Feature Space Curvature Map: A Method To Homogenize Cluster Densities

1st Kaveh Mahdavi, 2nd Jesus Labarta, 4rd Judit Gimenez
Computer Architecture
Barcelona Supercomputing Center
Barcelona Spain
{kaveh.mahdavi, jesus.labarta, judit}@bsc.es

3nd Atefeh Mousavinia
Technical Communication
Universitat de Barcelona
Barcelona Spain
amousamo@alumnes.ub.edu

5rd Atiyeh Mousavinia
Computer Software
Ashrafi Isfahani University
Isfahan, Iran
a.mousavinia@ashrafi.ac.ir

***Abstract*—The majority of density-based clustering algorithms can not perform properly when data expose very different density through the feature space. These algorithms implicitly presume that all clusters almost have the same density, therefore, they normally use global parameters. Consequently, they are often biased towards finding dense clusters in front of sparse ones. In this paper, we propose a parametric multilinear transformation method to homogenize cluster densities while preserving the topological structure of the dataset. The transformed clusters have approximately the same density while all inter-cluster regions become globally low-density. In our method, the feature space is locally bent by dense data point concentrations the same way as stars bend the space-time dimensions in Theory of Relativity. We present a new Gravitational Self-organization Map to model the feature space curvature by plugging the concepts of gravity and fabric of space into the Self-organization Map algorithm to mathematically describe the density structure of the data. To homogenize the cluster density, we introduce a novel mapping mechanism to project the data from a non-Euclidean curved space to a new Euclidean flat space. Specifically, this mechanism transfers the basis vectors instead of the feature vectors to guarantee the continuity of the mapping function and optimize the computation cost of the algorithm. As a result, our method can efficiently and explicitly homogenize the density of any dataset globally to then apply existing clustering algorithms without modification. Our experimental results over both real-world and synthetic datasets show that our approach outperforms the current statistical-based methods.**

***Index Terms*—Varied densities, Density-based clustering, Topology Preserving, Self-Organization Map**

I. Introduction

Density-based clustering algorithms are now widely used in a variety of applications, ranging from agriculture [6], high energy physics [17], material sciences [14], social network analysis [8] to molecular biology [3]. These approaches regard clusters as regions in the feature space in which the data points are dense and separated by regions of low data point density (noise). However, they fail to properly find clusters in data exposing different densities in various regions of the feature space. This failure results from using a single global density threshold on all the data points.

We refer to multi-density or varied densities clusters as the clusters in different regions of the feature space that are formed in considerably different densities [7]. Typically, density-based clustering algorithms require the user to specify the parameters (typically one or few constants) that define the density-level thresholds, as an input to the algorithm. Properly selecting the value of those parameters becomes even more difficult when the data expose a multi-density structure since a single density threshold parameter that appropriately detect such structure would be different in various regions of the feature space. Therefore, the main goal of this paper is to introduce a new preprocessing method for homogenizing the density of data in order to enable existing single density threshold algorithms to properly identify the actual clusters structure throughout the whole feature space.

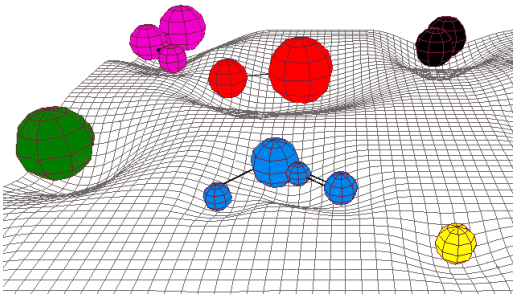


Fig. 1: The stars (distinguished by colors) bend space-time (grid), and the gravitational force among the stars (black lines) can be described by space-time curvature.

In analogy to the Theory of Relativity [4], the computation of distance between objects changes when objects are placed in a space that introduces local curvatures. In this work, we propose a method to map an originally euclidean feature space into a non-euclidean one with local curvatures introduced by the presence of the data points themselves. Then we project the data from a non-Euclidean curved space to a new Euclidean flat space. By projecting data points to their new coordinates in this transformed space, we observe a more uniform density distribution and we show how traditional clustering algorithms in this projected dataset result in significantly better capacity to identify its structure. The key idea is that data clouds can bend the feature space based on their density, which represents the density structure of the data.

According to the Theory of Relativity, see Fig.1, the planets warp space-time, and the standard quantized version of the theory includes massless gravitons delivering the gravitational

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes,creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. DOI 10.1109/IJCNN55064.2022.9892921

force. The space-time curvature is a quantity describing how the local geometry of a subspace differs from the flat space around any single planet. In our method, we assume that a data cluster bends the m-dimensional feature space locally based on its shape and density; we use this property as a source to model the data density structure.

In particular, our approach includes two main steps: first, we apply our new Gravitational Self-Organization Map (GSOM) method to compute the data density structure by using the gravity force in terms of relationships between clusters rather than distance. It computes the feature space bending with a correct topology preserving map to present the potentially complex multi-density structure of the data. Second, we apply our novel parametric multilinear transformation, using the GSOM map, to project the data points to a new linearized and euclidean feature space with more uniform density distribution. Any existing density-based clustering algorithm can then be applied to the projected data to identify its clusters. This very often leads to better clustering analysis performance than using original data, without applying algorithmic changes to the clustering algorithm itself.

Note that the multilinear transformation is a spatial transformation which has commonly been used in Mathematical Physics [18] and Graphics for video compression [9]. The novelty of our approach is that we apply multilinear transformation to map an originally euclidean feature space into a new euclidean space by a given curvature model, in our case computed by the GSOM method, where the projected data points show more uniform density distribution.

The rest of this paper is organized as follows: The intuition and motivation of the multi-density clustering analysis are presented along with an overview of related work and methods in section 2. In section 3, the basic notions and the problem of multi-density are defined. In section 4, the background techniques are briefly reviewed. In section 5, our new Feature Space Curvature Map (FSCM) algorithm is described. The experimental result of our FSCM method is illustrated in section 6. Section 7 concludes the paper with a summary.

II. Related work

Density-Based Clustering refers to the methods that detect clusters in the data based on the idea that a cluster in a feature space is an adjacent region of concentrated points, separated from other clusters by regions that are empty or sparse. DBSCAN [20] computes the density of each data point by counting the existing data points in its ϵ-neighbourhood. However, using a single ϵ can often not adequately characterize the datasets with clusters of very different densities. Several approaches have been proposed to cope with this weakness.

Many efforts have been devoted to solving the varied densities problem by variants of DBSCAN. The HDBSCAN($\hat{\epsilon}$) [13] and OPTIC [2] address this issue by producing the reachability plot to extract clusters. The reachability plot is a 2D plot, with the ordering of the points as processed by these algorithms on the $x-axis$ and the reachability distance on the $y-axis$. As points associating to a cluster have a low reachability distance to their nearest neighbor, the clusters appear as valleys in the reachability plot. However, they are methods to visualize the cluster structures without producing a clustering result explicitly. Thus, these methods manually need to extract the clusters, with varying densities, from the reachability plot. Our approach mathematically combines the strengths of statistical and topological methods to eliminate the need for expert human visual analysis.

The VDBSCAN [10] and MSDBSCAN [19] partition a dataset into different density level sets by statistical analysis, and then estimate ϵ for each density level set. Finally, they use DBSCAN clustering on each density level set with corresponding ϵ and $mPts$ to get clustering results. However, they are not suitable for large datasets since they require several passes of the data and consume high computational time. Our method is more efficient since it is a preprocessing step which enables us to carry out a single pass clustering. Furthermore, it is a scalable approach as it uses the SOM-based iterative resampling schemes rather than the whole dataset.

