Metricizing the Euclidean Space Toward Desired Distance Relations in Point Clouds

Stefan Rass, *Member, IEEE*, Sandra König, Shahzad Ahmad, and Maksim Goman

***Abstract*— We introduce the concept of an ε-semimetric that satisfies the same axioms as a topological metric, except for an arbitrarily small allowance to violate the triangle inequality. Under this modification, we demonstrate the possibility of taking arbitrary points in space, assigning arbitrary desired distances between them (independent of their geometric location relative to each other, that is, independent of their "features"), and constructing an ε-semimetric that measures exactly the desired distances in the point cloud. This results in a threat to fairness and objectiveness in applications of clustering algorithms: suppose that an adversary subjectively classifies people according to its whim or discriminatory preferences. Upon accusations of unethical behavior, the malicious data processor can plausibly deny these as follows: it designs a distance function (an ε-semimetric) that is (up to a fully controllable numeric "round-off-error" ε) equivalent to a standard distance like the Euclidean. However, this crafted distance will exactly reproduce the (malicious) results and thus confirm them while pretending objectivity and transparency, since only standard and explainable artificial intelligence was used. This demonstration works without any data poisoning. We illustrate the method on randomly chosen points with stochastically independent random classifications assigned to them. Then, we apply standard implementations of k-Means and DBSCAN on the data points, which both exactly reproduce the desired (randomly chosen) classes. We also discuss non-adversarial applications of ε-semimetrics, and corroborate the construction with examples and implementation in Octave.**

***Index Terms*— Adversarial machine learning, trust management, non-repudiation, security, artificial intelligence (AI), data integrity.**

I. Introduction

CLUSTERING algorithms enjoy widespread use to categorize objects based on similarity in terms of features. Objects classified as similar may undergo the same treatment or processing to save resources. If human error and manual labour shall be reduced or even eliminated by letting a machine learning model do a prior classification of an object or subject as a decision aid, questions of fairness and ethical use of artificial intelligence (AI) become relevant [1], [2]. For example, if the subjects of classification are people, AI shall be used to treat all persons objectively and fairly. We give a hypothetical example here to explain the issue but remark that documented cases of unethical treatment of humans by computers exist.[1]

TABLE I

Selected Related Work on Adversarial Clustering

Goal: make the clustering result in pre-specified label assignments.

Work on attacks	Input/training data	Clustering algorithm
[3]–[10], and refs. in Sect. II	poisoned within an ε-bound or otherwise polluted	*unmodified* (uses a standard metric, e.g., Euclidean or other)
our work	*unmodified* (original)	uses a crafted ε-semimetric

Suppose two customers, let us call them A and B, with very similar, but not entirely identical, features show up and ask for a credit loan. The creditor is malicious and has subjective discriminatory preferences for customer B, who would – objectively – not deserve the loan. In contrast, customer A shall not receive it (and would anyway be rejected for the same reasons as customer B, since they are almost the same regarding the relevant features, resp. aspects).

The creditor uses artificial intelligence to pretend transparency, fairness, and objectiveness to disguise the wrong intentions. To this end, it wants to apply some standard clustering algorithm, but since the features are very similar based on the Euclidean distance, B and A would both be rejected. However, the creditor wants to support customer B and hence seeks to forge the results by separating the otherwise similar customers. A standard method to accomplish this (see Table I for related references) is the addition of noise to the data to manipulate its classification. However, if the data is integrity-protected, for example, by a digital signature, such a manipulation would be at least detectable afterward if the creditor is asked to demonstrate (to an independent party) how the results were obtained. If adding noise is precluded, another way of forging would be a suitably designed distance function.

Siamese neural networks [11] are a natural method to construct a distance according to specific application requirements. Still, these constructions amount to training deep networks and may need massive data. Standard means of adversarial machine learning [12] may be applicable, and countermeasures are likewise known. However, the final deci-

Manuscript received 10 May 2023; revised 14 December 2023 and 3 May 2024; accepted 18 June 2024. Date of publication 27 June 2024; date of current version 5 August 2024. This work was supported by the Linz Institute of Technology (LIT) Secure and Correct Systems Laboratory funded by the State of Upper Austria and the LIT under Grant LIT-2019-7-INC-316. The associate editor coordinating the review of this article and approving it for publication was Dr. Daniel Moreira. *(Corresponding author: Stefan Rass.)*

Stefan Rass is with the LIT Secure and Correct Systems Laboratory, Johannes Kepler University Linz, 4040 Linz, Austria, and also with the Institute for Artificial Intelligence and Cybersecurity, Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt, Austria (e-mail: stefan.rass@jku.at).

Sandra König is with Research Driven Solutions Ltd., Dublin, D08 TX29 Ireland (e-mail: sandra.koenig@researchdrivensolutions.ie).

