LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-870933

Evaluation of Traditional and Deep Clustering Algorithms for Multivariate Spatio-Temporal Data

F. N. Nji, R. M. Salvi, S. Tirumala, J. Wang, X. Zheng

October 28, 2024

Seventh IEEE International Workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD 2024)
Washington DC, DC, United States
December 15, 2024 through December 18, 2024

Disclaimer

Evaluation of Traditional and Deep Clustering Algorithms for Multivariate Spatiotemporal Data

Francis Ndikum Nji[1], Rohan Mandar Salvi[1], Sai Sri Ram Kuram Tirumala[1], Jianwu Wang[1],
Xue Zheng[2]
[1]Department of Information Systems, University of Maryland, Baltimore County, Baltimore, MD, United States
Email: {fnji1,rsalvi2,saisrik1,jianwu}@umbc.edu
[2]Climate Sciences Group, Lawrence Livermore National Laboratory, Livermore, CA, United States
Email:zheng7@llnl.gov

***Abstract*—Spatiotemporal data is commonly available in many disciplines such as atmospheric science, Earth sciences and environment science, and data is generated by monitoring a certain area over a period of time. Analyzing such high-dimensional data is critical for uncovering hidden patterns and one important approach is to categorize it along the temporal dimension into smaller groups. While classical methods like K-means and Gaussian Mixture Models (GMM) are favored for their simplicity and interpretability, they encounter challenges in modeling complex, high-dimensional relationships inherent in nonlinear spatiotemporal data. In contrast, deep clustering algorithms that combine neural networks with unsupervised learning objectives excel by learning latent representations that better capture nonlinear spatiotemporal dependencies. This study provides a rigorous evaluation of both traditional and deep clustering algorithms on high dimensional multivariate spatiotemporal climate datasets. Our comparative study examines the performance of these techniques across synthetic and real-world datasets, assessing clustering accuracy and stability. We emphasize the advantages of deep clustering, particularly in applications such as climate data analysis and traffic flow prediction, where mining and understanding nonlinear high-dimensional correlations are critical. The results demonstrate that while traditional clustering algorithms are effective for basic tasks, deep learning-based approaches outperform them in managing complex nonlinear patterns present in high dimensional multivariate spatiotemporal data.**

***Index Terms*—Multivariate spatiotemporal data, traditional clustering, deep unsupervised clustering, performance evaluation, stability evaluation**

I. Introduction

Aerosol-cloud interactions are critical to understanding Earth's climate system, as they influence cloud formation, cloud properties and the planetary energy balance. Aerosols are small particles in the atmosphere which often act as cloud condensation nuclei and alter cloud albedo and cloud lifetime. These interactions are complex and dynamic, involving both direct effects; where aerosols scatter or absorb sunlight, and indirect effects; where aerosols modify cloud properties such as droplet size, which can influence cloud reflectivity and precipitation patterns. Aerosol-cloud interactions and their processes play a significant role in radiative forcing and global temperature regulation hence observing and modeling these interactions is essential for accurately predicting climate changes. In climate models, aerosol-cloud interactions are represented through various physical and chemical processes. However, significant uncertainties remain in quantifying these effects due to the inherent complexity of atmospheric processes and the influence of environmental confounding factors, such as temperature, humidity, and wind patterns. Observational signals of aerosol effects, such as changes in cloud droplet size or cloud optical depth, must be carefully interpreted by isolating the aerosol signal from these environmental factors. This makes it challenging to fully understand aerosol-induced changes and calls for *more sophisticated methods to disentangle aerosol effects from confounding variables.* Recent advancements in observational techniques and climate models have improved our understanding of aerosol-cloud interactions. Satellite data, ground-based measurements, and advanced climate models now provide multidimensional datasets that capture a wide array of variables, including aerosol concentrations, cloud properties, atmospheric conditions, and temporal dynamics. However, the sheer volume and complexity of these datasets make it difficult to extract meaningful insights without advanced data analysis techniques. One promising approach is the use of multi-dimensional data clustering, which allows researchers to group data points based on similarities across multiple variables such as aerosol concentration, cloud type, and environmental conditions, while taking into account their spatial and temporal distribution. This approach can help isolate the true effects of aerosols on clouds by identifying patterns that would otherwise be obscured by the high-dimensional nature of the data. The need for sophisticated clustering techniques is particularly pressing in the context of aerosol-cloud interactions, as these processes unfold across both space and time, creating datasets

This work is supported by the DOE Office of Science Early Career Research Program. This work was performed under the auspices of the U.S. Department of Energy (DOE) by LLNL under contract DE-AC52-07NA27344. LLNL-CONF-870933.

that are inherently four-dimensional (4D). Their data structures and domain include spatial dimensions, typically latitude and longitude, the temporal dimensions which often spans from minutes to decades as well as the associated numerous atmospheric and environmental variables such as aerosol optical depth, cloud droplet concentration, surface sensible heat flux, sea surface temperature and humidity. These datasets are typically generated through remote sensing technologies like satellites or are simulated using high-resolution climate models. For example, NASA's MODIS (Moderate Resolution Imaging Spectroradiometer) satellite provides global aerosol and cloud property data at high spatial and temporal resolutions. Climate models, on the other hand, generate synthetic datasets that simulate aerosol-cloud interactions under various climate scenarios, allowing for the study of these interactions in a controlled environment.

Clustering high-dimensional spatiotemporal climate data poses several challenges. Traditional clustering methods, such as k-means or hierarchical clustering, often struggle with large datasets, as they rely on Euclidean distances that fail to capture the complex, nonlinear relationships between climate variables. These methods may also suffer from the "curse of dimensionality," where the performance of clustering algorithms degrades as the number of variables increases. Moreover, spatiotemporal dependencies in the data make it difficult for traditional approaches to fully account for the interconnectedness between spatial regions and their associated time periods leading to suboptimal clustering results.

To address these challenges, deep unsupervised clustering methods have emerged as a promising alternative. These approaches leverage deep learning architectures to automatically learn hierarchical representations of the data, capturing the underlying patterns across spatial and temporal scales. Deep clustering models such as autoencoders or graph neural networks can extract latent features from high-dimensional climate data, enabling the clustering of complex datasets in a way that traditional methods cannot. These techniques also allow for the incorporation of spatial and temporal dependencies, leading to more accurate and meaningful clusters that reflect the true variability in aerosol-cloud interactions. To assess the relative strengths and limitations in handling climate data, it is vital to compare traditional and deep clustering approaches. While traditional methods are computationally efficient and easier to interpret, they may not fully capture the intricate, multi scale relationships present in aerosol-cloud interactions. In contrast, deep clustering methods though more complex and computationally intensive, offer a more flexible framework for modeling nonlinear dependencies and extracting meaningful patterns from high-dimensional data. Such a comparison could provide valuable insights into how best to cluster multivariate spatiotemporal climate data, ultimately advancing our understanding of aerosol-cloud interactions and improving climate models.