Recently, neighborhood-based density estimator approaches use local density estimation to overcome the issues of varying densities. CFSFDP [16] instead of finding core points uses a global threshold in the first step, it finds the density peak of every cluster and then links the neighboring points of each peak to form a cluster. LC-CFSFDP [5] and DPC-DBFN [11] improve the clustering performance of CFSFDP by enhancing local density estimators based on a kNN graph and a fuzzy neighbourhood measure, respectively. However, most of these algorithms face problems in handling large datasets since they require high storage to keep the distance matrix. In contrast, we use the GSOM to efficiently learn and build a density structure map of data by preserving its neighboring relations since this structure requires significantly lower storage than the distance matrix.

More recently, density-ratio based scaling methods have also been applied to handle the multi-density data. The density-ratio of a point is the ratio of two density estimates which are calculated by using the same density estimator, but with two different bandwidths. ReScale [21] enables a density-based clustering algorithm to identify clusters with varied densities by rescaling the given dataset. Nevertheless, it rescales each individual feature independently and if a significant overlap exists among clusters on some features, ReScale may become less effective.

Rather than rescaling on each individual feature, [22] proposes a new density-ratio DScale method which rescales the pairwise distance as a multi-dimensional scaling, such that points located at locally high-density areas have higher densities than points located at locally low-density areas. However, DScale fails to be a globally valid CDF (Cumulative Distribution Function) transformation for the entire dataset since it is valid within the λ-neighbourhood of each data point which is treated independently.

Furthermore, CDF-TS [23] density-ratio method applies the same CDF transform process as DScale with an additional "shift" to ensure that: (i) the transformed-and-shifted dataset

becomes approximately uniformly distributed in the scaled λ-neighbourhood; and (ii) then we can use standard Euclidean distance to measure distances between any two transformed-and-shifted points. These advantages are not available in DScale which relies on a rescaled distance which is non-metric and asymmetric. However, CDF-TS does not preserve the potentially complex topological structure of clusters since it still uses the CDF scaling. In contrast, our method uses a novel parametric and global mapping function that preserves potentially complex topological density structure.

The intention of our work is to present a novel generalized technique to homogenize the density of adjacent clusters of varying density. Our method introduces analogies between the data analysis domain and the Theory of Relativity in physics by using the tensor calculus and a modified Self-Organization Map method; Thereby, our FSCM is a parametric, scalable, automated method and it does not depend on any particular clustering algorithm. Especially, in our GSOM we use the SOM-based iterative resampling schemes to optimize the computation cost and increase the efficiency and scalability.

III. The problem of multi-density

In this section, we firstly provide a brief mathematical notation to use throughout this paper, then formalize the general weakness of existing density-based clustering algorithms to stratify multi-density clusters.

We use X to indicate a dataset of n data points $X = (x_1, x_2, ..., x_n)$ where $x_i \in R^m$. Let $pdf(x)$ and $\widehat{pdf}(x)$ stand for the true density of point x and its estimation based on sample data respectively. Besides that, let $\mathcal{N}(x, \epsilon) = \{x' \in X | d(x, x') \leqslant \epsilon\}$ denote the ϵ-neighbourhood of x, where $d : R^m \times R^m \to R$ is the euclidean distance function.

Technically, the density-based clustering algorithms (e.g. DBSCAN [20]) use a small ϵ-neighbourhood to estimate the $pdf(x)$ as follows:

$$\widehat{pdf}(x, \epsilon) = \frac{|\mathcal{N}(x, \epsilon)|}{n\mathcal{V}(\epsilon)} \tag{1}$$

Where $\mathcal{V}(\epsilon)$ is the volume of an m-sphere of radius ϵ [22].

Let C = $\{c_1, c_2, ..., c_q\}$ denote a set of non-overlapping and non-empty clusters where $c_i \subset X$, $\forall_{i \neq j}(c_i \cap c_j = \emptyset)$ and $c_i \neq \emptyset$. Let $m_i = argmax_{x \in c_i} \widehat{pdf}(x)$ and $p_i = \widehat{pdf}(m_i)$ denote the data point with the mode (highest density) of cluster c_i and its corresponding peak density value respectively.

In [21], they present a condition to guarantee these density-based clustering algorithms can identify all clusters in a dataset. The condition is that the estimated density p_i at the mode m_i of each cluster is greater than the maximum of the minimum estimated density along any path through $\widehat{pdf}$ which is linking any two modes:

$$\min_{i \in \{1...q\}} p_i > \max_{j \neq k \ \in \{1...q\}} g_{jk} \tag{2}$$

Where g_{jk} is the highest of the minimum density along the path that links clusters c_j and c_k.

This condition implies that a single density threshold τ must exist to fracture all trajectories between the peaks by nominating the regions with density less than τ to noise. Nevertheless, these density-based clustering algorithms fail to detect all clusters when the peak density of some clusters is lower than a low-density region between some other clusters.

In this paper, we propose a parametric transformation which acts as a preprocessing step to tackle this issue by projecting a given dataset X to X' that has a more uniform density distribution and it also better fulfills this property. As a result, it enables clusters with varied densities in the original space to be identified using a single threshold in the transformed space, something that would be impossible in the original space.

IV. Background

In this section, we discuss an overview of two techniques that we leverage and extend in our analytic approach. We firstly revisit the self-organization map algorithm, then we review the multilinear transformation technique.

A. Self-organization map

Kohonen's Self-Organization Map (SOM) [15] is one of the popular types of neural network which preserves and retains an accurate representation of the topology of the data space.

The SOM often arranges a set of neurons in a 2-D rectangular or hexagonal grid $\mathbb{T}$ in size ς, to establish a discrete topological mapping of an input space $X \in R^{n \times m}$. Ω is the set of neuron indexes. The neurons are represented by a set of weight vectors $V = \{v_1, v_2, ..., v_\varsigma\}$, where v_i is the weight vector associated with neuron i and is a vector of the same dimension $-m-$ of the input, ς is the total number of neurons, and let r_i be the location vector of neuron i on the grid. At the start of the learning, all the weights are initialized to small random numbers. Then the algorithm repeats next two steps until the map converges in order to preserve maximum topological properties of the data on the map.

Each iteration t, it first chooses one random point $x(t)$ of the dataset and selects the winner neuron:

$$\nu(t) = \underset{k \in \Omega}{\operatorname{argmin}} \|x(t) - v_k(t)\| \tag{3}$$

Then it updates the weights of the winner and its neighbors:

$$\Delta v_k(t) = \alpha(t)\eta(\nu(t), k, t)[x(t) - v_{\nu(t)}(t)] \tag{4}$$

where the coefficient $\alpha(t)$ is termed the 'learning rate' which is scalar-valued and it decreases monotonically [15]:

$$\alpha(t) = \alpha_0 . e^{\frac{-t}{\lambda_\alpha}} \tag{5}$$

η is the neighborhood function which quantifies the propagation and decay of the updates to the winner node on its neighbours through the grid topology. The Gaussian form is often used in practice, specifically:

$$\eta(\nu(t), k, t) = exp\left[-\frac{\|r_{\nu(t)} - r_k\|^2}{2\sigma(t)^2}\right] \tag{6}$$

Where $\sigma(t)$ represents the effective range of the neighborhood radius around $\nu(t)$. Like the learning rate, the neighborhood similarly decays with an exponential decay function:

$$\sigma(t) = \sigma_0 . e^{\frac{-t}{\lambda_\sigma}} \tag{7}$$

where λ_α and λ_σ are the decay time constants.

Note that the initial learning rate α_0, radius σ_0, and both time constants are empirically chosen based on the application. In our work, we later present their value in the section V.