Shahzad Ahmad and Maksim Goman are with the LIT Secure and Correct Systems Laboratory, Johannes Kepler University Linz, 4040 Linz, Austria.

Digital Object Identifier 10.1109/TIFS.2024.3420246

[1] see https://amsalgorithmus.at/en/; retrieved June 26th, 2024.

TABLE II
RELATED WORK ON NON-ADVERSARIAL APPLICATIONS

Prior work	Our contribution
Semi-supervised clustering (e.g., [29]–[31]) and hierarchical, k-Means and other clustering with constraints [32]–[38]	Provisioning of a distance function that respects constraints on data points that must be or must not be in the same cluster
Domain adaptation, e.g., [39]–[42]	Change the accuracy of the clustering if its distance function is designed to "fit" an expected cluster patterns that is (geometrically) also anticipated in other domains
Trustworthiness/Security of clustering/classification, e.g., [43]–[45]	Reliability of results by prevention of posterior manipulation of clustering results by verifiably committing to a fixed metric (e.g., Euclidean), for non-repudiation of clustering results (against deniability)

sions based on such a black box distance function can be challenging to explain, thus putting the objectivity of such decisions in question.

To avoid this situation, the creditor would prefer using a distance function independent of any unknown data or prior training, as the Euclidean distance would be. Only, it cannot apply the Euclidean metric (since the customer's features are too close to merit a distinct treatment under this distance), but it *can* find something that is not "too much different" from Euclidean but will lead to the desired separation and unequal treatment of A and B.

We show how this is possible by developing the concept of an ε-semimetric as a slight generalization of a topological metric.[2] Such a function can act as a distance but requires no training and can be made deterministic.

With this maliciously crafted distance, the clustering can run and confirm the malicious creditor's intended results while appearing trustworthy because:

- the classification was done by a standard clustering algorithm like k-Means or DBSCAN, neither of which is a black-box,
- and the distance function these algorithms were using is a topological metric, and hence "essentially the same" as the Euclidean distance, which is a common default.
- Thus, the final results depend only on feature similarity in explainable terms and may appear plausible.

Our goal in this work is the construction of distance functions that are "almost" topological metrics but can cause algorithms like k-Means [14] or DBSCAN [15] to produce any desired result, while still being explainable, transparent, and seemingly fair. In the above example, the creditor can even arbitrarily decide how A and B should be classified and claim only to follow the algorithm's recommendation. We demonstrate that if the similarity metric is not a priori fixed and verifiably committed before the actual clustering is done, a posterior forgery of the results becomes possible.

Besides adversarial clustering, our work also provides solutions to some challenges in non-adversarial settings, such as clustering under constraints, domain adaptation, and establishing trustworthiness. Table II provides an overview, with a discussion following in Section II.

[2]A preliminary version of this work is available from arxiv.org [13].

It is well known that unfair biases can already exist in the training data [16], [17], [18], in which case the AI will learn the bias and apply it subsequently (cf. [19]). In this work, we can safely suppose that a proper de-biasing has already happened [20] and that the data is hence void of any adversarial additions to it, whether it is noise or full adversarial samples. Instead, we show how the clustering is still manipulable if the configuration of the algorithm, specifically the distance function, is customizable. Table I relates our work to prior work on adversarial clustering, most of which assumes noise on the legitimate inputs (causative attack) or additional adversarial examples (exploratory attack) [6]. Both attack types have a common denominator for modifying the training data or inputting it into the clustering algorithm. The design of norms (and hence distance functions), detectors, or other methods to become robust against noise, was as studied in [21], [22], [23], [24], [25], and [26], as well as in [27] who considered correlations in the data to define clusters.

II. CONTRIBUTION AND FURTHER RELATED WORK

The authors of [7] study the challenging setting of how to modify the clustering, if the feature vectors are not directly accessible, and tackle this issue by letting the attacker construct a surrogate to approximate the hyperparameters, features, machine learning algorithms, and other information missing in the original limited view of the attacker. Their methods likewise inject noise or exploit specific graph properties in spectral clustering methods [28]. Our work deviates from these directions in a twofold way, since it targets the algorithm underlying spectral clustering (which is still a "standard" algorithm like k-Means [28]), and we leave the data entirely untouched, i.e., do not inject any noise to the input data of the clustering (thereby automatically satisfying any tolerable noise bounds, e.g. [8], [9], [10]). This delineates our work from most prior work on attacking clustering algorithms (cf. Table I), all of which share the idea of noise injection on the input data in common to let a conventional clustering method run into incorrect results. The work of [3] refines the attacker's goals to violations of integrity, availability, or privacy. Our work fits into this categorization as an integrity violation attack, in [3] described "as attacks aiming to deflect the grouping for specific samples, while limiting the changes to the original clustering". The modification studied in this work is limited in terms of being restricted to supplying a standard clustering algorithm with a custom metric (only) and not changing anything else. The goals here are similar to that in [5], which introduces the concept of an adversarial set: this is the transformation of a set into a slightly modified set whose classification would be unchanged for a human annotator, but the classification of a well-trained model would become different upon the transformation. Similar to noise, the input dataset is again manipulated, and the clustering algorithm is standard (spectral clustering). This method, however, is also vulnerable to changes of the metric inside the spectral clustering pipeline (briefly reviewed in section VII-B), thus no additional robustness is gained against our scenario by spectral techniques.