The rest of the paper is organized as follows. In Section II, we define the clustering problem and Section III details related work in this study area. Section IV introduces the selected traditional and deep clustering algorithms for this study while Section V describes the data used. Section VI describes the the evaluation approaches and present the results and Section VIII concludes our study.

II. Problem Definition

Given unlabeled multivariate spatiotemporal climate data, and without prior knowledge of sub-group memberships, our goal is to efficiently partition the data into distinct sub-groups based on temporal similarities among data points. To be specific, assume n atmospheric variables (x_i) measured over a grid region covering L longitudes and W latitudes and stored in a vector $X = \{x_1, x_2, x_3, ..., x_n\}$ such that, for each time step every grid location has n values for all variables. Variables are measured for T different time steps, $X_i = \{x_1, x_2, x_3, ..., x_n\}$, $i \in \{1, ..., T\}$.

Input: $Dataset = \{X_1, X_2, X_3, ..., X_T\}$,

$$X_i = \left\{ \begin{bmatrix} x_1(1,1) & x_1(1,2) & \cdots & x_1(1,W) \\ x_1(2,1) & x_1(2,2) & \cdots & x_1(2,W) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(L,1) & x_1(L,2) & \cdots & x_1(L,W) \end{bmatrix} \begin{bmatrix} x_2(1,1) & x_2(1,2) & \cdots & x_2(1,W) \\ x_2(2,1) & x_2(2,2) & \cdots & x_2(2,W) \\ \vdots & \vdots & \ddots & \vdots \\ x_2(L,1) & x_2(L,2) & \cdots & x_2(L,W) \end{bmatrix} \right. \\ \left. \vdots \\ \begin{bmatrix} x_n(1,1) & x_n(1,2) & \cdots & x_n(1,W) \\ x_n(2,1) & x_n(2,2) & \cdots & x_n(2,W) \\ \vdots & \vdots & \ddots & \vdots \\ x_n(L,1) & x_n(L,2) & \cdots & x_n(L,W) \end{bmatrix} \right\} \quad (1)$$

, where X_i represents one observation, x represents one variable of an observation, $i \in \{1, ..., T\}$, T represents the number of time steps, n represents the number of atmospheric variables, L and W represent the longitude and latitude respectively.

Output: Our proposed clustering model should partition the $Dataset = \{X_1, X_2, ..., X_T\}$ into k clusters: $C_1, C_2, ..., C_k$, where $k < T$, such that objects within the same cluster are similar to each other and dissimilar to those in other clusters.
Formally:

$$C_1 = \{X^1_{C_1}, X^2_{C_1}, ..., X^{n_1}_{C_1}\}, C_2 = \{X^1_{C_2}, X^2_{C_2}, ..., X^{n_2}_{C_2}\}, ...,$$
$$C_k = \{X^1_{C_k}, X^2_{C_k}, ..., X^{n_k}_{C_k}\}$$

$$X^i_{C_j} \in X, i \in \{1, ..., n_j\}, j \in \{1, ..., k\}$$

, where n_j = number of observations of cluster j.

$$\bigcup_{j=1}^{k} C_j = X \text{ and } C_j \cap C_l = \varnothing$$

Here, $j \neq l$ and $(j, l) \in \{1, ..., k\}$ for each pairs of clusters.

III. Related Work

Finding clusters of events in spatiotemporal data analysis is vital. Exploratory statistical techniques rely on unsupervised machine learning algorithms to identify groups of data in n-dimensional space by using a similarity or dissimilarity metric [6]. Steinbach et al [19] tested different clustering algorithms on high dimensional data and concluded that the principal challenge is to overcome the "curse of dimensionality". Ferstl et al [8] analyzed the evolution of single clusters over time, via space-time cluster surfaces, and the simultaneous depiction of all clusters in a spatiotemporal context, via stacked time-cuts from weather data. Ashesh et al [5] proposed an effective, simple, and algorithmic approach for labeling any spatiotemporal climate and environmental data with 22 principal components using K-Means. Birant et al [1] proposed an ST-DBScan, a variant of DBScan that discover clusters based on non-spatial, spatial and temporal values of the objects.

IV. Traditional and Deep Clustering algorithms

Tradition Clustering Algorithms

In order to capture all semantics, we shortlisted the most stable and prominent algorithms from each category. For this experiment, we choose the hierarchical agglomerative clustering(HAC) from hierarchical category, K-Means from Centroids-based category, spectral from the graph-based category, gaussion mixture model from distribution-based category and affinity propagation from exemplar-based category of clustering algorithms. The performance and stability of these algorithms were evaluated on both synthetic and real-world climate data.

A. Kmeans Clustering

The K-Means algorithm is a point-assignment type partitioning method that generates clusters using multivariate means estimation in the Euclidean space [18]. Given a dataset of n points $X = \{x_1, x_2, ..., x_n\}$, where each point x_i is in d-dimensional space, the K-means algorithm partitions these n data points into k clusters $C = \{C_1, C_2, ..., C_k\}$ by minimizing the following objective function: $\min \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$, where: μ_i is the centroid of cluster C_i, calculated as: $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$, $||x - \mu_i||^2$ is the squared Euclidean distance between a data point x and the centroid μ_i. The steps of the algorithm are:

- Initialize k centroids randomly.
- Assign each data point to the nearest centroid: $C_i = \{x : ||x - \mu_i|| \leq ||x - \mu_j||, \forall j \neq i\}$
- Recalculate the centroids.
- Repeat steps 2 and 3 until convergence.

B. Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is a distribution-based clustering algorithms. These are a powerful clustering technique that can handle more complex data distributions especially in tasks where cluster shapes are not necessarily spherical and where probabilistic assignments are beneficial [3]. The Gaussian Mixture Model assumes data is generated from a mixture of k Gaussian distributions. The probability density function of a GMM is: $p(x|\Theta) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i)$, where k is the number of Gaussian components, π_i are the mixture weights such that $\sum_{i=1}^{k} \pi_i = 1$, $\mathcal{N}(x|\mu_i, \Sigma_i)$ is the multivariate Gaussian distribution with mean μ_i and covariance Σ_i, defined as: $\mathcal{N}(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)\right)$

Objective Function

The objective is to maximize the log-likelihood of the observed data $\{x_1, x_2, ..., x_n\}$ given the model parameters $\Theta = \{\pi_i, \mu_i, \Sigma_i\}$: $\mathcal{L}(\Theta) = \sum_{j=1}^{n} \log\left(\sum_{i=1}^{k} \pi_i \mathcal{N}(x_j|\mu_i, \Sigma_i)\right)$.

C. Spectral Clustering

Spectral clustering methods are graph-based approaches to unsupervised clustering of data were clustering is translated into a graph partitioning problem [20]. Given a set of data points $X = \{x_1, x_2, ..., x_n\}$ and a similarity graph $G = (V, E)$ where V represents vertices (data points) and E represents edges (pairwise similarities), the steps of spectral clustering are as follows:

- **Construct the Similarity Graph** Define a similarity matrix $W \in \mathbb{R}^{n \times n}$ where W_{ij} measures the similarity between data points x_i and x_j. $W_{ij} = \text{similarity}(x_i, x_j)$
- **Compute the Graph Laplacian** The degree matrix D is a diagonal matrix where D_{ii} represents the sum of the similarities of node i: $D_{ii} = \sum_j W_{ij}$. The unnormalized graph Laplacian L is defined as: $L = D - W$. Alternatively, the normalized graph Laplacian can be defined as: $L_{\text{sym}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, which implies $L_{\text{rw}} = I - D^{-1}W$.
- **Compute the Eigenvectors** Compute the first k eigenvectors of the Laplacian L (or L_{sym}, L_{rw}) corresponding to the smallest k eigenvalues. Let $U \in \mathbb{R}^{n \times k}$ be the matrix where each row represents an eigenvector. $LU = \lambda U$
- **Clustering Objective** Construct the matrix $U = [u_1, u_2, ..., u_k] \in \mathbb{R}^{n \times k}$, where u_i are the first k eigenvectors. Treat each row of U as a point in $\mathbb{R}^k$, and cluster these points using the k-means algorithm: $\min \sum_{i=1}^{k} \sum_{x \in C_i} ||U_i - \mu_i||^2$, where μ_i is the centroid of cluster C_i in the space spanned by the eigenvectors.
- **Objective Function** The spectral clustering algorithm minimizes the following objective function, which captures the cut in the graph: $\text{MinCut}(C_1, ..., C_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(C_i, V \setminus C_i)}{|C_i|}$. This leads to clustering by minimizing the normalized cut: $\text{Ncut}(C_1, ..., C_k) = \sum_{i=1}^{k} \frac{W(C_i, V \setminus C_i)}{\text{vol}(C_i)}$.

D. Affinity Propagation

Affinity Propagation (AP) algorithm is a clustering algorithm that belongs to the category of exemplar-based clustering methods. An exemplar is a representative data point that best represents a cluster. Affinity Propagation clustering finds clusters by identifying *exemplars*, which are representative data points [9]. The algorithm operates by passing two kinds of messages between data points: responsibility $r(i,k)$ and availability $a(i,k)$. These messages are updated iteratively until convergence.

Objective Function Let $s(i,k)$ be the similarity between data point i and k, where larger values of $s(i,k)$ indicate a higher similarity. The objective is to maximize the net similarity of all points to their exemplars, defined as: $\max \sum_{i=1}^{n} s(i, \text{exemplar}(i))$.

Steps of the Algorithm

- Responsibility Update ($r(i,k)$): This reflects how well-suited point k is to serve as the exemplar for point i, relative to other candidate exemplars. It is updated as follows: $r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\}$.
- Availability Update ($a(i,k)$): This reflects how appropriate it would be for point i to choose point k as its exemplar, considering other points' support for k. It is updated as: $a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right)$. For the diagonal elements where $i = k$ (self-availability), the update rule is: $a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k))$.
- Convergence: The algorithm iterates the updates of $r(i,k)$ and $a(i,k)$ until the messages converge. Points are assigned to the exemplar that maximizes the sum of responsibility and availability: $\text{Exemplar}(i) = \arg\max_k (r(i,k) + a(i,k))$.

E. Agglomerative Hierarchical Clustering (HAC)

This is an important and well-established technique in unsupervised machine learning. The goal is to produce a hierarchical series of nested clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top [19]. Agglomerative Hierarchical Clustering (HAC) is a bottom-up clustering approach, where each data point starts as its own cluster, and clusters are merged iteratively based on a defined criterion (e.g., minimum distance). Let $X = \{x_1, x_2, \ldots, x_n\}$ represent the dataset of n points, and initially, we consider each point as its own cluster, i.e., $\{C_1, C_2, \ldots, C_n\}$.

Objective Function The objective of HAC is to minimize the inter-cluster distance $d(C_i, C_j)$ between clusters C_i and C_j. The commonly used distance metrics are: *Average-linkage (mean)*: $d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} \|x - y\|$.

Clustering Steps:

- Initialization: Each point starts as its own cluster: $C = \{C_1, C_2, \ldots, C_n\}$.
- Distance Calculation: Compute pairwise distances between clusters C_i and C_j using one of the linkage criteria.
- Cluster Merging: At each step, merge the two closest clusters: $C_i, C_j = \arg\min_{C_i, C_j} d(C_i, C_j)$.
- Update Clusters: After merging, update the cluster set and recalculate distances between the newly formed cluster and the remaining clusters.
- Repeat:Repeat steps 2-4 until all points are in a single cluster or a predefined number of clusters k is reached.
- Stopping Criterion: The algorithm stops when a desired number of clusters k is obtained, or when the minimum inter-cluster distance exceeds a threshold.

Deep Unsupervised Clustering Algorithms

The following algorithms are from a family of deep unsupervised clustering algorithms and basically employ deep neural network for dimensionality reduction. The low-dimensional latent features produced by the deep neural network are subjected to a traditional clustering algorithm to obtain desired clusters.

F. Deep Embedding Clustering (DEC)

DEC combines deep learning and clustering to enhance clustering performance [23]. Clustering is improved by learning a feature space in which data points cluster more naturally, improving the separability of distinct data clusters without any supervision. They use an autoencoder framework to jointly perform dimensionality reduction and clustering by mapping high-dimensional data to a lower dimensional latent space in order to maintain the essential structure of the original data. They begin by learning an initial embedding that reduces data noise and captures intrinsic structure, then introduce a clustering layer that performs iterative clustering by refining the embeddings using a Kullback-Leibler (KL) divergence objective function.