The "No Move" [12] criteria is widely used to detect convergence of the learning mechanism. It considers stopping condition that defines no-improvement in SOM's status as no training samples change their best match unit in a complete iteration of the training set. However, when we use the "No Move" criterion, then there could be a case where the weights kept oscillating between iterations. Thus causing the criterion is never met. Therefore, an iteration threshold is necessary to be selected whether the SOM doesn't converge.

Furthermore, the "Mean Distance to the Closest Unit" (MDCU) criteria is the quantization error of the SOM map [15] which is computed after the training process. It calculates the average distance of the sample vectors to the node centroids by which they are represented.

Finally, the SOM provides a topology preserving map [8] from input to output spaces, which includes grid $\mathbb{T}$ and weight vectors V. For SOM training, the weight vector associated with each neuron moves to become the center of a local group of input vectors. The group i is represented by its centroid vector v_i and the local groups are connected via $\mathbb{T}$.

In this work, we enhance the generic SOM technique to model the feature space curvature by plugging the gravitation and fabric of space concepts into the standard SOM, in analogy with Relativity theory, to model the density structure of dataset and still describe the whole original feature space.

B. Multilinear Transformation

The multilinear transformation is a spatial transformation function which is locally linear but the coefficients change across different regions of the space. In general, a multilinear transformation function $\mathcal{M} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ of m variables is called a $m-linear$ map. To represent it visually, we describe in details the bilinear transformation function $\mathcal{B} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, which is the multilinear map of two variables [18].

In Fig.2, on the 2D Cartesian coordinate system, we have a quadrilateral (left image) that we want to transform into a rectangle (right image). To interpolate each point ρ on the arbitrary quad into ρ' on the rectangle, we need to obtain a bilinear map function which describes the entire point space enclosed by the quadrilateral [9].

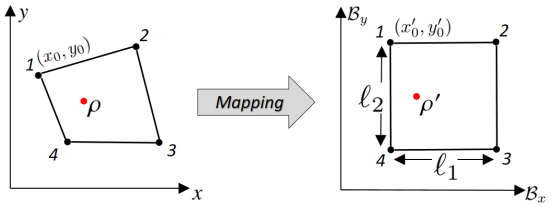


Fig. 2: A bilinear transformation enables us to represent the arbitrary shaped quadrilateral as a rectangle.

We need to compute the function which transfers the quadrilateral into the rectangle. We assume there are bilinear mapping functions $\mathcal{B}_x$ and $\mathcal{B}_y$:

$$\begin{aligned} x &= a_1\mathcal{B}_x(x,y) + a_2\mathcal{B}_y(x,y) + a_3\mathcal{B}_x(x,y)\mathcal{B}_y(x,y) + a_4 \\ y &= a_5\mathcal{B}_x(x,y) + a_6\mathcal{B}_y(x,y) + a_7\mathcal{B}_x(x,y)\mathcal{B}_y(x,y) + a_8 \end{aligned} \tag{8}$$

Where $a_1 ... a_8$ are the transformation parameters. When we have the values of $\mathcal{B}_x$ and $\mathcal{B}_y$ for four lateral points of the rectangle, we can compute the parameters by solving two linear systems, each of four equations with four unknowns.

Let $\vec{d}_1, \vec{d}_2, \vec{d}_3, \vec{d}_4$ be the displacement vectors of the rectangle corners which upper left one is point (x'_0, y'_0) and $\vec{d}_i = (d_i^x, d_i^y) = (x_i - x'_i, y_i - y'_i)$. In [9], they compute the parameters $a_1 ... a_8$ of the bilinear transformation for these displacements as follows:

$$\begin{aligned} a_1 &= \left[(d_2^x - d_1^x)(\ell_2 - y'_0) + y'_0(d_4^x - d_3^x) + \ell_1\ell_2\right]/\ell_1\ell_2 \\ a_2 &= \left[(d_1^x - d_3^x)(\ell_1 + x'_0) + x'_0(d_4^x - d_2^x)\right]/\ell_1\ell_2 \\ a_3 &= \left[d_2^x - d_1^x + d_3^x - d_4^x\right]/\ell_1\ell_2 \\ a_4 &= x'_0 + d_1^x - a_1x'_0 - a_2y'_0 - a_3x'_0y'_0 \\ a_5 &= \left[(d_2^y - d_1^y)(\ell_2 - y'_0) + y'_0(d_4^y - d_3^y)\right]/\ell_1\ell_2 \\ a_6 &= \left[(d_1^y - d_3^y)(\ell_1 + x'_0) + x'_0(d_4^y - d_2^y) + \ell_1\ell_2\right]/\ell_1\ell_2 \\ a_7 &= \left[d_3^y - d_1^y + d_2^y - d_4^y\right]/\ell_1\ell_2 \\ a_8 &= y'_0 + d_1^y - a_5x'_0 - a_6y'_0 - a_7x'_0y'_0 \end{aligned} \tag{9}$$

To obtain the $\mathcal{B}_x$ and $\mathcal{B}_y$, we solve the Equation system (8) to represent the $\mathcal{B}_x$ and $\mathcal{B}_y$ as a function of x and y. These functions are calculated as follows: we firstly solve for $\mathcal{B}_x(x,y)$ from the first equation of the system:

$$\mathcal{B}_x(x,y) = \left(\frac{x - a_4 - a_2\mathcal{B}_y(x,y)}{a_1 + a_3\mathcal{B}_y(x,y)}\right) \tag{10}$$

Then, substituting this into the second Equation of system (8), rationalizing the denominators and combining like powers of $\mathcal{B}_y(x,y)$, we find the following quadratic equation that must be solved to get $\mathcal{B}_y(x,y)$:

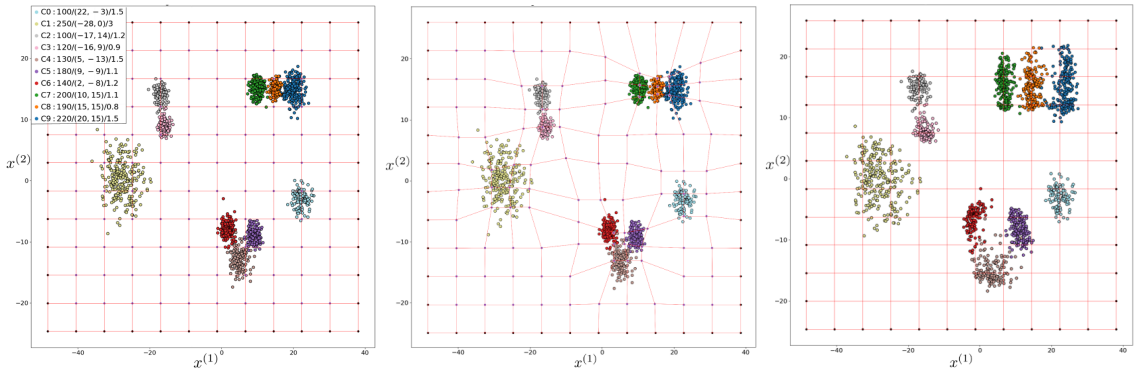
$$A\mathcal{B}_y(x,y)^2 + B\mathcal{B}_y(x,y) + C = 0 \tag{11}$$

Where:

$$\begin{aligned} A &= a_6a_3 - a_7a_2 \\ C &= a_8a_1 - a_5a_4 + a_5x - a_1y \\ B &= a_8a_3 - a_7a_4 + a_6a_1 - a_5a_2 + a_7x - a_3y \end{aligned} \tag{12}$$

Finally, we choose the positive root of the quadratic Equation (11) for $\mathcal{B}_y(x,y)$ as a feasible solution.

As a result, we obtain a bilinear map function which describes the entire point space enclosed by the quadrilateral and lets us displace each point ρ on the arbitrary quad into ρ' on the rectangle continually. This transformation is likewise generalizable to the multi-dimensional space, which is called multilinear transformation to map a hyper-quadrilateral to a hyper-rectangle. Note that the equation system is no longer linear; However, it can be solved quite easily by applying analytical solutions such as Grobner bases [1].



