static data [11] to time series data [44], where each multivariate time series is generated by one of k different parametrically specified models. A prevalent approach is to assume a finite mixture of data generating models, with Gaussian mixture models being a common specification [47, 49]. Roughly, the problem is to partition $\mathcal{X}$ probabilistically into k clusters, grouping those series generated by the same time series model into one cluster. Generically, the model-based k-clustering problem can be formulated as: $\arg\max \sum_{i \in [N]} \ln \sum_{l \in [k]} \alpha_l \cdot p_i(\mathcal{X}|\theta^{(l)})$, where α_l is the probability

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

of a time series belonging to cluster l and $p_i(\mathcal{X}|\theta^{(l)}$ is the likelihood of the data of entity i, given that the data is generated from model l (with parameters $\theta^{(l)}$). Temporal relationships in time series are commonly modeled as autocorrelations (AR)/moving averages (MA) [20, 56] that account for correlations across observations, and hidden Markov models (HMM) where the underlying data generating process is allowed to switch periodically [21, 57]. This paper focuses on the case when the data generation model for each segment $l \in [k]$ is a multivariate Gaussian with autocorrelations. More formally, the generative model for each cluster is from a Gaussian mixture: $x_{it} := \mu^{(l)} + e_{it}$, where $e_{it} = N(0, \Sigma^{(l)}) + \Lambda^{(l)} e_{i,t-1}$, where $N(0, \Sigma^{(l)})$ captures the mixture of Gaussian distributions from which entity level observations are drawn, and Λ captures the correlation between two successive observations through an AR(1) process [29, 41]. Overall, this model can be represented with cluster level model parameters $\theta^{(l)} = \{\mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)}\}$ and cluster probability α_l.

Time series datasets are also much larger than static datasets [44]. For instance, as noted in [39], ECG (electrocardiogram) data requires 1 GB/hour, a typical weblog requires 5 GB/week. Such large sizes lead to high storage costs and also make it necessary to work with a subset of the data to conduct the clustering analysis to be computationally practical. Further, long time series data on entities also entails significant data security and privacy risks around the entity as it requires longer histories of entity behaviors to be stored and maintained [24, 38]. Thus, there is significant value for techniques that can produce approximately similar clustering results with smaller samples as with complete data.

Coresets have emerged as an effective tool to both speed up and reduce data storage by taking into account the objective and carefully sampling from the full dataset in a way that any algorithm run on the sampled set returns an approximate solution for the original objective with guaranteed bounds on approximation error [31]. Coresets have been developed for both unsupervised clustering (e.g., k-means, Gaussian mixture models) and supervised machine learning (e.g., regression) methods for static data; for surveys on coresets for static data, see [7, 22]. A natural question is whether coresets can be also useful for addressing the time series clustering problem. Recently, there is evidence that small coresets can be efficiently constructed for time series data, albeit for regressions with time series (panel) data [34]. However, as we explain in Section 2, there are challenges in extending coresets for static data clustering and time series (panel) data regressions for time series clustering.

Our contributions. We study coresets for a general class of time series clustering in which all entities are drawn from Gaussian mixtures with autocorrelations (as described above; see Problem 3.1). We first present a definition of coresets for the log-likelihood k-clustering objective for this mixture model. One issue is that this objective cannot be decomposed into a summation of entity-time objectives due to the interaction of the $\ln$ term from log-likelihood and the $\exp$ term from Gaussians (Problem 3.1). To estimate this objective using a coreset, we introduce an analogous clustering objective on a subset of data (Definition 3.3). Our main result is an algorithm to construct coresets for this objective for the aforementioned mixture model (Theorem 4.3). Our algorithm is efficient and, assuming that the condition number of the covariance matrices of the Gaussians ($\Sigma^{(l)}$) are bounded and the eigenvalues of the autocorrelation matrices ($\Lambda^{(l)}$) are bounded away from 1 (Assumption 4.1), the size of our coreset is independent of the number of entities N and the number of observations T_is; it only polynomially depends on k, d and $1/\varepsilon$, where ε is the error parameter (Theorem 4.3).

Our coreset construction (Algorithm 1) leverages the Feldman-Langberg (FL) framework [26]. Due to multiple observations for each entity and associated autocorrelations, the objective is non-convex and quite complicated in contrast to the static setting where objective only contains one observation drawn from Gaussian mixtures for each entity. Thus, bounding the "sensitivity" (Lemma 4.6) of each entity, which is a key step in coreset construction using the FL-framework, becomes challenging. We handle this technical difficulty by 1) upper-bounding the maximum effect of covariances and autocorrelations by using the observation that the gap between the clustering objectives with and without Σ and Λ is always constant, and 2) reducing the Gaussian mixture time series clustering problem to a certain k-means clustering problem whose total sensitivity is upper bounded.

Empirically, we assess the performance of our coreset on synthetic data for a three cluster problem (Section 5). We find that our coreset performs better relative to uniform sampling on both data storage and computation speed for a range of accuracy guarantees: To achieve a similar fit with the full data, our coreset needs fewer entity-time observations (<6%); the computation time for a given level of accuracy reduces by a 2x-8x factor when compared to uniform. We note that the performance advantage of our coreset is greater when there is more time series data relative to entities ($T_i \gg N$).

2 Related work

Time series or panel data analysis are important in many fields– biological [18], engineering [3], economics [48] and the social sciences [54]. Clustering has been a central problem in unsupervised learning and also has a long history of use across many fields; for a historical overview of applications across fields, see [12]. While most standard treatments of clustering in machine learning tend to be on static data [32, 11], there is by now a significant and rapidly growing literature on time series clustering; recent surveys include [44, 2]. While there are many coresets for algorithms on static data (including for clustering), there has been little work on coresets on time series data (see below).

Coresets for clustering. Coresets for various clustering objectives on static data have been well studied, including k-median [31, 17, 50, 35, 19], k-means [31, 17, 27, 13, 10, 50, 35, 19], and k-center [1, 30]. [45, 25] present coreset constructions for clustering problems on static data generated using Gaussian mixture models. [45] constructs a coreset of size $\mathsf{poly}(k, d, 1/\varepsilon)$ under the same boundedness assumption as ours. [25] remove this assumption and construct a coreset whose size additionally depends on $\mathsf{poly} \log N$, but require another assumption that all x_is are integral and in a bounded range. In contrast, we consider a generalized problem where each entity has multiple observations over time and accounts for autocorrelations across successive observations. The new idea is to reduce the GMM time series clustering problem (Problem 3.1) to a k-means clustering problem whose points are linear combinations of those in X_i (Lemma A.12), and show that the effect of autocorrelation parameters Λ on the entity objectives can be upper bounded (Lemma A.11).