G. Deep Temporal Clustering (DTC)

DTC uses a temporal autoencoder to integrate dimensionality reduction and temporal clustering in a single end-to-end unsupervised learning framework [13]. The core idea is to jointly learn low-dimensional representations of time-series data while simultaneously clustering these temporal features without any supervision. Then it jointly optimizes the clustering objective and the dimensionality reduction objective. For effective latent representation, they used 1D convolution layer followed by a bidirectional LSTM. Finally, clustering is applied to the reduced latent representations. Optimization is done by minimizing the clustering and reconstruction loss.

H. Deep Spatiotemporal Clustering (DSC)

DSC leverages deep learning and spatiotemporal modeling to address multiple challenges in climate data analysis, such as high-dimensionality, noise, and complex temporal dependencies, and focuses on segmenting climate patterns over time and across geographic regions [7]. To effectively capture complex patterns, DSC integrates temporal convolutional layers and LSTM modules. Its key innovations is

in its clustering objective, which minimizes reconstruction error within clusters while capturing temporal variations in data points. DSC uses a joint loss function that incorporates clustering loss and reconstruction loss to preserve temporal continuity.

I. Deep Adaptive Image Clustering (DAC)

DAC leverages deep learning to adaptively adjust clustering parameters and refine feature representations. It uses an autoencoder to learn clustering-friendly latent representation and a clustering-oriented loss function for refining clusters [4]. DAC employs a two-phase process: feature extraction and adaptive clustering. In the first phase, an autoencoder is used to compress high-dimensional image data into a lower-dimensional latent space where essential features are preserved and in the second phase, a trainable clustering layer is applied which dynamically adjusts cluster centers by minimizing a clustering loss.

V. Dataset selection and preprocessing

A. Synthetic Dataset

To show the complexity of the real-world dataset, we generate a synthetic weather data, homogeneous with similar level of complexity and pre-defined number of temporal clusters. This number is set to five (5) with the following interval: cluster 1 ranges from $[-\infty, > 20]$, cluster 2 ranges from $[> 19, > 40]$, cluster 3 ranges from $[> 39, > 60]$, cluster 4 ranges from $[> 59, > 80]$, cluster 5 ranges from $[> 79]$, limited to five variables which are sea surface temperature (sst), temperature at 2m above sea level (t2m), snowmelt (smlt), skin temperature (skt) and Total cloud cover (tcc) with a ($10° \times 10°$) spatial coverage and 100 time steps. To test the effectiveness of both the clustering and evaluation algorithms on spatiotemporal data, we designed the records in five clusters to be much similar to each other in comparison to a third cluster. So, if the clustering algorithm works well, it should identify the boundary between the two identical clusters.

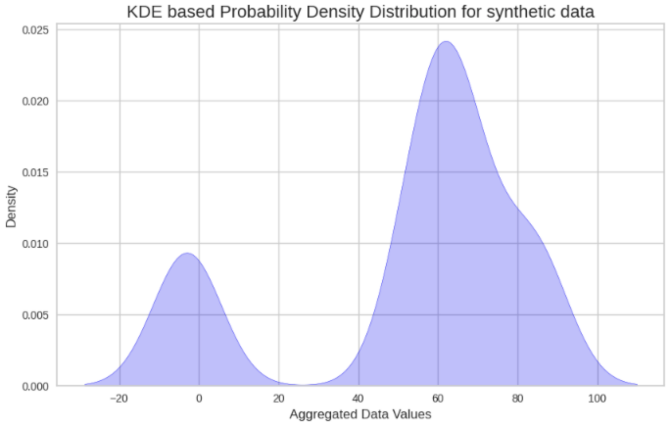

Fig. 1: Probability density distribution for synthetic Data

Figure 1 is the probability density distribution that shows the normal distribution of our synthetic data where we can deduce the mean and deviation. The densities of our continuous variables lie between the range [-20, 100]. From the distribution, we see that approximately -3 and 63 are most frequent in our dataset hence increased densities. To compute the probability of a given value (X) to lie in the range of an interval [a,b]:

$$P(a < X < B) = \int_a^b f(x)\,dx$$

, where a = lower bound and b = upper bound. We did not include null values or extreme values or other noise aspects often found in real world data. For this reason, we do not require data preprocessing.

B. Real-world Dataset

The real world spatiotemporal climate dataset used in this study is the fifth generation of atmospheric reanalyses of the global climate developed by the European Centre for Medium-Range Weather Forecasts [11]. We primarily use open-access atmospheric reanalysis data from European Centre for Medium-Range Weather Forecasts (ECMRWF) ERA-5 global reanalysis product [11]. It contains 31 km high resolution reanalysis and a reduced resolution ten-member ensemble.

TABLE I: ERA-5 Data Description.

Var	Variable	Range	Unit
sst	Sea Surface Temperature	[285, 300]	k
slp	Sea Level Pressure	[98260, 103788]	pa
sshf	Surface Sensible Heat Flux	[-674528, 200024]	J/m^2
t2m	2m temperature	[281, 299]	k
slhf	Surface Latent Heat Flux	[-1840906, 90131]	J/m^2
u10	10m u-wind component	[-16, 19]	m/s
v10	10m v-wind component	[-15, 16]	m/s

Table I describes the variables and their respective ranges. These variables are included in the selected dataset based on their impact on air-sea-cloud interactions. These are daily observation over a period of one year across a latitude-longitude grid of [41, 41] and after preprocessing the data is rendered in both 2D [365, 16128] and 4D [365, 48, 48, 7]. Figure 2 depicts the probability density distribution (PDF) of our real world dataset. We notice that densities are very high around 0. This shows a Gaussian distribution or a continuous probability distribution for our real-valued variable.

Data pre-processing is inevitable in data mining. Our spatial data spans across both land and water hence variables like SST will have missing values on land. To make our real world dataset homogeneous, we replaced every variable's value whose corresponding SST is missing to nan. Our pre-processing stage saw major processes and include data transformation, data normalization and dimensionality reduction.