(a) Original Data Distribution. *(b)* FSC Model of (a). *(c)* FSCM Projection of (a).

Fig. 3: Application of FSCM on a multi-density 2D dataset Synt10 containing ten clusters. (a) A scatter plot of clusters with varied densities. The legend shows the $size/\mu(x^{(1)}, y^{(2)})/\sigma$ per cluster, the colors represent the data original labeling and the red lines draw the initial FSF. (b) shows the FSC model that is computed with our FSCM method. Note that the red lines show the deformation of the FSF. (c) scatter plots the data (a) projected by applying our transformation through model (b). As a result, the diversity of the clusters' density scaled appropriately to achieve a better density-based clustering performance.

V. Feature Space Curvature Map

Our Feature Space Curvature Map (FSCM) method involves two main steps that will be formally described in this section. First, we apply our new Gravitational Self-Organization Map (GSOM) method to compute the clusters density structure. It computes the feature space bending with a correct topology preserving map to present the potentially complex multi-density structure of the data. Second, we apply our enhanced multilinear transformation, from the GSOM map to project the data points to the new euclidean feature space with more uniform density distribution. Note that the new feature space is typically considered to be euclidean that classic distances are defined on it. These steps perform a globally nonlinear transformation of the dataset to which any existing density-based clustering algorithm based on euclidean distances can be applied.

A. Feature Space Curvature Modeling

In the first step, we use a topological m-dimensional elastic mesh $\mathbb{T}$ to construct the fabric of feature space as a smooth manifold with a Riemannian metric, while covering the whole feature space. We call this mesh the Feature Space Fabric (FSF) through this paper. Then, we compute the Feature Space Curvature (FSC) model where the concentrations of data points can bend the FSF dramatically while the areas with less concentration just slightly bend it or leave it as locally euclidean. The FSC presents the density structure of the data.

To be able to derive the FSC, we propose a new Gravitational Self-Organization Map (GSOM) algorithm. We train a GSOM network to learn the density topology of the input data X, while still covering the whole feature space. To plugging the concept of the gravity and fabric of space to the generic SOM, we apply four key enhancements on it.

1) Initialization Method: For initializing the neurons in topological space $\mathbb{T}$ instead of random initialization, we use an m-dimensional Regular Rectangular Grid (RRG). Let I_j be the grid interval of the feature $x^{(j)} \in R^n$:

$$I_j = [min(x^{(j)}) - h, max(x^{(j)}) + h] \tag{13}$$

Where h is the marginal coefficient to create extra space around the grid where $h = 0.15$ works well. Let ξ be the number of grid lines for each grid interval I_j where $\xi = \sqrt[m]{\varsigma}$ and $\xi \in \mathbb{Z}^{3+}$. This RRG slices the entire feature space into subdivisions to accurately capture and represent potentially complex local density structures of the data. See Fig.3a, the red lines represent an initial 2D RRG.

2) Rigid Boundary: We modify the SOM algorithm to keep the boundary nodes rigid, during the weight updating of the winner node and its neighbors, to avoid the crumpling and folding the RRG. The node ω is boundary if it belongs to:

$$s = \{\omega \in \Omega | \exists j (v_\omega^{(j)} = min(I_j) \vee v_\omega^{(j)} = max(I_j))\} \tag{14}$$

Where $j \in [1, \ldots, m]$ and $s \subset \Omega$. To keep rigid the boundary nodes, we replace the Equation (4) with:

$$\Delta v_k(t) = \begin{cases} 0 & \text{if } k \in s \\ \alpha(t)\eta(\nu, k, t)[x(t) - v_\nu(t)] & \text{else} \end{cases} \tag{15}$$

See Fig.3a, the black nodes at the sides of RRG represent boundary ones which remain rigid after training, see Fig.3b.

3) Update Procedure: The Law of Universal Gravitation [4] states that every particle of matter in the universe attracts every other particle with a force that is directly proportional to the product of the masses of the particles and inversely proportional to the square of the distance between them. In like manner, we propose a novel Gaussian neighborhood function based on an assumed mutual force acting between all pairs of neurons, when this force is varied for each pair of neurons based on their mass and the euclidean distance between them. The mass of each neuron represents the number of times which this neuron selected as a winner neuron. Let $\mu_i(t)$ be the mass of the i_{th} neuron after step t where $\forall_{i \in \Omega}(\mu_i(0) = 1)$ at the initial state. Then, for each time t, the mass of the winner

neuron $\nu(t)$ is increased by one unit. Therefore, the winner will be the neuron generating more gravitational attraction force to the selected point instead of the closest neuron. For this purpose, we substitute the euclidean distance with gravity force in both Equations (3) and (6) respectively:

$$\nu(t) = \underset{k\in\Omega}{\operatorname{argmax}} \left[\frac{\mu_k(t)}{\|x(t) - v_k(t)\|^2} \right] \tag{16}$$

$$\eta(\nu, k, t) = exp \left[\frac{\mathcal{F}(\nu, k)}{2\sigma(t)^2} \right] \tag{17}$$

Where the mass of the presented input $x(t)$ is equal to one and $\mathcal{F}(\nu, k)$ is the gravity force between node ν and k:

$$\mathcal{F}(\nu, k) = \frac{\mu_\nu(t).\mu_k(t)}{\|r_\nu - r_k\|^2} \tag{18}$$

Fig.3b shows an example where the dense cluster $C8$ (orange), with big size and small standard deviation, curved the FSC dramatically while the sparse one $C1$ (yellow) only slightly bent it.

We empirically figure out the appropriate value for the time constants $\lambda_\alpha = 1000/\alpha_0$ and $\lambda_\sigma = 1000.\sigma_0$ by conducting several experiments. These values lead the GSOM to generate the smooth FSC without crumpling and folding that is necessary for the transformation step of our approach.

There are two principal consequences of this gravitation-based function: (1) the updated weight of neurons depend on both their distances and masses, which guarantees the smoothness of the final FSC, and (2) the GSOM converges faster to stable model when the massive neurons stabilize earlier than light mass neuroses which arrange around them later on, the same way as the stars in a galaxy.

4) Early Stopping Condition: In our approach, we use MDCU to learn convergence mechanism per each iteration rather than calculate after the training process like the standard SOM. We compute the $MDCU(t)$ per each iteration t, as follows:

$$MDCU(t) = \frac{1}{n} \sum_{i=1}^{n} \min_{k\in\Omega} \|x_i - v_k(t)\| \tag{19}$$

It considers the stopping condition that defines the proper GSOM's status as no training samples reduces the overall MDCU. However, a more elaborate trigger is required in practice since the training of the GSOM is stochastic that can be noisy. Therefore, GSOM monitors the overall MDCU for a given number of consecutive epochs in order to try to avoid falling into a local minima. Using this condition, the training process is automatically stopped as soon as it successively sees no reduction in MDCU metric over a given number of epochs $\mathcal{E}$ where $\mathcal{E} = 10$ works well based on our conducted experiments. We compute early stop function $ES(t)$ per each iteration t, as follows:

$$ES(t) = |\{e \in E | MDCU(t-e) - MDCU((t-1)-e)) \leq 0\}| \tag{20}$$

Where $t > \mathcal{E}$ and $E = [0, \ldots, \mathcal{E}]$. The training is stopped as soon as the $ES(t)$ is equal to $\mathcal{E}$.

Note that the FSF modeling divides up the feature space into regular rectilinear cells to capture the density structure of data, see Fig.4a. Although, the computed FSC's cells are not longer rectilinear but are irregular quadrilaterals, see Fig.4b.