Regression with time series data. [34] construct coresets for a regression problem, called GLSE_k, for time-series data including autocorrelations and a k-partition of entities. This is a supervised learning task that includes an additional label y_{it} for each entity-time pair. It aims to find a k-partition of entities and additional regression vectors $\beta^{(l)} \in \mathbb{R}^d$ for each partition $l \in [k]$ such that a certain linear combination of terms $\|y_{it} - (\beta^{(l)})^\top x_{it}\|_2^2$ is minimized. In contrast, our problem is an unsupervised learning task, estimates points x_{it} by Gaussian means μ instead of β, and includes covariances Σ between different features. The definitions of coresets in [34] and ours consist of entity-time pairs but our coresets need to include multiple weight functions (w for entities and $w^{(i)}$ for time periods of selected entity i; see Definition 3.3) instead of one weight function as in the regression problem. This is to deal with the interaction of the $\ln$ term from the log-likelihood and the $\exp$ term from the Gaussians. [34]'s result relies on an assumption that the regression objective of each entity is in a bounded range over the parameter space. This may not be satisfied in our setting since the clustering objective of an entity may be unbounded when all Gaussian means $\mu^{(l)}$ are far away from observations of this entity. To bypass this, we reduce our problem to a k-means clustering problem (Definition 4.2), whose total sensitivity is provably $O(k)$ (Lemma A.12), by upper bounding the affect of covariance and autocorrelation parameters on the clustering objectives of entities (Lemma A.11) under the assumption that the condition number of covariance matrix is bounded (Assumption 4.1), which provides an upper bound for the total sensitivity of entities on our problem (Lemma 4.6).

3 Clustering model and coreset definition

Given $\mathcal{X} = \left\{X_i = (x_{i1}, \ldots, x_{i,T_i}) \subset \mathbb{R}^{T_i \times d} \mid i \in [N]\right\}$, we first model a general class of time series clustering problems. Then we specify our setting with Gaussian mixture time series data (Problem 3.1), and define coresets accordingly (Definition 3.3).

Clustering with time series data. Given an integer $k \geq 1$, let $\Delta_k \subset \mathbb{R}^k$ denote a probability simplex satisfying that for any $\alpha \in \Delta_k$, we have $\alpha_i \in [0, 1]$ for $i \in [k]$ and $\sum_{l \in [k]} \alpha_i = 1$. Let $\mathcal{P}$ denote a parameter space where each $\theta \in \mathcal{P}$ represents a specific generative model of time series data. For each entity $i \in [N]$ and model $\theta \in \mathcal{P}$, define $p_i(\mathcal{X} \mid \theta) := \Pr[X_i \mid \theta]^{1/T_i}$ to be the average likelihood/realized probability of X_i from model θ. The ratio $1/T_i$ is used to normalize the contribution to the objective of entity i due to different lengths T_i. The time series clustering problem is to compute $\alpha \in \Delta^k$ and $\theta = (\theta^{(1)}, \ldots, \theta^{(k)}) \in \mathcal{P}^k$ that minimizes the negative log-likelihood, i.e., $-\sum_{i \in [N]} \ln \sum_{l \in [k]} \alpha_l \cdot p_i(\mathcal{X} \mid \theta^{(l)})$. Here, for each $l \in [k]$, α_l represents the probability that each time series is generated from model $\theta^{(l)}$. Note that the clustering objective depends on the choice of model family $\mathcal{P}$. In this paper, we consider the following specific model family $\mathcal{P}$ that each time series is generated from Gaussian mixtures.

GMM clustering with time series data. Let $\alpha \in \Delta_k$ be a given probability vector. Let $\mathcal{S}^d$ denote the collection of all symmetric positive definite matrices in $\mathbb{R}^{d\times d}$. For each $i \in [N]$, with probability α_l ($l \in [k]$), let $x_{it} := \mu^{(l)} + e_{it}$ for each $t \in [T_i]$ where $\mu^{(l)} \in \mathbb{R}^d$ represents a Gaussian mean and $e_{it} \in \mathbb{R}$ represents the error term drawn from the following normal distribution: $e_{it} := \Lambda^{(l)} e_{i,t-1} + N(0, \Sigma^{(l)})$, where $\Lambda^{(l)} \in \mathcal{S}^d$ is an AR(1) autocorrelation matrix and $\Sigma^{(l)} \in \mathcal{S}^d$ is the covariance matrix of a multivariate Gaussian distribution. Now we let $\mathcal{P} := \mathbb{R}^d \times \mathcal{S}^d \times \mathcal{S}^d$ and note that each Gaussian generative model can be represented by a tuple $\theta = (\mu, \Sigma, \Lambda) \in \mathcal{P}$. Moreover, we have that the realized probability of each entity $i \in [N]$ is

$$p_i(\mathcal{X} \mid \theta) = p_i(\mathcal{X} \mid \mu, \Sigma, \Lambda) := \frac{\exp(-\frac{1}{2T_i}\psi_i(\mu,\Sigma,\Lambda))}{(2\pi)^{d/2}|\Sigma|^{1/2}}, \tag{1}$$

where $|\Sigma|$ is the determinant of Σ and $\psi_i(\mu, \Sigma, \Lambda) := \sum_{t\in[T_i]} \psi_{it}(\Sigma, \Lambda)$ with

$$\psi_{i1}(\Sigma, \Lambda) := (x_{i1} - \mu)^\top \Sigma^{-1}(x_{i1} - \mu) - (\Lambda(x_{i1} - \mu))^\top \Sigma^{-1} (\Lambda(x_{i1} - \mu)), \text{ and}$$

$$\psi_{it}(\Sigma, \Lambda) := ((x_{it} - \mu) - \Lambda(x_{i,t-1} - \mu))^\top \Sigma^{-1} ((x_{it} - \mu) - \Lambda(x_{i,t-1} - \mu)), \forall\ 2 \le t \le T_i.$$

We note that each sub-function ψ_{it} contains at most two entity-time pairs: x_{it} and $x_{i,t-1}$. Our Gaussian mixture time series model gives rise in the following clustering problem.