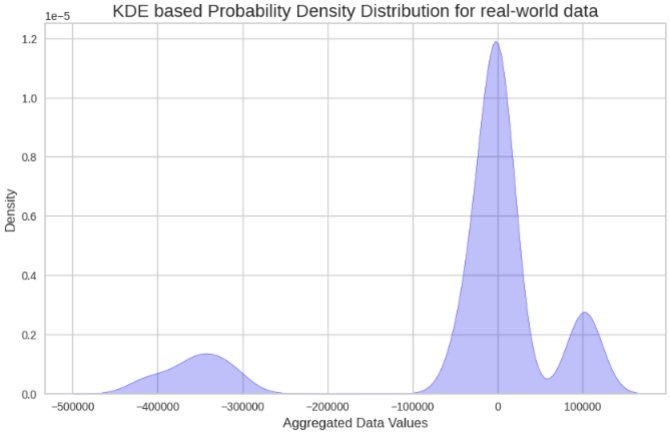

Fig. 2: Probability density distribution for real world Data

Data Transformation Clustering algorithms accept specified data structures and data types as input parameters, none of which accepts our data structure, hence there is the need to transform our 4-dimensional data to a reduced two dimension consumable by our algorithms. Furthermore transformation is also necessary for the ease of algorithm comparison and interpretation [14]. Our data in the form of a NetCDF (network Common Data Form) file is initially read as an x-array. To achieve robust data mapping, we convert our x-array into a dask DataFrame and later read into a pandas DataFrame. In the process, the temporal aspect (time dimension) of the data now represents the row index and the spatial aspect (longitude and latitude) form the columns of the respective attributes present i.e every variable with its associated longitude and latitude pair will together form respective columns. Our resulting data set is in the form of a 2 dimension pandas dataframe of Z number of rows and [Z, X, Y] number of columns considering that Z is the number of time steps (365 days), X is the longitude (41) and Y is the latitude (41).

Data Normalization Normalization is crucial and proven by our achieved clustering results. Our goal is to scale the entire data to fit in a common scale, without distorting differences in the ranges of values or losing information. Both our mock and real world dataset are unevenly distributed. Normalization is done to reduce the tendency (of a single data point) to steer the performance of the algorithm to one direction due to their values either being too large of too small. There are two popular normalization algorithms: The MinMaxScaler and StandardScaler. In this paper, we will use the MinMaxScaler function from sklearn [2]. This function scales our derived dataset to a specific range using each feature's minimum and maximum value. We achieved this by applying the below equation:

$$X\_norm = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

where: min, max = feature_range, X : Feature values, X_min: Minimum feature value, X_max:Maximum feature value This approach will improve the performance and speed of the execution of our clustering algorithms. Scaling also plays a great role when comparing our models in terms of their performances. To normalise our resulting data, we call the MinMaxScaler function which accepts our transformed data as input and outputs transformed data in the form of a pandas dataframe with the same number of input rows and columns.

Dimensionality Reduction Dimensionality reduction is a supervised learning approach. Given the number of rows n = 365 of a 2-D dataset and the number of columns p = 11,767 the dataset is considered high dimension when: $n \leq p$. Reducing these number of columns without distorting the pattern will reduce training time, remove noise and improve visualization. Reduction is achieved through the Principal Component Analysis (PCA), a technique designed to model linear variabilities in high-dimensional data. [10] This algorithm requires that the number of components

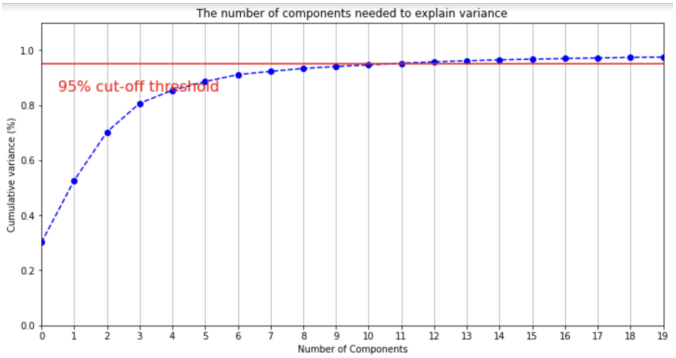

Fig. 3: Optimal number of components

are pre-defined. Figure 3 is a line plot depicting the optimal number of components needed to explain variance and their respective performance level on our real world dataset. The optimal number of components is the number where the line touches the cumilative variance threshhold and in our case it is 11 components.

C. Optimal numbers of clusters

We determined the optimal number of clusters using the Distortion Score Elbow approach for K-Means clustering. Distortion uses the Euclidean distance metric to compute the sum of squared distances from each point to its assigned center. Figure 4 are the results obtained when our elbow approach is applied to our real world dataset. We notice that the SSE begins to flatten out and we see an inflection point in the plot generated. To determine the optimal number of clusters, we select the value of K at the "elbow"; the point at which the distortion starts decreasing in a linear fashion. When visualized, the line takes the form of an elbow, hence the name of the method. For our real world dataset, we can conclude that the optimal number of clusters is 7. We tested a range of K's from $[2, 20]$ and $K = 7$ yield better results.

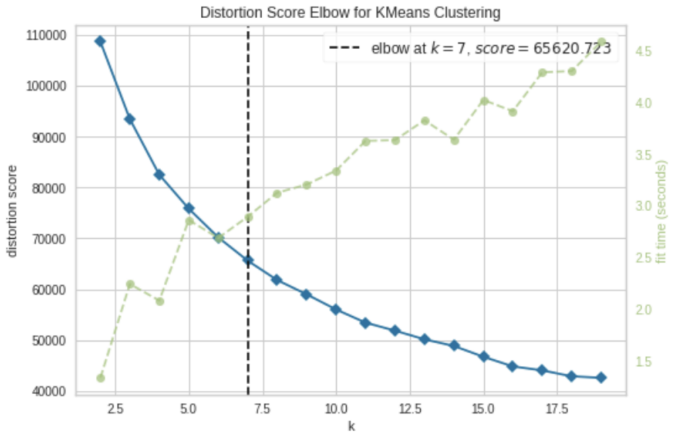

Fig. 4: Elbow Method for Real Data

VI. Evaluation

A. Performance Evaluation Metrics

In the absence of ground truth, we evaluate the performance of our proposed model on six internal cluster validation measures: These measures seek to balance the *compactness* and the *separation* of formed clusters through minimizing intra-cluster distance and maximizing the inter-cluster distance respectively.

Silhouette Score: This index measures the normalised difference between the intracluster and intercluster average distances [17]. Silhouette Score $= \frac{1}{N}\sum_{i=1}^{N}\frac{b_i - a_i}{\max(a_i, b_i)}$, where N is the total number of samples, a_i is the average distance between the sample i and other samples within the same cluster, and b_i is the average distance between the sample i and samples in the nearest neighboring cluster.