As a result, by applying the GSOM on X, we obtain an m-dimensional irregular grid $\mathbb{T}$ in size ς and $V = [v_1, v_2, ..., v_\varsigma]$, $v_i \in R^m$, which accurately represents the density structure of the clusters by preserving the topological structure of data. We summarize the FSC modeling algorithm in Algorithm 1.

Algorithm 1 : Feature Space Curvature modeling

Input: n data points with m features;
σ_0: The initial neighborhood size;
α_0: The initial learning rate;
ξ: The number of grid lines;
$\mathcal{E}$: The validation number of epochs;

Output: The grid $\mathbb{T}$ and the neurons $V = [v_1, v_2, ..., v_\varsigma]$;

1: Initiate RRG map $\mathbb{T}$ of size ξ. (Eq.13)
2: **repeat**
2.1: Select $x_i \in X$ randomly
2.2: Find the winner neuron $\nu(t)$ (Eq.16)
2.3: Update weight of the winner and its neighbors (Eq.17)
2.4: $\mu_{\nu(t)} = \mu_{\nu(t)} + 1$
2.5: Decrease the learning rate $\alpha(t)$ (Eq.5)
2.6: Decrease the neighborhood size $\sigma(t)$ (Eq.7)
3: **until** $ES(t) < \mathcal{E}$ (Eq.20)
4: **return** $\mathbb{T}$, $V = [v_1, v_2, ..., v_\varsigma]$

B. Curvature Map

To apply traditional density-based clustering algorithms which are designed for euclidean spaces, we need to project the data from a non-Euclidean curved space FSC to the new euclidean flat space. Our transformation mechanism reduces the density of high-density regions intensively while it slightly reduces the density of low-density ones.

As euclidean space, it is described by orthogonal dimensions and its discretized version can be represented by a rectangular grid FSF. The resulting FSC is described by a deformed grid where the nodes in the original coordinate space now form irregular quadrilaterals. Our data transformation method "relinearizes" the FSC considering its original grid structure and performing a multilinear projection of each original data point to the new euclidean grid space based on its position in its enclosing irregular quadrilateral of the FSC.

Let $U' = [u'_1, .., u'_\varphi]$ be the initial position of grid $\mathbb{T}$ cells in m-dimensional space, where $\varphi = (\xi - 1)^m$, $u'_i \subset V$ and $|u'_i| = 2^m$; and $L = [\ell_1, .., \ell_m]$ stand for the side sizes of each regular cells where $\ell_\iota = |I_\iota|/(\xi - 1)$. In the same way, $U = [u_1, .., u_\varphi]$ denotes the cells position in FSC when $u_i \to u'_i$.

We begin by computing $\mathcal{B} = [\beta_1, .., \beta_\varphi]$ a set of parametric transformation function between FSC and FSF. For each cell $u_j \in U$, we compute individual transformation function β_j by giving the displacement vectors $\vec{D} = [\vec{d_1}, .., \vec{d}_{2^m}]$ of the cell corners, see section IV-B.

Then, for each $x_i \in X$, we identify the cell $u_\phi \in U$ which x_i is located inside it. To do that, we firstly find the Best Unit Match (BMU) $\nu(x_i)$ by applying Equation (3).

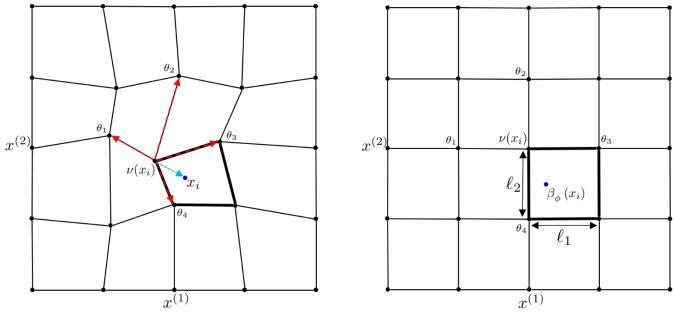
According to the $\mathbb{T}$ topology, the neighborhood $\mathcal{N}(\nu(x_i))$ in R^m is split into 2^m cells. If the direct topological neighbors of the $\nu(x_i)$ expresses by $\Theta = [\theta_1, .., \theta_{2m}]$ where $\theta \subset \Omega$. Then we can specify the u_ϕ with $\nu(x_i)$ and m lattice indicator points in the neighborhood. Therefore, to single out the cell $u_\phi \in U$ which x_i is located inside, we find the most similar lattice point $l_0 \in \Theta$ to x_i:

$$l_0 = \underset{k \in \Theta}{\operatorname{argmax}}(\vec{\mathcal{P}}(x_i).\vec{\mathcal{P}}(\theta_k)) \tag{21}$$

Where $\vec{\mathcal{P}}(\theta_k) = \vec{v}_{\theta_k} - \vec{\nu}(x_i)$ is the position vector of the lattice point θ_k, similarly $\vec{\mathcal{P}}(x_i) = \vec{x_i} - \vec{\nu}(x_i)$ is the position vector of x_i. Then, to distinguish the source cell $u_\phi \in U$, we complete the lattice indicator set as follows:

$$L_{x_i} = \{k \in \Theta : \vec{\mathcal{P}}(l_0).\vec{\mathcal{P}}(\theta_k) \leqslant \vec{\mathcal{P}}(x_i).\vec{\mathcal{P}}(\theta_k)\} \tag{22}$$

After we get cell lattice indicator set L_{x_i}, we could find the cell u_ϕ easily since we keep the grid topology $\mathbb{T}$. Finally, for each $x_i \in X$ we apply the appropriate multilinear transformation $\beta_\phi(x_i)$ to map it into the regular FSF.



(a) Wrapped FSC *(b)* Transformed by FSCM

Fig. 4: Data point transformation between a bent FSC (a) and a regular FSF (b) based on the Multilinear Mapping in $\mathbb{R}^2$.

Fig.4 shows a 2D grid example where its cells are represented by 2^2 corners. Therefore, to map the wrapped FSC (Fig.4a) onto the regular FSF (Fig.4b), we compute the transformation function $\mathcal{B}_j$ for each cell u'_j, where $\vec{d}_{j1}, \vec{d}_{j2}, \vec{d}_{j3}, \vec{d}_{j4}$ are the displacement vectors of cell corners. The $\nu(x_i)$ is the BMU. The red and blue arrows represent the position vectors of lattices and position vector of x_i respectively. The computed indicator set is $L_{x_i} = \{\theta_4, \theta_3\}$ where $l_0 = \theta_4$. As a result, the cell u'_ϕ which x_i is located inside (the thick block) is depicted with this lattice set. Finlay, $\mathcal{B}_\phi(x_i)$ shows the transformer of x_i after the applying mapping function $\mathcal{B}_\phi : u'_\phi \rightarrow u_\phi$. We summarize the Curvature Map algorithm in Algorithm 2.

VI. Application of FSCM in the Real Data

In this section, we have carried out several experiments to show the efficiency and effectiveness of our proposed FSCM method to overcome the weaknesses of existing density-based clustering algorithms in diagnosing all clusters with varying densities.