Non-Adversarial Applications: In practice, the need to adapt the results of a classification algorithm may arise in case the training accuracy is significantly different from the verification accuracy, or the accuracy of results if a model trained on one domain is applied to data of another domain. Concepts like Siamese networks or pairwise similarity regulation [39] have been developed to mitigate this issue. Our work can provide a "direct solution" in the sense of allowing us to construct a metric that, by design, accomplishes the clustering as desired. It is, however, an open question if this construction lends itself to domain adaptation, too, since the metric is specifically crafted to put given points into desired distances but makes no assertions about the resulting distances between yet unseen data points. Another possible application relates to must-link or must-not-link constraints in semi-supervised learning [37], [38]: the explicit construction of metrics towards separating points that must-not-link in the same cluster, or must-link in the same cluster [29], is a method to incorporate prior information into the clustering. We do exactly this in our experiments in Section VI-A.

A notable non-adversarial instance of the idea to replace the metrics used for clustering was also reported in [46], which designed the metric for separating real from adversarial examples to train generative adversarial networks (GANs). An idea similar to our deterministic addition of noise has also been used to improve the quality of clustering, such as done in [47], where the number of neighbors in increasing distance radii is used to map points into a high-dimensional feature space. Closely related is also the concept of *metric embeddings* [48]. These deal with the problem of mapping a point cloud into another metric space at *new locations*, but with *unchanged pairwise distances* (isometric mapping). Our goals are "reversed" in the sense of leaving the *locations unchanged* but *manipulating* the pairwise *distances*. Moreover, we do not change the data space, whereas metric embeddings such as Borgain's theorem, would lift the points from the Euclidean space to a sequence space, for example.

Many studies of clustering [3], [23], [26], [46], [47], [49], [50] are concerned with noise robustness or poisoning attacks (e.g., [51]), and demonstrate how sensitive the algorithm may become upon insertion of (even a few) adversarial examples. Our work extends these thoughts to not only poisoning the input to the algorithm but also its configuration by replacing the distance functions a posteriori to justify some (desired) behavior. This is particularly problematic in, for example, recommender systems, as was thoroughly studied in [49]. Attacks similar to our setting were also reported as *adversarial backdoors* such as was proposed by [52], but with our modification being not in the algorithm, but merely in its configuration.

Interesting possible countermeasures to our attack technique are methods of posterior cluster validation [53], [54], which may judge the "plausibility" of clusters on different means than our crafted metric. This is indeed conceptually close to the countermeasure that we propose as a prior commitment to the exact configuration of the clustering algorithm, including the choice of distance in particular. Such commitments would be up to independent party verification, and the cited reference offers an additional tool for such third-party verification to be done. Specifically, some robust classification techniques, e.g., [55], work with specially designed metrics that should not be silently replaceable.

Some algorithm implementations (see, e.g., [56]) allow the specification and supply of a dissimilarity matrix, fixing the ij-th element as the desired distance between data point $\mathbf{y}_i$ and data point $\mathbf{y}_j$, not necessarily constraining this dissimilarity matrix to correspond to any topological metric. A famous related result is Schoenberg's criterion [57], which gives conditions on a matrix to correspond to Euclidean distances of points placed in some (possibly high-dimensional) space. Constructive proofs of Schoenberg's criterion take the norm as fixed (to be Euclidean), and the desired distances, and from this determine the placement of points such that they are in consistent locations. Our work uses the same three items, but two different ingredients for the third item as the outcome: we first fix the distances, then place the points in the space, and from there, construct a metric that puts the points into the desired distance from each other (although their locations are fixed).

III. Preparation

Symbols and Notation: The notation so far, and to be continued, lets vectors appear as lower-case bold printed letters, while scalar variables and functions are lower-case Latin letters. Greek letters are used for constants while admitting that some constants may depend on other constants (but are hence themselves not variable). Upper case letters denote sets, and when bold-printed mean matrices. Finally, calligraphic letters denote probability distributions. We hereafter call the set $\mathbb{R}$ or the higher-dimensional vector space $\mathbb{R}^\ell$ the Euclidean space (for short), *not* implying that it is endowed with the Euclidean metric.