Davies-Bouldin score (DB): This measures the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances [16]. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering.

$DB = \frac{1}{n_c}\sum_{i=1}^{n_c}\max_{j\neq i}\left(\frac{S_i+S_j}{M_{ij}}\right)$, where n_c :number of clusters, S_i : average distance within cluster i, M_{ij} : distance between cluster i and cluster j

Calinski-Harabas score (CH): This is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances [21]. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering.

$CH = \frac{BGSS}{WGSS}\times\frac{N-K}{K-1}$, where BGSS is between-group sum of squares (between-group dispersion), WGSS is within-group sum of squares (within-group dispersion), N is the total number of observations, and K is the total number of clusters.

Average inter-cluster distance (I-CD): This is the minimum distance between any two data points belonging to different clusters [6]. The intercluster distance between clusters C_i and C_j using Euclidean distance can be expressed as: $d(C_i, C_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$, where x_{ik} and x_{jk} are data points in clusters C_i and C_j, and n is the number of dimensions.

Average Variance: The generated cluster compactness and homogeneity can be derived through a measure of the variance of the cluster. The distribution of the time series over various clusters is observed to minimize the intracluster variance [15]. For a cluster C_i belonging to a set of our final clustering, we compute its variance as shown below:

ClusterVariance$(C_i) = \frac{1}{|C_i|}\sum_{x\in C_i}\|x - \mu_i\|^2$, where C_i is the cluster, $|C_i|$ is the number of data points in cluster C_i, x represents a data point in the cluster, and μ_i is the centroid (mean) of cluster C_i.

Average root mean squared error (RMSE): For each observation, the error rate is the distance between every observation and its associated cluster centroid. The average total error is then the root sum of individual errors associated with each data point [22]. Average RMSE $= \frac{1}{k}\sum_{i=1}^{k}\sqrt{\frac{1}{n_i}\sum_{j=1}^{n_i}(d(x_j, \mu_i))^2}$, where k is the number of clusters in the clustering solution, n_i is the number of data points in cluster C_i, $d(x_j, \mu_i)$ represents the distance between data point x_j in cluster C_i and the centroid μ_i of that cluster.

B. Stability Evaluation Metrics

Cluster stability refers to the consistency of clustering results when the algorithm is run multiple times (e.g., with different initializations, subsamples of data, or different hyperparameters). A stable clustering algorithm will produce similar clusters across multiple runs.

Optimal Transport Alignment (OTA) for cluster stability: Li et al. [12] recently developed an optimal transport framework for cluster stability. The approach seek ways to quantify the cost of transporting a probability distribution between two clusterings. By computing the average OT transport cost between clusters from different runs, we can quantify how much clusters change between runs. They defined the stability between two clustering assignments C_1 and C_2 with n clusters each as the OT transport cost T. $T = \min_{\pi\in\Pi(C_1,C_2)}\sum_{i=1}^{n}\sum_{j=1}^{n}\pi_{ij}\, d(c_i^{(1)}, c_j^{(2)})$, where π is the transport plan, which specifies how much mass (or points) from cluster i in C_1 is transported to cluster j in C_2. $d(c_i^{(1)}, c_j^{(2)})$ is the cost (e.g., Euclidean distance) of transporting between clusters $c_i^{(1)}$ in C_1 and $c_j^{(2)}$ in C_2. $\Pi(C_1, C_2)$ represents the set of all valid transport plans between clusters in C_1 and C_2. A lower average OT transport cost indicates that clusters are more consistent and thus the clustering algorithm is more stable.

TABLE II: Performance Evaluation on both synthetic and real world data: The column labeled *Data* describes all datasets used for this study and their respective predefined number of clusters. The performance column presents six (6) internal clustering measures used to obtain the performance score. Row-wise the table is split into an upper section which presents the results obtained from applying these algorithms on synthetic data while the lower section presents results from applying the selected algorithms on real-world data.

	Synthetic data									
		Traditional Clustering Algorithms					**Deep Clustering Algorithms**			
Data	**Performance**	**Kmeans**	**HAC**	**GMM**	**SC**	**AP**	**DAC**	**DTC**	**DEC**	**DSC**
Synthetic Data	**Silhouette↑**	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	DB ↓	1.2588	1.2588	1.2588	1.2588	1.2588	1.2588	1.2588	1.2588	1.2588
with five(5)	**CH ↑**	1.1949	1.1949	1.1949	1.1949	1.1949	1.1949	1.1949	1.1949	1.1949
	RMSE ↓	3.4174	3.4174	3.4174	3.4174	3.4174	3.4174	3.4174	3.4174	3.4174
pre-defined clusters	**Var↓**	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676
	I-CD ↑	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Real World data									
Data	**Performance**	**Kmeans**	**HAC**	**GMM**	**SC**	**AP**	**DAC**	**DTC**	**DEC**	**DSC**
ERA5	**Silhouette ↑**	0.2939	0.2354	0.3488	0.1942	0.2750	-	0.3023	0.3171	**0.3497**
Reanalysis Data	**DB ↓**	1.5929	1.5669	**1.5035**	1.7401	1.6764	-	1.5294	1.5772	1.5126
	CH ↑	107.2416	97.6476	100.2409	96.0978	100.7999	-	89.3054	121.8213	**122.6466**
with seven (7)	**RMSE ↓**	**13.4119**	13.8148	13.7023	13.8834	13.6784	-	14.1963	14.5532	14.3933
	Var ↓	0.1030	0.1032	0.1029	0.1032	0.1032	-	0.1032	0.1029	**0.1028**
optimal clusters	**I-CD ↑**	7.8795	7.7810	**7.9655**	5.4339	6.0290	-	7.2847	7.8563	7.6077

VII. Experiment Results

Hardware - All models are executed on AWS cloud environment using 20GB of S3 storage with 30 GB of ml.g4dn.xlarge GPU. The hardware used is a macOS Sonoma version 14.4.1, 16 GB, M1 pro chip. We applied the same python library across all models for homogeneity.

Results in detail: This study seeks to measure the performance (i.e the ability to effectively capture the intrinsic and complex spatial and temporal patterns) and robustness (i.e ability to produce stable and consistent results) of existing traditional and deep unsupervised clustering algorithms when applied to high dimensional spatiotemporal data.