Algorithm 2 : Curvature Map

Input: n data points with m features;
The grid $\mathbb{T}$ and the neurons $V = [v_1, v_2, ..., v_\varsigma]$;
Output: Homogenized data $X' = (x'_1, ..., x'_n)$, $x'_i \in R^m$;

Compute the regular cell side size $L = [\ell_1, .., \ell_m]$
Compute the initial position of cells $U' = [u'_1, .., u'_\varphi]$
Compute the final position of cells $U = [u_1, .., u_\varphi]$
for ϕ = 1 to φ **do**
  Compute the displacement vectors $\vec{D} = [\vec{d}_1, .., \vec{d}_{2^m}]$
  Compute the map function $\mathcal{B}_\phi : u_\phi \rightarrow u'_\phi$ (Sec. IV-A)
end for
for i = 1 to n **do**
  Find the BMU $\nu(x_i) = \operatorname{argmin}_{k \in \Omega} \|x_i - v_k\|$
  Compute the indicator set L_{x_i} of source cell u_ϕ (Eq.22)
  Find the source cell u_ϕ
  Compute the transformation $x'_i = \beta_\phi(x_i)$
end for
return $X' = (x'_1, ..., x'_n)$

A. Datasets

We used 10 real-world datasets from the UCI Repository[1] and a synthetic datasets Syn10, to validate the capability of our method in homogenizing multi-density dataset. Table.I outlines the basic properties of the datasets.

Syn10 is syntactic "hard distributed" 2-dimensional data that is generated by sampling a mixture of 10 Gaussian distributions $N(\mu, \sigma)$. Therefore, this data set is labeled and can be used to measure the quality of the different clustering approaches. The statistics $(size/\mu(x^{(1)}, y^{(2)})/\sigma)$ of each cluster was shown in Fig.3a. As shown in the density plot Fig.5a, the clusters do not satisfy the clause stated in the Equation (2). As a result, DBSCAN fails to precisely stratify the clusters. For example, the three clusters on the top-right and the other three's at bottom of the density plot represent high-density areas which DBSCAN is unable to separate them by applying an appropriate global ϵ value, see Fig.5b.

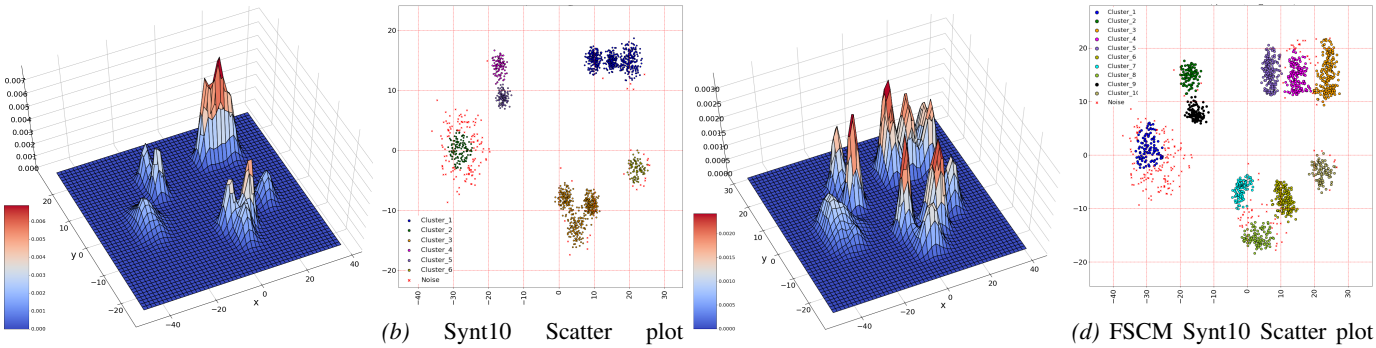
TABLE I: Datasets properties

Dataset	Points (n)	Features (m)	Classes (q)
Segment	2310	19	7
Wine	178	7	3
Wifi	2000	7	4
Iris	150	4	3
Breast	569	30	2
ForestType	523	27	4
Diabetes	768	8	2
Ecoli	336	7	8
Pendig	10992	16	10
ILPD	579	9	2
Syn10	1530	2	10

B. Evaluation Metric

We use the Overall F-measure denoted by "FScore" which is commonly used in the literature to compare the quality of the clustering results. $FScore$ is computed by the harmonic mean

[1]https://archive.ics.uci.edu



(a) Synt10 Histogram *(b)* Synt10 Scatter plot Dbscan(ϵ=1.41,$mPts$=22) *(c)* FSCM Synt10 Histogram *(d)* FSCM Synt10 Scatter plot Dbscan(ϵ=1.62,$mPts$=17)

Fig. 5: Comparison of 3D original histogram (a) of dataset Syn10, previously shown in Fig.3a, and the homogenized density (c) by our FSCM method, subsequently their DBSCAN clustering results (b) and (d). Without applying FSCM, DBSCAN ends up merging the three blobs on the top-right into a single one to be able to identify some cluster point in the sparse blob on the left. FSCM as a preprocessing step to DBSCAN allows to better identify the overall structure of the data.

of precision score and recall score, and the overall $FScore$ is the unweighted average over all clusters:

$$FScore = \frac{1}{q}\sum_{i=1}^{q}\frac{2P_iR_i}{P_i + R_i} \tag{23}$$

Where P_i and R_i are the precision score and the recall score of the cluster c_i respectively. The higher the value of $FScore$ is, the better the clustering performance is.

C. Experiment Setup

Our method, denoted here by FSCM, is compared with: both DScale and ReScale, which consists of first using these scaling algorithms for data preprocessing, and then implementing three state-of-the-art density-based clustering algorithms (DBSCAN, OPTICS and DP) to partition the data.

Before the experiments began, we normalized each dataset by scaling each feature to [0,1] range by min-max normalization. To be able to visualize both our methodology and the empirical result in 2D, we reduced the data dimensionality by applying Principal Component Analysis (PCA). Then, we conducted the whole set of experiments by taking into account the first two Principal Components. We apply the Exhaustive Grid Search method to search the hyper-parameter space for the best validation $FScore$. Table.II specifies the parameters and their search space for each algorithm. The search ranges of both ψ and η are as used by [22] where ψ is the number of intervals to estimate and control the precision of $\hat{f}_\eta$. We implemented the entire set of algorithms used in our experiments in python.

TABLE II: Parameters and their search ranges.

Algorithm	Parameters & Search Range
DBSCAN	$\epsilon \in [0,2]$, $mPts \in \{1,2,..,25\}$
OPTIC	$\epsilon \in [0,2]$, $mPts \in \{1,2,..,25\}$
DP	$\epsilon \in [0,2]$, $k \in \{1,2,..,20\}$
ReScale	$\psi = 100$, $\eta \in \{0.1, 0.2, ..., 0.5\}$
DScale	$\eta \in \{0.1, 0.2, ..., 0.5\}$
FSCM	$\xi \in \{3,..,10\}$, $\alpha_0 \in \{0.005, 0.006, ..., 0.015\}$, $\sigma_0 \in \{1, 1.1, ..., 2.0\}$

D. Clustering Results

The results obtained in our experiments are shown in Table III. The highest obtained $FScore$ values for each dataset-algorithm are highlighted in bold. The second last row reports the average $FScore$. As we can see, our proposed FSCM method persistently surpasses both competitors on all the three clustering algorithms. The highest magnitude of improvement depicted for DBSCAN from 0.59 to 0.77. However, for DP and OPTIC, the performance gap shrinks slightly since they are an advanced version of DBSCAN which are using multiple density thresholds to efficiently detect cluster centers. Still, FSCM increases the clustering performance of DP and OPTIC significantly more than both ReScale and DScale.

The last row of Table III shows the proportional performance winner. It is computed by the number of times a preprocessing method wins the performance divided by total number of datasets for each clustering algorithm. By looking at the first five columns, we see that FSCM-DBSCAN outperforms the other two methods in a large majority, 8 out of 11, of the datasets (in many cases by a large margin); while FSCM-OPTICS and FSCM-DP are the performance winner on 7 and 9 out of 11 datasets, respectively. In the only three datasets where FSCM does not perform best (Breast, ForestType, and Ecoli), its $FScore$ values are very close to the "winner".