We let the symbol $B_{2,\varepsilon} = \left\{\mathbf{z} \in \mathbb{R}^n \ : \ \|\mathbf{z}-\mathbf{y}\|_2 < \varepsilon\right\}$ be the Euclidean neighborhood of a point $\mathbf{y} \in \mathbb{R}^n$, being the open ball of radius $\varepsilon > 0$. More generally, for a norm induced by a positive definite quadratic form $q : \mathbb{R}^n \to \mathbb{R}$, we will write $B_{q,\varepsilon}$ for the radius ε-ball, w.r.t. the norm $\|\mathbf{x}\|_q := \sqrt{q(\mathbf{x})}$. Since later, we will make use of random points chosen uniformly from a neighborhood, we will let the symbol $\mathcal{U}(B)$ for the continuous uniform distribution supported on a set B of the Euclidean vector space. Further symbols will be introduced along with the explanations. Table III provides an overview.

In the following, we will make frequent use of distances δ_{ij} between two points $\mathbf{y}_i, \mathbf{y}_j$. We will write the distance δ_{ij} synonymously as δ_k, with the convention that the integer k one-to-one corresponds to the pair ij, written as $k \simeq ij$ in a slight abuse of notation. The reason is that we will several times need to denote an arbitrary pair or enumerate all pairs, and use a "single" index k for this purpose, while at other times, we may speak about a specific pair, in which case we use the (equivalent) double index ij.

A. Definitions

We will hereafter use different properties for a distance measure, and therefore, define a metric together with a tabular

TABLE III
LIST OF SYMBOLS

Symbol	Meaning
m, ℓ	number ($m > 1$) and dimension ($\ell > 1$) of data points
$\mathbf{y_1}, \ldots, \mathbf{y}_m \in \mathbb{R}^\ell$	the data points to be placed at desired pairwise distances
$\delta_{ij} > 0$	desired distance between $\mathbf{y}_i$ and $\mathbf{y}_j$; we take $\delta_{ij} = \delta_{ji}$
$\epsilon > 0$	tolerance level (violation of triangle inequality and noise magnitude)
$\|\cdot\|_Q$	norm based on a quadratic form with matrix $\mathbf{Q}$
$B_{2,\epsilon}$, $B_{Q,\epsilon}$	open ball of radius $< \epsilon$ w.r.t. Euclidean norm or $\|\cdot\|_Q$
$\mathbf{y}'_i \in \mathbb{R}^h$	canonic embedding of $\mathbf{y_i} \in \mathbb{R}^\ell$ inside $\mathbb{R}^h$ with $h = \binom{m}{2}$
$\mathbf{z}_{i,j} \in \mathbb{R}^h$	j-th neighbor of $\mathbf{y}_i$ within $B_{2,\epsilon}(\mathbf{y}_i)$; cf. Fig. 1
$X \sim \mathcal{U}(B)$	random var. X, uniformly distributed over B

TABLE IV
DEFINITION OF A METRIC AND GENERALIZATIONS THEREOF

	(I)	(P)	(S)	(T)
premetric	no	yes	no	no
metric	yes	yes	yes	yes
quasi-metric	yes	yes	no	yes
semimetric	yes	yes	yes	no
ε-semimetric	yes	yes	yes	approximately; see (1)

overview of generalizations. We stress that the terms used here are not all standardized in the literature, and some authors may prefer slight variations [58], [59].

Definition 1 (Metric, (ε-)Semimetric, and Generalizations): Let V be a vector space. We call a function $d : V \times V \to \mathbb{R}$ a metric, semimetric, premetric *or* quasimetric, *depending on which conditions of the following list is satisfied:*

(I) *Identity of indiscernibles: if $d(x, y) = 0$ then $x = y$.*
(P) *Positivity: $\forall x \neq y : d(x, y) \geq 0$ and $d(x, x) = 0$.*
(S) *Symmetry: $\forall x, y : d(x, y) = d(y, x)$.*
(T) *Triangle inequality $d(x, y) \leq d(x, z) + d(z, y)$.*

The following table specifies a "yes" if a condition is required for the respective term, and a "no" if the condition is not demanded (although it may hold).

For any given $\varepsilon > 0$, we call d an ε-semimetric, *if it satisfies the triangle inequality up to an additive error $\leq \varepsilon$ on the right-hand side, i.e., an ε-semimetric, satisfies (I), (P), (S) and*

$$\forall x, y, z \in V : d(x, y) \leq d(x, z) + d(z, y) + \varepsilon. \quad (1)$$

By our definition, a 0-semimetric is a metric, or otherwise saying that if the ε-semimetric d is actually "zero-semi", then it is a metric.