Problem 3.1 (**Clustering with GMM time series data).** *Given a time series dataset $\mathcal{X} = \{X_i = (x_{i1}, \ldots, x_{i,T_i}) \subset \mathbb{R}^{T_i\times d} \mid i \in [N]\}$ and integer $k \ge 1$, the GMM time series clustering problem is to compute $\alpha \in \Delta^k$ and $\theta = (\mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)})_{l\in[k]} \in \mathcal{P}^k$ that minimize*

$$f(\alpha, \theta) := \sum_{i\in[N]} f_i(\alpha, \theta) = -\sum_{i\in[N]} \ln \sum_{l\in[k]} \alpha_l \cdot p_i(\mathcal{X} \mid \mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)}).$$

From Equation (1) it follows that the coefficient of each Gaussian before the $\exp$ operation is $\frac{\alpha_l}{(2\pi)^{d/2}|\Sigma^{(l)}|^{1/2}}$, whose summation $Z(\alpha, \theta) := \sum_{l\in[k]} \frac{\alpha_l}{(2\pi)^{d/2}|\Sigma^{(l)}|^{1/2}}$ is not a fixed value and depends on $\Sigma^{(l)}$s. To remove this dependence on the coefficients, we define $\alpha'_l(\theta) := \frac{\alpha_l}{(2\pi)^{d/2}|\Sigma^{(l)}|^{1/2} Z(\alpha,\theta)}$ to be the normalized coefficient for $l \in [k]$, define offset function $\phi : \Delta^k \times \mathcal{P}^k \to \mathbb{R}$ to be $\phi(\alpha, \theta) := -N \ln Z(\alpha, \theta)$, and define $f'(\alpha, \theta) := -\sum_{i\in[N]} \ln \sum_{l\in[k]} \alpha_l \cdot \exp(-\frac{1}{2T_i}\psi_i(\mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)}))$ whose summation of coefficients before the $\exp$ operations is 1. This idea of introducing f' also appears in [23, 25], which is useful for coreset construction. This leads to the following observation.

Observation 3.2. *For any $\alpha \in \Delta^k$ and $\theta \in \mathcal{P}^k$, $f(\alpha, \theta) = f'(\alpha'(\theta), \theta) + \phi(\alpha, \theta)$.*

Our coreset definition. The clustering objective f can only be decomposed into the summation of f_is instead of sub-functions w.r.t. entity-time pairs. A simple idea is to select a collection of weighted entities as a coreset. However, in doing so, the coreset size will depend on T_is and fail to be independent of both N and T_is. To address this problem, we define our coreset as a collection of weighted entity-time pairs, which is similar to [34]. Let $P_{\mathcal{X}} := \{(i, t) \mid i \in [N], t \in [T_i]\}$ denote the collection of indices of all x_{it}. Given $S \subseteq P$, we let $I_S := \{i \in [N] \mid \exists t \in [T_i], s.t., (i, t) \in S\}$ denote the collection of entities that appear in S. Moreover, for each $i \in I_S$, we let $J_{S,i} := \{t \in [T_i] : (i, t) \in S\}$ denote the collection of observations for entity i in S.

Definition 3.3 (**Coresets for GMM time series clustering).** *Given a time series dataset $\mathcal{X} = \{X_i = (x_{i1}, \ldots, x_{i,T_i}) \subset \mathbb{R}^{T_i\times d} \mid i \in [N]\}$, constant $\varepsilon \in (0, 1)$, integer $k \ge 1$, and parameter space $\Delta^k \times \mathcal{P}^k$, an ε-coreset for GMM time series clustering is a weighted set $S \subseteq P$ together with weight functions $w : I_S \to \mathbb{R}_{\ge 0}$ and $w^{(i)} : J_{S,i} \to \mathbb{R}_{\ge 0}$ for $i \in I_S$ such that for any $\alpha \in \Delta^k$ and $\theta = (\mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)})_{l\in[k]} \in \mathcal{P}^k$, $f'_S(\alpha, \theta) := -\sum_{i\in I_S} w(i) \cdot \ln \sum_{l\in[k]} \alpha_l \cdot \exp(-\frac{1}{2T_i} \sum_{t\in J_{S,i}} w^{(i)}(t) \cdot \psi_{it}(\mu^{(l)}, \Sigma^{(l)}), \Lambda^{(l)})) \in (1 \pm \varepsilon) \cdot f'(\alpha, \theta)$.*

Combining the above definition with Observation 3.2, we note that for any $\alpha \in \Delta^k$ and $\theta \in \mathcal{P}^k$,

$$f'_S(\alpha'(\theta), \theta) + \phi(\alpha, \theta) \in (1 \pm \varepsilon) \cdot f'(\alpha, \theta) + \phi(\alpha, \theta) \in f(\alpha, \theta) \pm \varepsilon f'(\alpha, \theta). \tag{2}$$

As ε tends to 0, $f'_S(\alpha'(\theta), \theta) + \phi(\alpha, \theta)$ converges to $f(\alpha, \theta)$. Then, given such a coreset S, we can minimize $f'_S(\alpha'(\theta), \theta) + \phi(\alpha, \theta)$ to solve Problem 3.1. Note that we introduce multiple weight functions (w for entities and $w^{(i)}$ for time periods of selected entity i) in Definition 3.3 unlike the coreset for time series regression in [34] which uses only one weight function. This is because f'_i contains $\ln$ and $\exp$ operators instead of linear combinations of ψ_{it}; it is unclear how to use a single weight function to capture both the entity and time levels.

4 Theoretical results

We present our coreset algorithm (Algorithm 1) and the main result on its performance (Theorem 4.3). Theorem 4.3 needs the following assumptions on covariance and autocorrelation parameters.

Assumption 4.1. *Assume (1) that there exists constant $D \geq 1$ such that $\frac{\max_{l\in[k]} \lambda_{\max}(\Sigma^{(l)})}{\min_{l\in[k]} \lambda_{\min}(\Sigma^{(l)})} \leq D$ where $\lambda_{\max}(\cdot)$ represents the largest eigenvalue and $\lambda_{\min}(\cdot)$ represents the smallest eigenvalue, and (2) for $l \in [k]$, $\Lambda^{(l)} \in \mathcal{D}_\lambda^d$ for some constant $\lambda \in (0,1)$. Here $\mathcal{D}_\lambda^d$ is a collection of all diagonal matrix in $\mathbb{R}^{d\times d}$ whose diagonal elements are at most $1-\sqrt{\lambda}$.*

The first assumption requires that the condition number of each covariance matrix is upper-bounded, which also appears in [55, 42, 45] that consider Gaussian mixture models with static data. The second assumption, roughly, requires that there exist autocorrelations only between the same features. The upper bound for diagonal elements ensures that the autocorrelation degrades as time period increases, which is also assumed in [34]. Let $\mathcal{P}_\lambda^k := (\mathbb{R}^d \times \mathcal{S}^d \times \mathcal{D}_\lambda^d)^k$ denote the parameter space. For preparation, we propose the following k-means clustering problem.