In case of Syn10, our FSCM remarkably improves the clustering performance of all existing algorithms, probably because the data is generated from a mixture of Gaussian sources. For example, in Fig.5c we present the histogram of the homogenized Syn10 by applying our FSCM. The mapping yields that both three sets of clusters in the top-right and bottom of original space (corresponding to the two high density reigns, see Fig.5a) are expanded and segregated in the new space with valleys appearing between them. As a result, the densities at the peak of different clusters are closer after this mapping. Thereafter, we could efficiently stratify 10 distinct clusters which correspond to the actual generative model for the data by using DBSCAN, see Fig.5d. However, ReScale fails to increase clustering performances on this dataset since it is just using one-dimensional scaling.

TABLE III: The best $FScore$ for DBSCAN, OPTIC, DP, and their ReScale, DScale, and FSCM versions. For each clustering algorithm, the best performer in each dataset is boldfaced. Orig, PCA, ReS, and DS represent the Original algorithm, Principal Component Analysis, ReScale, and DScale respectively.

	DBSCAN					OPTIC					DP				
Dataset	Orig	PCA	ReS	DS	FSCM	Orig	PCA	ReS	DS	FSCM	Orig	PCA	ReS	DS	FSCM
Segment	0.59	0.62	0.62	0.61	**0.70**	0.69	0.69	0.67	0.70	**0.74**	0.78	0.78	0.77	0.80	**0.85**
Wine	0.64	0.65	0.86	0.80	**0.91**	0.76	0.78	0.84	0.88	**0.91**	0.93	**0.98**	0.95	0.96	0.97
Wifi	0.74	0.76	0.87	0.86	**0.90**	0.79	0.76	0.88	0.85	**0.93**	0.90	0.91	0.92	0.92	**0.95**
Iris	0.85	0.89	0.90	0.93	**0.96**	0.85	0.85	0.84	0.88	**0.94**	0.97	0.97	0.97	0.97	**0.97**
Breast	0.82	0.82	0.95	**0.96**	0.94	0.84	0.79	**0.96**	0.95	0.95	0.97	0.79	0.97	0.97	**0.97**
ForestType	0.27	0.32	**0.51**	0.48	0.49	0.29	0.51	**0.64**	0.52	0.61	0.69	0.75	**0.83**	0.70	0.80
Pima	0.43	0.46	0.48	0.64	**0.72**	0.65	0.65	0.65	0.66	**0.70**	0.62	0.64	0.66	0.67	**0.71**
Ecoli	0.37	0.42	0.40	**0.54**	0.53	0.44	0.50	**0.57**	0.50	0.51	0.48	0.53	0.55	**0.63**	0.61
Pendig	0.70	0.73	0.78	0.74	**0.78**	0.74	0.78	0.78	0.78	**0.81**	0.79	0.79	0.82	0.82	**0.84**
ILPD	0.41	0.42	0.42	0.56	**0.57**	0.47	0.47	0.47	**0.57**	0.54	0.60	0.60	0.63	0.62	**0.63**
Syn10	0.66	0.61	0.58	0.72	**0.89**	0.67	0.65	0.65	0.76	**0.91**	0.69	0.70	0.60	0.78	**0.92**
Average	0.59	0.60	0.67	0.71	**0.77**	0.65	0.68	0.72	0.73	**0.78**	0.77	0.78	0.79	0.81	**0.85**
Winner %	0.0	0.0	9.1	18.1	**72.8**	0.0	0.0	27.3	9.1	**63.6**	0.0	9.1	9.1	9.1	**72.7**

Furthermore, the DScale moderately increased the $FScore$ values of DBSCAN, OPTIC and DP about 0.06, 0.09 and 0.09 unit respectively, since it still depends on a rescaled distance. On the other hand, our FSCM method significantly improved $FScore$ values of DBSCAN, OPTIC, and DP from 0.66, 0.67 and 0.69 to 0.89, 0.92 and 0.92, respectively.

In Fig.6 and Fig.7, we compare the DBSCAN clustering quality on two original WiFi and Breast datasets and their transformation due to FSCM. They show that both transformed datasets are stratified more efficiently than the original data by DBSCAN. For the WiFi dataset, see Fig.6, plot(a) shows the original data with original labeling, including four distinct classes, and the embedded FSC computed by FSCM. As we can see that the $Class\_2$ (red) is dense while the $Class\_1$ (pink) is sparse. As shown in plot (b), DBSCAN just identified three clusters due to the fact that we observed varied densities in data. Consequently, classes 2 and 4 are jointly identified as a single cluster, see the blue cluster in Fig.6b. In contrast, $Class\_2$ becomes sparser using FSCM and we generally observe that the projected data has approximately uniform densities in the populated areas, as shown in plot (c). As a result, DBSCAN identified four distinct clusters by applying single density threshold, as shown in plot (d). Further, our FSCM method significantly improved $FScore$ values of DBSCAN from 0.76 to 0.9, see Table.III.

For the Breast dataset, see Fig.7, plot(a) shows the original data with original labeling, including two distinct classes, and the embedded FSC computed by FSCM. As we can see that the $Class\_1$ (dark blue) is significantly dense while the $Class\_2$ (light blue) is sparse. As shown in plot (b), DBSCAN just identified the dense cluster. Consequently, the majority of the data points belonged to the sparse class are designated as noise, the red $\times$. In contrast, $Class\_1$ becomes sparser using FSCM and we generally observe that the projected data has approximately uniform distribution, as shown in plot (c). As a result, DBSCAN identified two distinct clusters by applying single density threshold, as shown in plot (d). As illustrated in Table.III, our FSCM method significantly improved $FScore$ values of DBSCAN from 0.82 to 0.96.

E. Complexity Analysis

The computational complexity of ReScale, DScale, and FSCM are shown in Table IV. The result shows DScale has higher computational complexity than ReScale since it computes and frequently updates a $n \times n$ distance matrix. While the computational complexity of FSCM is roughly corresponding to the performance of Feature Space Curvature modeling phase which is similar to the standard SOM algorithm. Considering a map of ς neurons and the input data $X \in R^{n \times m}$, then each learning epoch $t \in [1, \ldots, t_f]$ of the GSOM algorithm costs $\mathcal{O}(m\varsigma + n\varsigma^2)$ elementary operations. Therefore, its overall theoretical complexity is $\mathcal{O}(t_f(m\varsigma + n\varsigma^2))$. The size of map ς is the only parameter which could be quite big for the high-dimensional data.

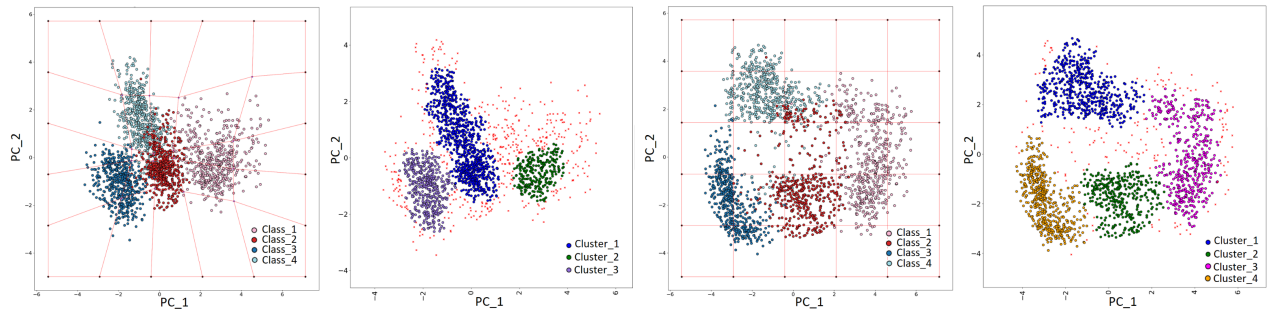
Furthermore, the computational complexity of the majority of existing density-based clustering algorithms is $\mathcal{O}(n^2)$ [23], thus our method doesn't notably affect their final complexities.