B. Problem Statement

Our goal is to endow the space $\mathbb{R}^\ell$ with a metric d, such that a set $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ of points comes to lie in $\mathbb{R}^\ell$ at pairwise distances $d(\mathbf{y}_i, \mathbf{y}_j) = \delta_{ij}$ for all $\mathbf{y}_i, \mathbf{y}_j \in Y$, where we can choose the distance value δ_{ij} freely and in advance (but consistent to be a reasonable distance, i.e., non-negative and symmetric), i.e., we require that

1) d is a topological metric,
2) $\delta_{ij} = \delta_{ji} > 0$ for all $i \neq j$, but not further constrained.

It is easy to see that finding such a d will not generally be possible since once the locations of the data points are fixed, their Euclidean distances are determined. Any other norm-induced metric would necessarily be bounded within constant multiples of these distances. However, by scaling down to local neighborhoods, we can accomplish pairwise closeness/distance relations, in the sense of, for example, putting a point $\mathbf{y}_i$ closer to $\mathbf{y}_j$ than to $\mathbf{y}_k$, although the Euclidean distances would induce different neighborship.

Specifically, we cannot hope that our metric d will satisfy exactly $d(\mathbf{y}_i, \mathbf{y}_j) = \delta_{ij}$, but we can design it to satisfy almost this equation, up to a multiplicative constant that depends on the set Y, i.e., we can accomplish $d(\mathbf{y}_i, \mathbf{y}_j) = \alpha \cdot \delta_{ij}$ for all i, j, and some value $\alpha > 0$ that depends only on Y (as a whole), but does not change for different i, j's.

This is enough for manipulation of a clustering: if the algorithm needs to choose a class label based on "closeness", the situation that $\mathbf{y}_i$ is closer to $\mathbf{y}_j$ than it is to $\mathbf{y}_k$ is reflected by the distance relation $\delta_{ij} < \delta_{ik}$. This inequality remains intact if we multiply by any constant $\alpha > 0$, so the "closest" cluster to $\mathbf{y}_i$ remains unchanged if we manage to put it at a distance that is proportional (not necessarily equal) to what we desire. This will be Theorem 2.

IV. RESULTS

Given the set $Y \subset \mathbb{R}^\ell$ of points and having chosen the distances δ_{ij} at which we want $\mathbf{y}_i$ and $\mathbf{y}_j$ to be separated for all $i < j$, we now proceed by showing how we can accomplish this as long as $m = |Y| \in O(\sqrt{\ell})$ holds. If the cluster centers are known or definable a priori, such as in algorithms like k-Means, we can place the cluster centers wherever we like and at any distance we desire. This will cause the points in the topological vicinity to be assigned to the nearest cluster center, just as we like it (Section VI-B). For other algorithms like DBSCAN, we can directly choose the cluster in which points shall be put and set the distance metric accordingly to produce this outcome (Section VI-C).

A. Embedding Points at Desired Distances

We start with a simple technical result that asserts a set of points drawn at random is almost surely linearly independent. The crucial point is that all they are sampled from distributions that are absolutely continuous.

Lemma 1: Let $n > 1$ be fixed and let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^n$ be independently, but not necessarily identically sampled from distributions that are all absolutely continuous w.r.t. the Lebesgue measure[3] on $\mathbb{R}^n$. Put the vectors as columns into an $(n \times n)$-matrix $\mathbf{M}$.

Then, $\mathbf{M}$ has almost surely full rank, i.e., $\Pr(\mathit{rank}(\mathbf{M}) = n) = 1$.

We prove Lemma 1 in Appendix A. Armed with this as a tool to assure that we can select linearly independent vectors in $\mathbb{R}^\ell$ with high probability, we can state our first main result.

[3]Measures that are not absolutely continuous in this sense are, for example, the Cantor distribution, or degenerate point masses defined on $\mathbb{R}$. Conversely, all "standard" distributions like continuous uniform, Gaussian noise, Laplace- and all distributions with a continuous density function will satisfy this requirement. Those are the only ones of interest here.

Theorem 1: Let $\ell > 1$ and choose a finite set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^\ell$ of arbitrary but linearly independent points. For every $1 \leq k \leq n$, fix a value $\delta_k > 0$ associated with $\mathbf{x}_k$. Then, there is a norm $\|\cdot\|_q$ that satisfies $\|\mathbf{x}_k\|_q = \delta_k$ for all $k = 1, 2, \ldots, n$.