Definition 4.2 (k**-means clustering of entities).** *Given an integer $k \geq 1$, the goal of the k-means clustering problem of entities of X is to find a set $C^\star = \{c_1^\star, \ldots, c_k^\star\} \subset \mathbb{R}^d$ of k centers that minimizes $\sum_{i\in[N]} \min_{l\in[k]} \|\frac{\sum_{t\in[T_i]} x_{it}}{T_i} - c_l\|_2^2$ over all k center sets $C = \{c_1, \ldots, c_k\} \subset \mathbb{R}^d$. Let $\mathsf{OPT}^{(O)}$ denote the optimal k-means value of $C^\star$.*

Note that there exists an $O(2^{\mathsf{poly}(k)}Nd)$ time algorithm to compute a nearly optimal solution for both $\mathsf{OPT}^{(O)}$ and $C^\star$ [40]. Another widely used algorithm is called k-means++ [6], which provides an $O(\ln k)$-approximation in $O(Ndk \ln N \ln k)$ time but performs well in practice. We also observe that $\psi_i(\mu, I_d, 0_d) = \|\frac{\sum_{t\in[T_i]} x_{it}}{T_i} - \mu\|_2^2 + \frac{1}{T_i}\sum_{t\in[T_i]} \|x_{it}\|_2^2 - \frac{\|\sum_{t\in[T_i]} x_{it}\|_2^2}{T_i^2}$ for any $\mu \in \mathbb{R}^d$. This observation motivates us to consider a reduction from Problem 3.1 to Definition 4.2, which is useful for constructing I_S.

Our coreset algorithm. We first give a summary of our algorithm (Algorithm 1). The input to Algorithm 1 is a time series dataset $\mathcal{X}$, an error parameter of coreset $\varepsilon \in (0,1)$, an integer $k \geq 1$ of clusters, $\lambda \in (0,1)$ (vector norm bound), and a covariance eigenvalue gap $D \geq 1$. We develop a two-staged importance sampling framework which first samples a subset I_S of entities (Lines 1-8) and then samples a subset of time periods $J_{S,i}$ for each selected entity $i \in I_S$ (Lines 9-14).

In the first stage, we first set M as the number of selected entities $|I_S|$ (Line 1). Then we solve the k-means clustering problem over entity means b_i (Definition 4.2), e.g., by k-means++ [6] (Lines 2-3), and obtain an (approximate) optimal clustering value $\mathsf{OPT}^{(O)}$, a k-center set $C^\star := \{c_1^\star, \ldots, c_k^\star\} \subset \mathbb{R}^d$, and an offset value a_i for each $i \in [N]$. Next, based on the distances of points b_i to $C^\star$, we partition $[N]$ into k clusters where entity i belongs to cluster $c_{p(i)}^\star$ (Line 4). Based on b_i, $c_{p(i)}^\star$ and $\mathsf{OPT}^{(O)}$, we compute an upper bound $s(i)$ for the sensitivity w.r.t. entity $i \in [N]$ (Lines 5-6). Finally, we sample a weighted subset I_S of entities as our coreset for the entity level by importance sampling based on $s(i)$ (Lines 7-8), which follows from the Feldman-Langberg framework (Theorem A.6).

In the second stage, we first set L as the number of time periods $|J_{S,i}|$ for each selected entity $i \in I_S$ (Line 9). Then for each selected entity $i \in I_S$, we compute $\mathsf{OPT}_i^{(O)}$ as the optimal 1-means clustering objective of x_{it}s (Line 10). Next, we compute an upper bound $s_i(t)$ for the sensitivity w.r.t. time period $t \in [T_i]$ (Lines 11-12) based on b_i and $\mathsf{OPT}_i^{(O)}$. Finally, we sample a weighted subset $J_{S,i}$ of time periods by importance sampling based on $s_i(t)$ (Lines 13-14).

Our main theorem. Our main theorem is as follows, which indicates that Algorithm 1 provides an efficient coreset construction algorithm for GMM time series clustering.

Theorem 4.3 (**Main result).** *Under Assumption 4.1, with probability at least 0.9, Algorithm 1 outputs an ε-coreset of size $O\left(\frac{D^2k^5d^{16}\ln^2\frac{k}{\lambda}}{\lambda^2\varepsilon^4}\right)$ for GMM time-series clustering in $O(d\sum_{i\in[N]} T_i + Ndk\ln N \ln k)$ time.*

[1]Here, we slightly abuse the notation by using $\mathsf{OPT}^{(O)}$ and $C^\star$ to represent the obtained approximate solution.

Algorithm 1: CRGMM: Coreset construction for GMM time series clustering

Input: $\mathcal{X} = \left\{X_i = (x_{i1}, \ldots, x_{i,T_i}) \subset \mathbb{R}^{T_i \times d} \mid i \in [N]\right\}$, constant $\varepsilon, \lambda \in (0,1)$, number of clusters $k \geq 1$, dimension of autocorrelation vectors $q \geq 0$, constant of the variance gap $D \geq 1$, and parameter space $\mathcal{P}_\lambda^k = (\mathbb{R}^d \times \mathcal{S}_\tau^d \times \mathcal{D}_\lambda^d)^k$.

Output: a subset $S \subseteq P$ together with weight functions $w : I_S \to \mathbb{R}_{\geq 0}$ and $w^{(i)} : J_{S,i} \to \mathbb{R}_{\geq 0}$ for $i \in I_S$.

% Constructing a subset of entities

$M \leftarrow O\left(\frac{kD}{\lambda\varepsilon^2}(k^4d^4 + k^3d^8)\ln\frac{k}{\lambda}\right)$.

For each $i \in [N]$, compute point $b_i \in \mathbb{R}^d$ by $b_i \leftarrow \frac{\sum_{t\in[T_i]} x_{it}}{T_i}$ and value $a_i \geq 0$ by $a_i \leftarrow \frac{1}{T_i}\sum_{t\in[T_i]} \|x_{it}\|_2^2 - \frac{\|\sum_{t\in[T_i]} x_{it}\|_2^2}{T_i^2}$. Let $A \leftarrow \sum_{i\in[N]} a_i$.

Compute (an $O(1)$-approximate for) $\mathsf{OPT}^{(O)}$ and $C^\star = \{c_1^\star, \ldots, c_k^\star\} \subset \mathbb{R}^d$ for the k-means clustering problem over points b_i (Definition 4.2),[1] by [40, 6].

For each $i \in [N]$, compute index $p(i) \in [k]$ by $p(i) \leftarrow \arg\min_{l\in[k]} \|b_i - c_l^\star\|_2^2$.

For each $i \in [N]$, $s^c(i) \leftarrow \frac{1}{|\left\{i' \in [N] : p(i') = c_{p(i)}^\star\right\}|}$.

For each $i \in [N]$, $s(i) \leftarrow \min\left\{1, 4D\left(\frac{4\|b_i - c_{p(i)}^\star\|_2^2}{\mathsf{OPT}^{(O)} + A} + 3s^c(i)\right)/\lambda\right\}$.

Pick a random sample $I_S \subseteq [N]$ of size M, where each $i \in I_S$ is selected w.p. $\frac{s(i)}{\sum_{i'\in[N]} s(i')}$

For each $i \in I_S$, $w(i) \leftarrow \frac{\sum_{i'\in[N]} s(i')}{M \cdot s(i)}$.