TABLE IV: Complexity of ReScale, DScale and FSCM.

Algorithm	Time complexity	Memory complexity
ReScale	$\mathcal{O}(mn\psi)$	$\mathcal{O}(mn + m\psi)$
DScale	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn + n^2)$
FSCM	$\mathcal{O}(t_f(m\varsigma + n\varsigma^2))$	$\mathcal{O}(m\varsigma + n\varsigma^2)$

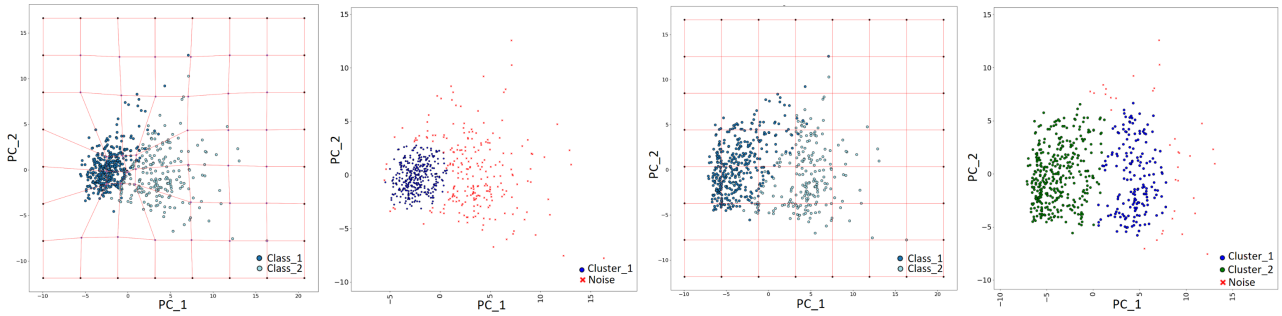
VII. CONCLUSION AND FUTURE WORK

We have presented a new topological Feature Space Curvature Map (FSCM) method to homogenize the density of data to overcome the weakness of density-based clustering algorithms in finding clusters of varied densities. Our FSCM involves two steps: Feature Space Curvature modeling and Curvature Mapping to non-linearly project a multi-dimensional dataset. In analogy to Relativity Theory, we assume an m-dimensional feature space as an elastic Feature Space Fabric (FSF) which is bent when data points are placed in it depending on their density. Therefore, the massive data clouds wrap the FSC intensively while the sparse ones wrap it slightly. We propose



(a) Original labeled data & FSC *(b)* Dbscan(ϵ=0.365,$mPts$=10) *(c)* Projected data by FSCM *(d)* Dbscan(ϵ=0.248,$mPts$=13)

Fig. 6: Application of FSCM-DBSCAN on the Wifi dataset and the original datasets in $\mathbb{R}^2$.



(a) Original labeled data & FSC *(b)* Dbscan(ϵ=1.05,$mPts$=25) *(c)* Projected data by FSCM *(d)* Dbscan(ϵ=1.695,$mPts$=19)

Fig. 7: Application of FSCM-DBSCAN on the Breast dataset and the original datasets in $\mathbb{R}^2$.

a new gravitation-based version of the self-organization map (GSOM) to model the density structure of the data, which is defined on the m-dimensional Regular Rectangular Grid (RRG) of neurons to recognize data density structure. In consequence, GSOM guarantees the smoothness and continuity of the final FSC model and it converges faster to a stable model than standard SOM. Then, we apply a novel multilinear transformation to straighten out the wrapped FSC to reach a new equidistant feature space when the data points are attached to the Feature Space Fabric. Our parametric multilinear transformation approach projects the data points to the new feature space, showing more uniform density among data aggregations than in the original space. As a result, existing density-based clustering algorithms can efficiently identify clusters within this data homogenized by FSCM, as it is clearly demonstrated empirically in this paper. Here FSCM is a general and an efficient preprocessing method that is parametric, assumption-free, and automated that can be applied before clustering analysis.

REFERENCES

[1] Adams, W. W., Adams, W. W., Loustaunau, P., & Adams, W. W. (1994). An introduction to Grobner bases (No. 3). American Mathematical Soc..

[2] Ankerst, M. & Breunig, M. M. (1999). OPTICS: Ordering points to identify the clustering structure. ACM Sigmod record, 28(2), 49-60.

[3] Carly GK Ziegler & Samuel J Allon, (2020). Sars-cov-2 receptor ace2 is an interferon-stimulated gene in human airway epithelial cells and is detected in specific cell subsets across tissues. Cell, 181(5):1016–1035.

[4] Carroll, S.M., (2019). Spacetime and geometry. Cambridge University Press, (pp. 108-112)

[5] Chen, B. (2018). Local contrast as an effective means to robust clustering against varying densities. Machine Learning, 107(8), 1621-1645.

[6] Cui, X. & Guo, L. (2021). GPR-Based Automatic Identification of Root Zones of Influence Using HDBSCAN. Remote. Sens., 13, 1227.

[7] Fahy, C., & Yang, S. (2019). Finding and tracking multi-density clusters in online dynamic data streams. IEEE Transactions on Big Data.

[8] Khatoon, M.(2019). An efficient method to detect communities in social networks using DBSCAN algorithm. ASONAM, 9(1), 1-12.

[9] Konstantopoulos, C. (2015). A parallel algorithm for motion estimation in video coding using the bilinear transformation. Springer 4(1), 1-20.

[10] Liu, P., Zhou, D., & Wu, N. (2007). VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise. 2007 International Conference on Service Systems and Service Management, 1-4.

[11] Lotfi, A. & Moradi, P. (2020). Density peaks clustering based on density backbone and fuzzy neighborhood. Pattern Recognition, 107, 107449.

[12] Mahdavi, K., Mancho, J. L., & Lucas, J. G. (2021). Organization Component Analysis: The method for extracting insights from the shape of cluster. In 2021 IJCNN (pp. 1-10).

[13] Malzer, C., & Baum, M. (2019). HDBSCAN($\tilde{\epsilon}$): An Alternative Cluster Extraction Method for HDBSCAN. ArXiv, abs/1911.02282.

[14] Marquis, E. A. & Chou, P. (2019). On the use of density-based algorithms for the analysis of solute clustering in atom probe tomography data. 18th EnvDeg conference (pp. 2097-2113). Springer.

[15] Mariño, L. M., & de Carvalho, F. D. A. (2020). A new batch SOM algorithm for relational data with weighted medoids. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8).

[16] Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. science, 344(6191), 1492-1496.

[17] Rovere, M. & Seez, C. (2020). CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High-Energy Physics. Frontiers in big Data, 3.

[18] Von, W.C. (2009). Differential forms in mathematical physics. Elsevier.

[19] Wang, S. & Shen, B. (2016). MDBSCAN: Multi-level Density Based Spatial Clustering of Applications with Noise. 11th KMO Conference.

[20] Xiao, N., Li, & Zhou, X. (2019). A Novel Clustering Algorithm based on Directional Propagation of Cluster Labels. In 2019 IJCNN (pp. 1-8).

[21] Zhu, Y. & Ting, K. M.(2016) Density-ratio based clustering for discovering clusters with varying densities. Pattern Recognition, 60, 983–997.

[22] Zhu, Y., Ting, K. M. and Angelova, M. (2018) A distance scaling method to improve density-based clustering. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, 389–400.

[23] Zhu, Y., Ting, K. M., Carman, M. J., & Angelova, M. (2021). CDF Transform-and-Shift: An effective way to deal with datasets of inhomogeneous cluster densities. Pattern Recognition, 117, 107977.