Proof: Since $|D| < \infty$, we can enumerate the elements of D as $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in any fixed order (e.g., lexicographic or other). We show that we can define a quadratic form $q : \mathbb{R}^\ell \to \mathbb{R}$ that takes any desired value δ_k^2 at point $\mathbf{x}_k$ for all $k = 1, 2, \ldots, n$.

For the moment, take k as fixed but arbitrary, and let the matrix $\mathbf{B}_k$ besuch that its null-space is the subspace of $\mathbb{R}^\ell$ spanned by $\{\mathbf{x}_j : j \neq k\}$ (such a matrix is easy to find using singular value decomposition). Define $\tilde{\mathbf{A}}_k := \mathbf{B}_k^\top \mathbf{B}_k$, then $\tilde{\mathbf{A}}_k$ is positive semi-definite. Moreover, since all $\mathbf{x}_j$ with $j \neq k$ are linearly independent of $\mathbf{x}_k$ by hypothesis, we have $\mathbf{x}_k$ falling outside the null-space of $\mathbf{B}_k$, so that $\mathbf{x}_k^\top \tilde{\mathbf{A}}_k \mathbf{x}_k > 0$, and $\mathbf{x}_j^\top \tilde{\mathbf{A}}_k \mathbf{x}_j = 0$ for all $j \neq k$. Finally, we will use the scaled matrix $\mathbf{A}_k := \frac{\delta_k^2}{\mathbf{x}_k^\top \tilde{\mathbf{A}}_k \mathbf{x}_k} \cdot \tilde{\mathbf{A}}_k$ and define the quadratic form $q_k(\mathbf{x}) := \mathbf{x}^\top \mathbf{A}_k \mathbf{x}$, which now satisfies

$$q_k(\mathbf{x}) = \begin{cases} \delta_k^2 & \text{for } \mathbf{x} = \mathbf{x}_k; \\ 0 & \text{for } \mathbf{x} = \mathbf{x}_j,\ j \neq k. \end{cases} \tag{2}$$

Repeating this construction for all $k = 1, 2, \ldots, n$, we can define a "Lagrange-interpolation-like" quadratic form with the matrix $\mathbf{A} := \sum_{k=1}^n \mathbf{A}_k$, which is $q(\mathbf{x}) := q_1(\mathbf{x}) + q_2(\mathbf{x}) + \ldots q_n(\mathbf{x})$. From this, we can define the norm $\|\mathbf{x}\|_q := \sqrt{q(\mathbf{x})}$.

By construction, this norm achieves our goal, since for every $\mathbf{x}_k$, we get $\sqrt{q(\mathbf{x}_k)} = \sqrt{\delta_k^2} = \delta_k$. □

We seek to apply Theorem 1 to points $\mathbf{x}_k$ that arise as pairwise differences between a set of given data points $\mathbf{y}_1, \ldots, \mathbf{y}_m$ whose classification we want to forge. However, we cannot directly define $\mathbf{x}_k := \mathbf{y}_i - \mathbf{y}_j$ as a sequence of pairwise differences, since these will (unless $m = 2$) never be linearly independent. Moreover, the number of pairwise differences between m data points in total is $\binom{m}{2}$, which needs to be $\leq \ell$ necessarily, as any larger set cannot be linearly independent within $\mathbb{R}^\ell$. This puts a limit of at most

$$m \leq \frac{1}{2}(1 + \sqrt{8\ell + 1}) \tag{3}$$

data points that we can work with. Even under this limit, we will still need to add random distortions to the points for linear independence, but Lemma 1 tells us that the setting $\mathbf{x}_k := \mathbf{y}_i - \mathbf{y}_j + \varepsilon_{ij}$ for a random noise ε_{ij} (with an absolutely continuous probability distribution) will almost surely give the desired independence, for all $1 \leq i, j \leq m$ and hence $k = 1, 2, \ldots, \binom{m}{2}$. Under this setting, we arrive *almost* at a metric, since $\|\mathbf{x}_k\|_q = \|\mathbf{y}_i - \mathbf{y}_j + \varepsilon_{ij}\|_q$ differs from a metric only in the additive ε_{ij}-error term. Using the triangle inequality, we see that $\|\mathbf{y}_i - \mathbf{y}_j + \varepsilon_{ij}\|_q \leq \|\mathbf{y}_i - \mathbf{y}_j\|_q + \|\varepsilon_{ij}\|_q = d_q(\mathbf{y}_i, \mathbf{y}_j) + \|\varepsilon_{ij}\|_q$, where d_q is the metric that the norm $\|\cdot\|_q$ canonically induces. The additive error is what comes in new, indicating that we may need to allow some limited violation of the triangle inequality (motivating our introduction of an ε-semimetric). We will make this intuition rigorous in the next section.

Informally, for a maximum number of points as given by (3) and with random noise added, Theorem 1 supports the following statement.