% Constructing a subset of time periods for each selected entity

$L \leftarrow O\left(\frac{Dd^8 \ln\frac{1}{\lambda}}{\lambda\varepsilon^2}\right)$.

For each $i \in I_S$, compute $\mathsf{OPT}_i^{(O)} \leftarrow \sum_{t\in[T_i]} \|x_{it} - b_i\|_2^2$.

For each $i \in I_S$ and $t \in [T_i]$, $s_i^c(t) \leftarrow \frac{2\|x_{it} - b_i\|_2^2}{\mathsf{OPT}_i^{(O)}} + \frac{6}{T_i}$.

For each $i \in I_S$ and $t \in [T_i]$, $s_i(t) \leftarrow \min\left\{1, 4D\lambda^{-1}\left(s_i^c(t) + \sum_{j=1}^{\min\{t-1,1\}} s_i^c(t-j)\right)\right\}$.

For each $i \in I_S$, pick a random sample $J_{S,i}$ of L points, where each $t \in J_{S,i}$ is selected w.p. $\frac{s_i(t)}{\sum_{t'\in[T_i]} s_i(t')}$.

For each $i \in I_S$ and $t \in J_{S,i}$, $w^{(i)}(t) \leftarrow \frac{\sum_{t'\in[T_i]} s_i(t')}{L \cdot s_i(t)}$.

% Output coreset S of entity-time pairs

Let $S \leftarrow \{(i,t) \in P_{\mathcal{X}} : i \in I_S, t \in J_{S,i}\}$.

Output $\left(S, w, \left\{w^{(i)}\right\}_{i\in I_S}\right)$.

Note that the success probability in the theorem can be made $1 - \delta$ for any $\delta \in (0,1)$ at the expense of an additional factor $\log^2 1/\delta$ in the coreset size. The coreset size guaranteed by Theorem 4.3 has a polynomial dependence on factors $k, d, 1/\varepsilon, D$, and $1/\lambda$ and, importantly, does not depend on N or T_is. Compared to the clustering problem with GMM static data [45], the coreset has an additional dependence on $1/\lambda$ due to autocorrelation parameters. Compared to the regression problem (GLSE$_k$) with time series data [34, Theorem 5.2], the coreset does not contain the factor "M" that upper bounds the gap between the maximum and the minimum entity objective. The construction time linearly depends on the total number of observations $\sum_{i\in[N]} T_i$, which is efficient. The proof of Theorem 4.3 can be found in Section C.

Proof overview of Theorem 4.3. The proof consists of three parts: 1) Bounding the size of the coreset, 2) proving the approximation guarantee of the coreset, and 3) bounding the running time.

Parts 1 & 2: Coreset size and correctness guarantee of Theorem 4.3. We first note that the coreset size in Theorem 4.3 is $|S| = ML$. Here, M is the number of sampled entities $|I_S|$ that guarantee that I_S is a coreset for f' at the entity level (Lemma 4.4). L is the number of sampled observations $|J_{S,i}|$ for each $i \in I_S$ that guarantees that $J_{S,i}$ is a coreset for ψ_i at the time level (Lemma 4.5).

Lemma 4.4 (**The 1st stage outputs an entity-level coreset).** *With probability at least* 0.95*, the output* I_S *of Algorithm 1 with* $M := |I_S| = O\left(\frac{kD}{\lambda\varepsilon^2}(k^4d^4 + k^3d^8)\ln\frac{k}{\lambda}\right)$ *satisfies that for any* $\alpha \in \Delta_k$ *and* $\theta \in \mathcal{P}_\lambda^k$, $-\sum_{i\in I_S} w(i) \cdot \ln\sum_{l\in[k]} \alpha_l \cdot \exp(-\frac{1}{2T_i}\psi_i(\mu^{(l)}, \Sigma^{(l)}, \Lambda^{(l)})) \in (1 \pm \varepsilon) \cdot f'(\alpha, \theta)$.

Lemma 4.5 (**The 2nd stage outputs time level coresets for each** $i \in I_S$)**.** *With probability at least* 0.95*, for all* $i \in I_S$*, the output* $J_{S,i}$ *of Algorithm 1 with* $L := |J_{S,i}| = O\left(\frac{Dd^8 \ln \frac{1}{\lambda}}{\lambda \varepsilon^2}\right)$ *satisfies that for any* $(\mu, \Sigma, \Lambda) \in \mathcal{P}_\lambda$*,* $\sum_{t \in J_{S,i}} w^{(i)}(t) \cdot \psi_{it}(\mu, \Sigma, \Lambda) \in (1 \pm \varepsilon) \cdot \psi_i(\mu, \Sigma, \Lambda)$.

From Lemmas 4.4 and 4.5, we can conclude that S is an $O(\varepsilon)$-coreset for GMM time-series clustering, since the errors ε for the entity level and the time level are additively accumulated, and that the size of the coreset ML as claimed; see Section C for a complete proof.

Both lemmas rely on the Feldman-Langberg framework [26, 13] (Theorem A.6). The key is to upper bound both the "pseudo-dimension" $\dim$ (Definition A.5) that measures the complexity of the parameter space, and the total sensitivity $\mathcal{G}$ (Definition A.4) that measures the sum of maximum influences of all sub-functions (f'_i for Lemma 4.4 and ψ_{it} for Lemma 4.5). The coreset size guaranteed by the Feldman-Langberg framework then is $O(\varepsilon^{-2}\mathcal{G} \dim \ln \mathcal{G})$.

For Lemma 4.5, we can upper bound the pseudo-dimension by $\dim = O(d^8)$ (Lemma B.1). For the total sensitivity, a key step is to verify that s_i (Line 12) is a sensitivity function of ψ_i. This step uses a similar idea as in [34] and leads to showing a $\mathcal{G} = O(D/\lambda)$ bound for the total sensitivity (Lemma B.2). Then we can verify that $L = O(\varepsilon^{-2}\mathcal{G} \dim \ln \mathcal{G}) = O\left(\frac{Dd^8 \ln \frac{1}{\lambda}}{\lambda \varepsilon^2}\right)$ is enough for Lemma 4.5 by the Feldman-Langberg framework (Theorem A.6). The proof can be found in Section B.

Lemma 4.4 is the most technical lemma. The proof can be found in Section A and we provide a proof sketch in the following.

Proof sketch of Lemma 4.4. By [4, 53], the pseudo-dimension is determined by the number of parameters that is upper bounded by $O(kd^2)$ and the number of operations on parameters that is upper bounded by $O(kd^6)$ including $O(k)$ exponential functions. Then we can upper bound the pseudo-dimension by $\dim = O(k^4d^4 + k^3d^8)$ (Lemma A.9 in Section A.2).