A. Performance

In the absence of ground truth labels in our real world data, the performance of the selected algorithms is evaluated on both synthetic and real world data using internal validation metrics. These metrics evaluate cluster compactness and separation. Table II presents our experiment results. All models were executed 20 times and the median results were considered. The results of best performing algorithms are bolded. Selected algorithms perform equally on our synthetic data with 99.9% accuracy as shown by the silhouette index. Consistent metric values suggested all models achieved similar high precision, despite a low inter-cluster distance and low cluster variance (0.0676), alongside a high intra-cluster RMSE (3.4174), highlighting challenges in cluster separation within high-dimensional feature space. In real-world data evaluation, Gaussian Mixture Models (GMM) outperformed traditional and some deep clustering methods, achieving a silhouette score of 0.3488, due to its capacity for modeling high-dimensional, elliptical clusters using maximum likelihood estimation and the Expectation-Maximization (EM) algorithm. Secondly, GMMs assume that all clusters follow a Gaussian distribution which allows them to capture data with elliptical shapes rather than requiring spherical clusters. SC, however, struggled due to noise sensitivity and diminishing discriminatory power in high-dimensionality, leading to low performance. DSC achieved the best results with a silhouette score of 0.3497, CH: 122,6466 and average variance: 0.1028, attributed to its dual optimization approach of clustering and reconstruction loss, which consistently refines the clustering in iterative cycles.

B. Stability

Table 5 illustrates the performance of our selected clustering algorithms on a complex real-world dataset. Notably, we excluded stability measurements for the DAC due to its subpar performance discussed in Section VII-A. Despite receiving the lowest performance score in Section VII-A, SC exhibited the highest stability, reflected in its lowest stability score of 0. This robustness is attributed to its use of eigenvectors and eigenvalues, which mitigates the initialization issues commonly faced by centroid-based models like K-means. Conversely, AP demonstrated the least stability, primarily due to its dependence on a damping factor that regulates the influence of prior iterations on current updates. The typical default value for this factor ranges from 0.5 to 0.9, but it requires meticulous tuning to prevent oscillations or convergence to suboptimal solutions. Second, its high reliance on preference values (i.e values that determine which points are likely to be exemplars) heavily influence the number of clusters so much so that small changes in preference can lead to large changes in the number and size of clusters. Last, AP highly relies on a message passing mechanism easily converges to different

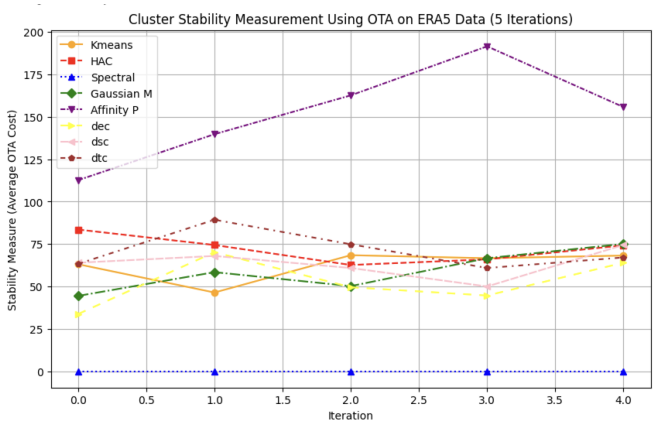

Fig. 5: Optimal Transport Alignment (OTA) for Stability Assessment of traditional and deep unsupervised clustering algorithms on ERA5 data. Evaluation was done after 5 iterations. Different colors of the line chart represent different algorithms. The stability scores range between 0 and 200. The higher the OTA measurement score, the less stable is the algorithm.

local minima and makes the final clustering sensitive to initialization and randomness in the process.

C. Cluster visualization

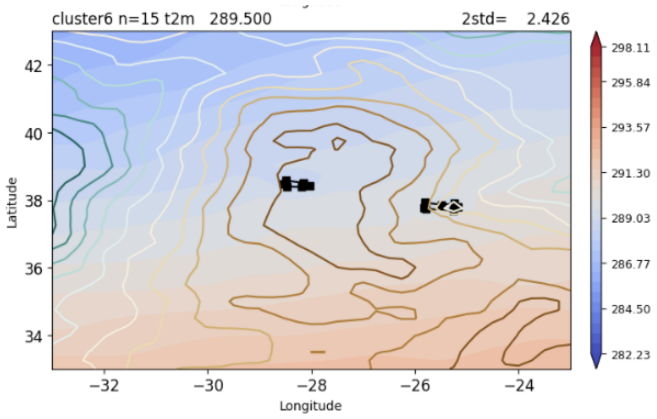

Fig. 6: Smallest generated cluster by the GMM model.

To gain further understanding and support the claims embedded in our performance and stability results, we visualize all generated cluster. For space reasons we present the visualizations Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11 of the largest and smallest generated final clusters of the best performing traditional and deep clustering model for t2m atmospheric variable, these models are GMM and DSC respectively. Each image is associated with a cluster id and number of data points that make up the clusters. At the top left corner is located the value of two standard deviation.

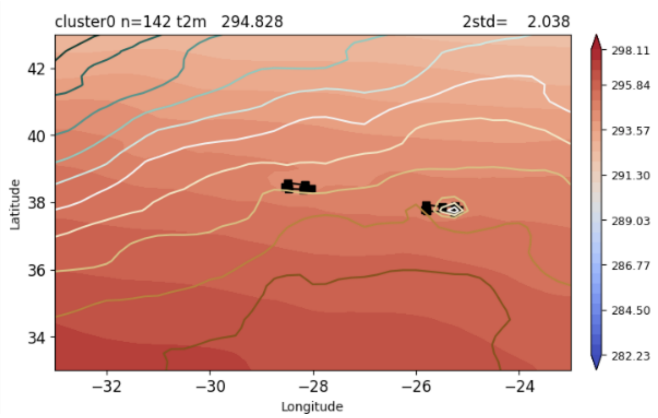

Fig. 7: Largest generated cluster by the GMM model.