> Inside $\mathbb{R}^\ell$, we can fix any set of $O(\sqrt{\ell})$ many points almost anywhere (up to random displacement for linear independence of the pairwise different vectors), and define a metric that puts these points into mutual distances that we can choose irrespectively of the geometric location of the points.

Theorem 1 thus lets us put a "relative" number of points (relative to the dimension of the space) at absolute (chosen arbitrarily) distances. We now show how to twist this around into the possibility of placing an absolute (arbitrarily chosen) number of points such that we can still find a norm that gives us distances relative, in the sense of proportional, to the predefined distances.

B. Dropping the Constraint on m

We transfer the construction into a larger dimensional space to escape the dimensionality bound that limits the number of points whose pairwise distance vectors we can have as linearly independent. We will map each point to a higher-dimensional pendant that serves to compute the distance to all neighbors. This construction leads to the following result:

Theorem 2: Let a set of points $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathbb{R}^\ell$ with $m > 1$ and $\ell > 1$ be given and let $h = \binom{m}{2}$. To each pair of distinct points $\mathbf{y}_i, \mathbf{y}_j$, assign a positive number δ_{ij}, subject to the constraint that $\delta_{ij} = \delta_{ji} > 0$ for $i \neq j$. Fix an $\varepsilon > 0$ such that $B_{2,\varepsilon}(\mathbf{y}_i) \cap B_{2,\varepsilon}(\mathbf{y}_j) = \emptyset$ for all $i \neq j$.

Then, with probability 1, we can endow $\mathbb{R}^h$ with a norm $\|\cdot\|_Q$ with the following properties:

(a) *for every triple i, j, k, there are points $\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k$ in a neighborhood $\|\mathbf{y}_i - \mathbf{z}_i\|_2 < \varepsilon$, $\|\mathbf{y}_j - \mathbf{z}_j\|_2 < \varepsilon$ and $\|\mathbf{y}_k - \mathbf{z}_k\|_2 < \varepsilon$, such that the distance relation $\delta_{ij} \lessgtr \delta_{ik}$ holds if and only if $\|\mathbf{z}_i - \mathbf{z}_j\|_Q \lessgtr \|\mathbf{z}_i - \mathbf{z}_k\|_Q$. The symbol $\lessgtr$ will practically become an explicit $<$, $=$ or $>$, depending on the left- and right-sides.*

(b) *$B_{Q,\varepsilon}(\mathbf{y}) := \{\mathbf{z} \in \mathbb{R}^h : \|\mathbf{y} - \mathbf{z}\|_Q < \varepsilon\} \subseteq B_{2,\varepsilon}(\mathbf{y})$ for all $\mathbf{y} \in \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$, i.e., disjoint Euclidean neighborhoods remain disjoint in the topology that the norm $\|\cdot\|_Q$ induces.*

We provide the full proof of Theorem 2 in Appendix B, and confine ourselves to an outline here:

Theorem 2 is proved by an application of Theorem 1, but we cannot do so on a large number m of data points $\mathbf{y}_i \in \mathbb{R}^\ell$, since their pairwise difference vectors will not be linearly independent. Since there will be $h = \binom{m}{2}$ difference vectors, we first map our data points into a space of dimension h giving the canonic embeddings $\mathbf{y}_i' \in \mathbb{R}^h$, where enough linearly independent vectors can exist. Then, we pseudorandomly construct slightly displaced neighbors $\mathbf{z}_{i,s} \in \mathbb{R}^h$ for each point $\mathbf{y}_i'$, between which the difference vectors *will* be linearly independent. Figure 1 illustrates the idea. On the so-constructed set of neighbors, we can finally apply Theorem 1 to get a norm measuring the distances in the desired way. The neighborhood of each $\mathbf{z}_i$ will contain the given data point $\mathbf{y}_i$

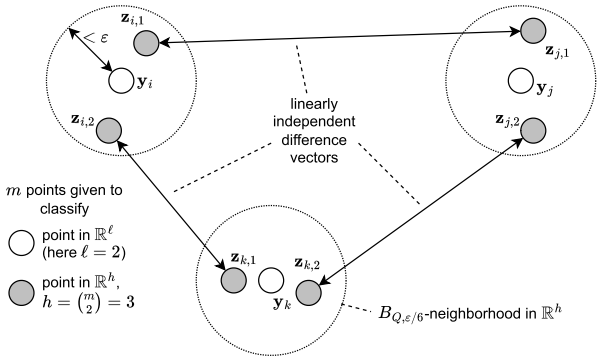

Fig. 1. Linear independence by (stochastically) independent neighbor choices.