For the total sensitivity, the key is to verify that s (Line 6) is a sensitivity function w.r.t. f'; this is established by Lemma 4.6 below. Then the total sensitivity is at most $\mathcal{G} = (16D + 12Dk)/\lambda$. This completes the proof of Lemma 4.4 since $M = O(\varepsilon^{-2}\mathcal{G} \dim \ln \mathcal{G}) = O\left(\frac{kD}{\lambda\varepsilon^2}(k^4d^4 + k^3d^8) \ln \frac{k}{\lambda}\right)$ is enough by the Feldman-Langberg framework (Theorem A.6).

Lemma 4.6 (s **is a sensitivity function w.r.t.** f')**.** *For each* $i \in [N]$*, we have* $s(i) \geq \max_{\alpha \in \Delta^k, \theta \in \mathcal{P}^k_\lambda} \frac{f'_i(\alpha,\theta)}{f'(\alpha,\theta)}$*. Moreover,* $\sum_{i \in [N]} s(i) \leq (16D + 12Dk)/\lambda$.

Proof sketch of Lemma 4.6. We first define $\psi_i^{(O)}(\mu) := \sum_{t \in [T_i]} \|x_{it} - \mu\|_2^2$ for any $\mu \in \mathbb{R}^d$, and define $f_i^{(O)}(\alpha, \theta^{(O)}) = -\ln \sum_{l \in [k]} \alpha_l \cdot \exp\left(-\frac{1}{2T_i \cdot \min_{l' \in [k]} \lambda_{\min}(\Sigma^{(l')})} \psi_i^{(O)}(\mu^{(l)})\right)$. Based on $f_i^{(O)}$, we define another sensitivity function $s^{(O)}(i) := \max_{\alpha \in \Delta^k, \theta^{(O)} \in \mathbb{R}^{d \times k}} \frac{f_i^{(O)}(\alpha,\theta^{(O)})}{f^{(O)}(\alpha,\theta^{(O)})}$ and want to reduce s to $s_i^{(O)}$. Note that $\psi_i^{(O)}(\mu) = \psi_i(\mu, I_d, 0_d)$, i.e., $\psi_i^{(O)}$ removes the covariance and autocorrelation parameters from ψ_i. Due to this observation, we have that $s^{(O)}(i) \leq \max_{\alpha \in \Delta^k, \theta \in \mathcal{P}^k_\lambda} \frac{f'_i(\alpha,\theta)}{f'(\alpha,\theta)} \leq 4D \cdot s^{(O)}(i)/\lambda$ by upper bounding the affect of covariance parameters (controlled by factor D) and autocorrelation parameters (controlled by factor $1/\lambda$), summarized as Lemma A.11. Then to prove Lemma 4.6, it suffices to prove that $s^{(O)}(i) \leq \frac{4\|b_i - c^\star_{p(i)}\|_2^2}{\mathsf{OPT}^{(O)} + A} + 3s^c(i)$ which implies that the total sensitivity of $s^{(O)}$ is upper bounded by $4 + 3k$ (Lemma A.12), due to fact that $\sum_{i \in [N]} s^c(i) \leq k$ (Lemma A.13). Finally, the proof of Lemma A.12 is based on a reduction from $\psi^{(O)}$ to the k-means clustering problem of entities (Definition 4.2) by rewriting $\frac{1}{T_i}\psi_i^{(O)}(\mu) = \|b_i - \mu\|_2^2 + a_i$ and then projecting each b_i to its closest center in $C^\star$. The full proof of this lemma can be found in Section A.3.

Part 3: Running time in Theorem 4.3. The first stage costs $O(d \sum_{i \in [N]} T_i + Ndk \ln N \ln k)$ time. The dominating steps are 1) to compute b_i in Line 2 which costs $O(d \sum_{i \in [N]} T_i)$ time; 2) to solve the k-means clustering problem of entities (Line 3) which costs $Ndk \ln N \ln k)$ time. The second stage costs at most $O(d \sum_{i \in [N]} T_i)$ time. The dominating step is to compute $\mathsf{OPT}_i^{(O)}$ for all $i \in I_S$ in Line 10, where it costs $O(dT_i)$ time to compute each $\mathsf{OPT}_i^{(O)}$.

5 Empirical results

We compare the performance of our coreset algorithm CRGMM, with uniform sampling as a benchmark using synthetic data. The experiments are conducted with PyCharm IDE on a computer with 8-core CPU and 32 GB RAM.

Datasets. We generate two sets of **synthetic data** all with 250K observations with different number of entities N and observations per individual T_i: (i) $N = T_i = 500$ for all $i \in [N]$ and (ii) $N = 200$, $T_i = 1250$ for all $i \in [N]$. As cross-entity variance is greater than within entity variance, this helps to assess coreset performance sensitivity when there are more entities (high N) vs. more frequent measurement/ longer measurement period (high T_i).

We fix $d = 2$, $k = 3$, $\lambda = 0.01$ and generate multiple datasets with model parameters randomly varied as follows: (i) Draw α from a uniform distribution over Δ_k. (ii) For each $l \in [k]$, draw $\mu^{(l)} \in \mathbb{R}^d$ from $N(0, I_d)$. (iii) Draw $\Sigma^{(l)}$ by first generating a random matrix $A \in [0, 1]^{d\times d}$ and then let $\Sigma^{(l)} = (AA^\top)^{-1}$, and (iv) draw all diagonal elements of $\Lambda^{(l)} \in \mathcal{D}^d$ from a uniform distribution over [0, $1 - \sqrt{\lambda}$]. Given these draws of parameters, we generate a GMM time-series dataset as follows: For each $i \in [N]$, draw $l \in [k]$ given α. Then, for all $t \in [T_i]$ draw $e_{it} \in \mathbb{R}^d$ with covariance matrix $\Sigma^{(l)}$ and autocorrelation matrix $\Lambda^{(l)}$ and compute $x_{it} = \mu^{(l)} + e_{it} \in \mathbb{R}^d$.

Baseline and metrics. We use a coreset based on uniform sampling (**Uni**) as the baseline. Given an integer Γ, uniformly sample a collection S of Γ entity-time pairs $(i, t) \in P_{\mathcal{X}}$, let $w(i) = \frac{N}{|I_S|}$ for each $i \in I_S$, and let $w^{(i)}(t) = \frac{T_i}{|J_{S,i}|}$ for each $t \in J_{S,i}$. For comparability, we set Γ to be the same as our coreset size.

Let $V^\star$ (negative log-likelihood) denote the objective of Problem 3.1 with full data. Given a weighted subset $S \subseteq [N] \times [T]$, we first compute $(\alpha^\star, \theta^\star) := \arg\min_{\alpha\in\Delta_k,\theta\in\mathcal{P}_\lambda^k} f'_S(\alpha'(\theta), \theta) + \phi(\alpha, \theta)$ as the maximum likelihood solution over S.[2] Then $V_S^\star = f(\alpha^\star, \theta^\star)$ over the full dataset serves as a metric of model fit given model estimates from a weighted subset S. We use the likelihood ratio, i.e., $\gamma_S := 2(V_S^\star - V^\star)$ as a measure of the quality of S [14]. The running time for GMM clustering with full data ($V^\star$) and coreset ($V_S^\star$) are $T_{\mathcal{X}}$ and T_S respectively. T_C is the coreset S construction time.