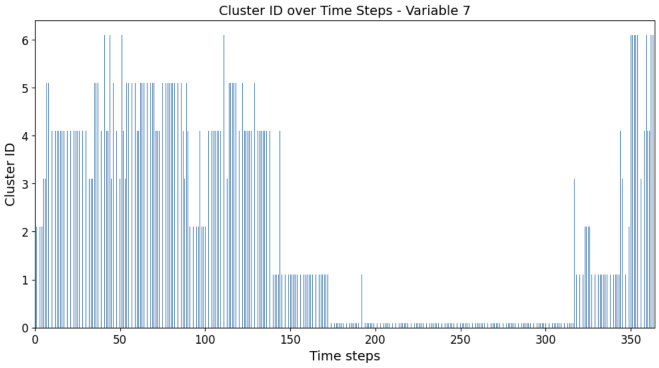

Fig. 8: Final clustering results: This is a bar chart that depicts the final partition of the dataset by the GMM clustering model

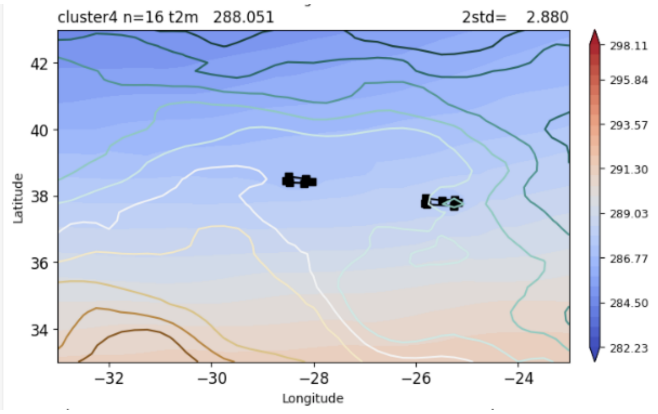

Fig. 9: Smallest generated cluster by the DSC model

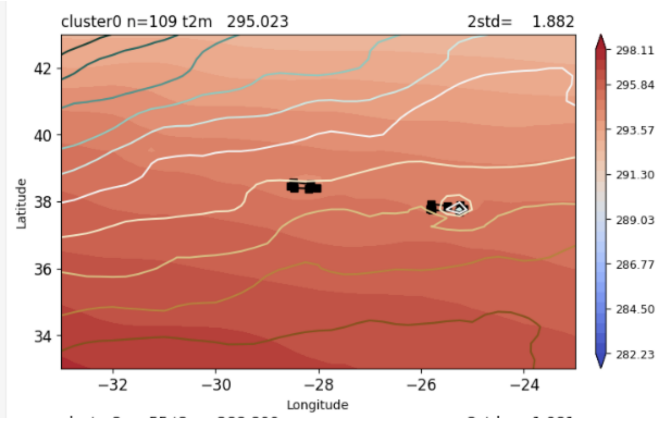

Fig. 10: Largest generated cluster by the DSC model

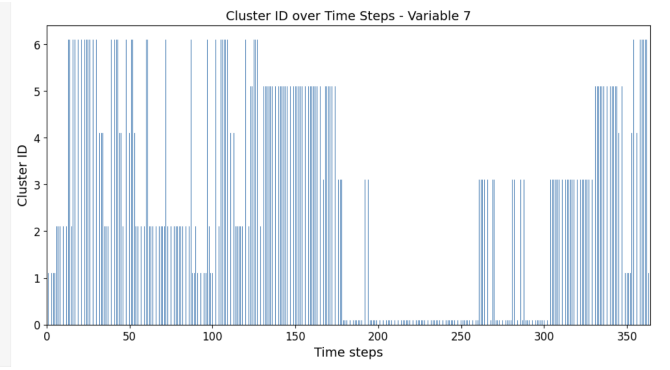

Fig. 11: Final clustering results: This is a bar chart that depicts the final partition of the dataset by the DSC clustering model

VIII. CONCLUSIONS AND FUTURE WORK

With the growing nature of spatiotemporal datasets utilized in disciplines such as atmospheric science, Earth sciences and environment science, public safety and transportation, there is a growing need for automated discovery of spatiotemporal knowledge embedded in these datasets. The complexity of this data limits the usefulness of popular data mining techniques for exploiting embedded patterns. It is therefore imperative to bridge this gap by reducing the complexity of this datasets while making it easily consumable by stakeholders. In this paper we reviewed nine clustering algorithms and their tendency to generate accurate clusters from spatiotemporal weather data. Our findings are as follows. All models performed well on synthetic data. GMM outperformed all traditional and some deep clustering models used in this study. DSC model is the best performing model in terms of average accuracy and average stability when applied to high dimensional spatiotemporal climate data. Temporal multivariate clustering is a relatively new field in the area of spatiotemporal weather data analysis. For future work, we plan to test more clustering models from different categories.

REFERENCES

[1] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.

[2] Jason Brownlee. How to use standardscaler and minmaxscaler transforms in python. *Machine Learning Mastery*, 10, 2020.

[3] Satish Chander and P Vijaya. Unsupervised learning methods for data clustering. In *Artificial Intelligence in Data Mining*, pages 41–64. Elsevier, 2021.

[4] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887, 2017.

[5] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Saba Pasha. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Scientific reports*, 10(1):1–13, 2020.

[6] Brian S Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. Cluster analysis. –john wiley & sons. *Ltd., New York*, page 330, 2011.

[7] Omar Faruque, Francis Ndikum Nji, Mostafa Cham, Rohan Mandar Salvi, Xue Zheng, and Jianwu Wang. Deep spatiotemporal clustering: A temporal clustering approach for multi-dimensional climate data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 90–105. Springer, 2023.

[8] Florian Ferstl, Mathias Kanzler, Marc Rautenhaus, and Rüdiger Westermann. Time-hierarchical clustering and visualization of weather forecast ensembles. *IEEE transactions on visualization and computer graphics*, 23(1):831–840, 2016.

[9] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[10] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 37(38):2006, 2006.

[11] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, ..., Sebastien Villaume, and Jean-Noël Thépaut. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[12] Jia Li, Beomseok Seo, and Lin Lin. Optimal transport, mean partition, and uncertainty assessment in cluster analysis. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 12(5):359–377, 2019.

[13] Naveen Sai Madiraju. Deep temporal clustering: Fully unsupervised learning of time-domain features. Master's thesis, Arizona State University, 2018.

[14] S Manikandan. Data transformation. *Journal of Pharmacology and Pharmacotherapeutics*, 1(2):126, 2010.

[15] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John wiley & sons, 2010.

[16] Frédéric Ros, Rabia Riad, and Serge Guillaume. Pdbi: A partitioning davies-bouldin index for clustering evaluation. *Neurocomputing*, 528:178–199, 2023.

[17] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[18] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[19] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer, 2004.

[20] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

[21] Xu Wang and Yusheng Xu. An improved index for clustering validation based on silhouette index and calinski-harabasz index. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052024. IOP Publishing, 2019.

[22] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[23] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.