Empirical setup. We vary $\varepsilon = 0.1, 0.2, 0.3, 0.4, 0.5$. For each ε, we run CRGMM and **Uni** to generate 5 coresets each. We estimate the model with the full dataset $\mathcal{X}$ and the coresets S and record $V^\star$, $V_S^\star$ and the run times.

Table 1: Performance of ε-coresets for CRGMM w.r.t. varying ε. We report model fit as the mean of $V_S^\star$ (negative log-likelihood) w.r.t. 5 repetitions for our algorithm CRGMM and **Uni** on two synthetic datasets. "Synthetic 1" ($N = 500$ entities and $T_i = 500$ observations for each entity), and "Synthetic 2" ($N = 200$, $T_i = 1250$); Size (Γ): number of sampled entity-time pairs for both CRGMM and **Uni**; T_C: construction time of coresets. T_S and $T_{\mathcal{X}}$: computation time for GMM clustering time over coresets and full data respectively. Model fit $V_S^\star$ of our coreset is very close relative to the optimal $V^\star$ with full data; and much better than **Uni**. Our coreset fit also does not decay as much as **Uni** with larger error guarantees (0.1 to 0.5). Our coreset achieves a 2x-8x acceleration in solving Problem 3.1 compared to **Uni** using same sample size and a 17x-92x acceleration relative to the full data.

	ε	$V_S^\star$		$V^\star$	γ_S		size	T_C (s)		$T_C + T_S$ (s)		$T_{\mathcal{X}}$ (s)
		CRGMM	**Uni**		CRGMM	**Uni**		CRGMM	**Uni**	CRGMM	**Uni**	
Synthetic 1	0.1	1787	1935	1780	16	311	1668	1.27	0.17	38	309	1448
	0.2	1795	2312		31	1066	420	1.22	0.08	24	128	
	0.3	1843	2655		127	1750	196	1.18	0.06	20	79	
	0.4	1914	3041		269	2523	98	1.18	0.06	17	53	
	0.5	1951	2922		342	2285	72	1.17	0.05	16	40	
Synthetic 2	0.1	1081	1100	1072	18	28	1719	1.45	0.1	85	444	1285
	0.2	1088	1230		33	316	435	1.29	0.06	43	297	
	0.3	1098	1295		52	447	195	1.24	0.05	34	167	
	0.4	1108	1631		73	1119	97	1.21	0.05	30	114	
	0.5	1154	1687		165	1230	72	1.2	0.05	26	84	

Results. Table 1 summarizes the quality-size-time trade-offs of our coresets for different error guarantees ε. The $V_S^\star$ achieved by our coreset is always close to the optimal value $V^\star$ and is always

[2] We solve this optimization problem using an EM algorithm similar to [5]. The M step in each iteration is based on IRLS [37] and the E-step involves an individual level Bayesian update for α.

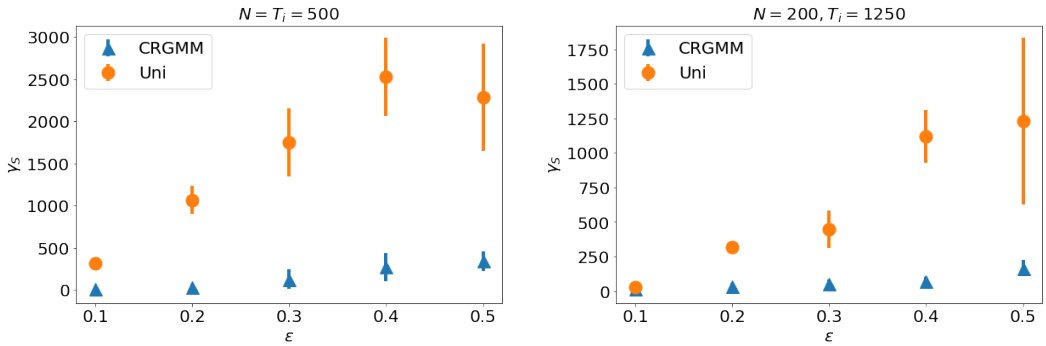

Figure 1: Mean +/- std. of γ_S, w.r.t. varying ε. γ_S is the likelihood ratio (LR) for estimates from coresets relative to full data. CRGMM not only has better fit than **Uni**, but much small std. in performance. The std. for our coreset is even smaller for Synthetic 2 with fewer entities.

smaller than that of **Uni**. We next assess coreset size. For the $N = T_i = 500$ case, the coreset achieves a likelihood ratio $\gamma_S = 269$ with only 98 entity-time pairs (0.04%), but **Uni** achieves the closest (but worse) likelihood ratio of $\gamma_S = 311$, with 1668 entity-time pairs (0.7%). Thus, our coreset achieves a better likelihood than uniform with less than 6% of observations. Further, from Figure 1, we see that not only is the quality of our coreset (γ_S) superior,it has lower standard deviations suggesting lower variance in performance. In terms of total computation time relative to the full data ($\frac{T_{\mathcal{X}}}{T_C+T_S}$), our coresets speed up by 17x-92x. Also, the computation time ($T_C + T_S$) of **Uni** is always larger than that of our coreset (2x-8x).[3] Finally, $V_S^\star$ is lower for Synthetic 2 given that there are fewer entities and cross-entity variance is greater than within entity variance. From Figure 1, we also see that both the fit and std in performance for coresets is much lower for Synthetic 2. Thus, overall our coreset performs better when there is more time series data relative to entities ($T_i \gg N$)—a feature of sensor based measurement that motivates this paper.

6 Limitations, conclusion, and future work

In this paper, we study the problem of constructing coresets for clustering problems with time series data; in particular, we address the problem of constructing coresets for time series data generated from Gaussian mixture models with auto-correlations across time. Our coreset construction algorithm is efficient under a mild boundedness assumption on the covariance matrices of the underlying Gaussians, and the size of the coreset is independent of the number of entities and the number of observations and depends only polynomially on the number of clusters, the number of variables and an error parameter. Through empirical analysis on synthetic data, we demonstrate that the coreset sampling is superior to uniform sampling in computation time and accuracy.

Our work leaves several interesting directions for future work on time series clustering. While our current specification with autocorrelations assumes a stable time series pattern over time, future work should extend it to a hidden Markov process, where the time series process may switch over time. Further, while our focus here is on model-based clustering, it would be useful to consider how coresets should be constructed for other clustering methods such as direct clustering of raw data and indirect clustering of features.

Overall, we hope the paper stimulates more research on coreset construction for time series data on a variety of unsupervised and supervised machine learning algorithms. The savings in storage and computational cost without sacrificing accuracy is not only financially valuable, but also can have sustainability benefits through reduced energy consumption. Finally, recent research has shown that summaries (such as coresets) for static data need to include fairness constraints to avoid biased outcomes for under-represented groups based on gender and race when using the summary [16, 33]; future research needs to extend such techniques for time series coresets.

[3] A possible explanation is that our coreset selects I_S of entities that are more representative that **Uni**, and hence, achieves a better convergence speed.

References

[1] Pankaj K Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[3] Hirotugu Akaike and Genshiro Kitagawa. *The practice of time series analysis*. Springer Science & Business Media, 2012.

[4] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University press, 2009.

[5] Peter Arcidiacono and John Bailey Jones. Finite mixture distributions, sequential likelihood and the EM algorithm. *Econometrica*, 71(3):933–946, 2003.

[6] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007.

[7] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

[8] Badi Baltagi. *Econometric analysis of panel data*. John Wiley & Sons, 2008.

[9] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, 140:112896, 2020.

[10] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-means: beyond subspaces and the Johnson-Lindenstrauss lemma. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1039–1050. ACM, 2019.

[11] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020.

[12] Hans-Hermann Bock. Clustering methods: a history of k-means algorithms. *Selected contributions in data analysis and classification*, pages 161–172, 2007.

[13] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

[14] Adolf Buse. The likelihood ratio, wald, and lagrange multiplier tests: An expository note. *The American Statistician*, 36(3a):153–157, 1982.

[15] Jorge Caiado, Elizabeth A Maharaj, and Pierpaolo D'Urso. Time series clustering. *Handbook of cluster analysis*, pages 241–263, 2015.

[16] L Elisa Celis, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth K Vishnoi. Fair and diverse dpp-based data summarization. In *ICML*, 2018.

[17] Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, August 2009.

[18] Mike X Cohen. *Analyzing neural time series data: theory and practice*. MIT press, 2014.

[19] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. *CoRR*, abs/2104.06133, 2021.

[20] Pierpaolo D'Urso and Elizabeth Ann Maharaj. Autocorrelation-based fuzzy clustering of time series. *Fuzzy Sets and Systems*, 160(24):3565–3589, 2009.

[21] Jason Ernst, Gerard J Nau, and Ziv Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(suppl_1):i159–i168, 2005.

[22] Dan Feldman. Core-sets: Updated survey. *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 23–44, 2020.

[23] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2142–2150, 2011.

[24] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 361–370, 2009.

[25] Dan Feldman, Zahi Kfir, and Xuan Wu. Coresets for Gaussian mixture models of any shape. *arXiv preprint arXiv:1906.04895*, 2019.

[26] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011. `https://arxiv.org/abs/1106.1379`.

[27] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. SIAM, 2013.

[28] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[29] John N Haddad. A simple method for computing the covariance matrix and its inverse of a stationary autoregressive process. *Communications in Statistics-Simulation and Computation*, 27(3):617–623, 1998.

[30] Sariel Har-Peled. Clustering motion. *Discrete & Computational Geometry*, 31(4):545–565, 2004.

[31] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *36th Annual ACM Symposium on Theory of Computing,*, pages 291–300, 2004.

[32] Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci. *Handbook of cluster analysis*. CRC Press, 2015.

[33] Lingxiao Huang, Shaofeng H-C Jiang, and Nisheeth K Vishnoi. Coresets for clustering with fairness constraints. In *NeurIPS*, 2019.

[34] Lingxiao Huang, K Sudhir, and Nisheeth Vishnoi. Coresets for regressions with panel data. *Advances in Neural Information Processing Systems*, 33, 2020.

[35] Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in Euclidean spaces: Importance sampling is nearly optimal. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1416–1429. ACM, 2020.

[36] Jui-Long Hung, Morgan C Wang, Shuyan Wang, Maha Abdelrasoul, Yaohang Li, and Wu He. Identifying at-risk students for early interventions—a time-series clustering approach. *IEEE Transactions on Emerging Topics in Computing*, 5(1):45–55, 2015.

[37] Murray Jorgensen. Iteratively reweighted least squares. *Encyclopedia of Environmetrics*, 3, 2006.

[38] T Tony Ke and K Sudhir. Privacy rights and data security: GDPR and personal data driven markets. *Available at SSRN 3643979*, 2020.

[39] Eamonn Keogh. A decade of progress in indexing and mining large time series databases. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1268–1268, 2006.

[40] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 454–462. IEEE Computer Society, 2004.

[41] James P LeSage. The theory and practice of spatial econometrics. *University of Toledo. Toledo, Ohio*, 28(11), 1999.

[42] Jiao-fen Li, Wen Li, and Seak-Weng Vong. Efficient algorithms for solving condition number-constrained matrix minimization problems. *Linear Algebra and its Applications*, accepted, 08 2020.

[43] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, 2021.

[44] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[45] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training Gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[46] Benjamin M Marlin, David C Kale, Robinder G Khemani, and Randall C Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT international health informatics symposium*, pages 389–398, 2012.

[47] Ming Ouyang, William J Welsh, and Panos Georgopoulos. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20(6):917–923, 2004.

[48] M Hashem Pesaran. *Time series and panel data econometrics*. Oxford University Press, 2015.

[49] Aarthi Reddy, Meredith Ordway-West, Melissa Lee, Matt Dugan, Joshua Whitney, Ronen Kahana, Brad Ford, Johan Muedsam, Austin Henslee, and Max Rao. Using gaussian mixture models to detect outliers in seasonal univariate network traffic. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 229–234. IEEE, 2017.

[50] Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *FOCS*, pages 802–813. IEEE Computer Society, 2018.

[51] VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264, 1971.

[52] Kasturi Varadarajan and Xin Xiao. On the sensitivity of shape fitting problems. In *32nd International Conference on Foundations of Software Technology and Theoretical Computer Science*, page 486, 2012.

[53] Mathukumalli Vidyasagar. *A theory of learning and generalization*. Springer-Verlag, 2002.

[54] William WS Wei. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*. 2006.

[55] Joong-Ho Won, Johan Lim, Seung-Jean Kim, and Bala Rajaratnam. Condition-number-regularized covariance estimation. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 75(3):427–450, 2013.

[56] Yimin Xiong and Dit-Yan Yeung. Time series clustering with ARMA mixtures. *Pattern Recognition*, 37(8):1675–1689, 2004.

[57] Yun Yang and Jianmin Jiang. Hmm-based hybrid meta-clustering ensemble for temporal data. *Knowledge-Based Systems*, 56:299–310, 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3, 4
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Section 5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]