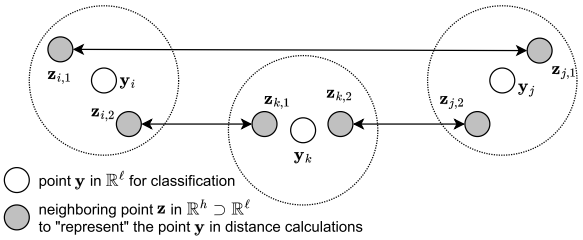

Fig. 2. Violation of the triangle inequality: the triple $\mathbf{y}_i, \mathbf{y}_k, \mathbf{y}_j$ would satisfy the triangle inequality on the distances between them. However, the distances between $(\mathbf{z}_{i,2}, \mathbf{z}_{k,1})$ and between $(\mathbf{z}_{k,2}, \mathbf{z}_{j,2})$ add up to a value less than the direct distance from $\mathbf{z}_{i,1}$ to $\mathbf{z}_{j,1}$. Hence, the triangle inequality cannot generally hold for distances between $\mathbf{y}_i, \mathbf{y}_j, \mathbf{y}_k$ evaluated on the $\mathbf{z}$-neighbors as done by the ε-semimetric $\tilde{d}$, since the neighbors are determined under other constraints than this inequality.

(also $\mathbf{y}'_i$), and be in the *desired* distance *relation* to all other points by part (a) of the above theorem. This is all we need for clustering, based on proximity relations, to come up arbitrarily as we wish.

Intuitively and informally, we can rephrase Theorem 2 as

> Any number of given data points can be placed within disjoint neighborhoods of (virtual cluster center) points in a higher-dimensional space, whose distances we can freely control (up to a common proportionality factor).

Roughly speaking, we can view Theorem 2 and Theorem 1 as two different answers to the challenge of placing points in space at certain pairwise distances: Theorem 1 lets us embed a *relative* number (w.r.t. the dimension of the space) of points at desired *absolute* distances. Conversely, Theorem 2 lets us embed an *absolute* number (in the sense of being independent of the dimension) of points into a (larger) space, such that *relative* proximities (in the sense of proportional distances in the larger space) can be accomplished.

V. ε-Semimetrics to Manipulate Distances

We can carry the ideas further by avoiding the explicit work in the high-dimensional space, and instead endow the original data space with an ε-semimetric that returns the desired distances. The necessity of using only an ε-semimetric instead of a full metric is because we cannot alter a norm-induced topology in $\mathbb{R}^\ell$ to an arbitrary extent, which is precluded by the equivalence of all norms on $\mathbb{R}^\ell$, especially any norm that we construct would be equivalent to the Euclidean norm. However, we can bypass this natural limit by a simple trick: we define a function $d : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ that does the following to compute the value $\tilde{d}(\mathbf{x}, \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^\ell$:

1) It takes the input points $\mathbf{x}, \mathbf{y}$ and adds noise to map them into the high-dimensional space $\mathbb{R}^h$. The noise will be such that it "depends" on both, $\mathbf{x}$ and $\mathbf{y}$, intending to create linearly independent noisy $\mathbf{z}$-substitutes, which we need for the previous construction (cf. Figure 1). Let us call the resulting points $\mathbf{z}_x$ and $\mathbf{z}_y$.
2) It then evaluates the distance $\|\mathbf{z}_x - \mathbf{z}_y\|_Q$ inside $\mathbb{R}^h$, with the norm $\|\cdot\|_Q$ as given by Theorem 2, and returns this value as the value of $\tilde{d}(\mathbf{x}, \mathbf{y})$.

We stress that the added noise is only used "internally" by the function $\tilde{d}$, but not brought onto the original data that is to be clustered. The random distortion is only required to achieve linear independence.

The result obtained by materializing this plan is twofold: we can show that $\tilde{d}$ is an ε-semimetric, for any $\varepsilon > 0$ that we fix in advance, and second, we can let the noise even be deterministic. This is important in its own right since it means that we do not add information to the pair $(\mathbf{x}, \mathbf{y})$ when evaluating their distance. In other words, a clustering building upon our ε-semimetric will judge $\mathbf{x}$ against $\mathbf{y}$ only on grounds of their features, but nothing else.

Making the above outline rigorous lets us prove (in Appendix C) the following result:

Theorem 3: Let a finite set of points $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \subset \mathbb{R}^\ell$ with $\ell \geq 1$ be given and let $h = \binom{m}{2}$. To each pair of distinct points $\mathbf{y}_i, \mathbf{y}_j$, assign a positive number $\delta_{ij} \leq \|\mathbf{y}_i - \mathbf{y}_j\|_2$, only further constrained to satisfy $\delta_{ij} = \delta_{ji} > 0$ for $i \neq j$. Fix any $\varepsilon > 0$.

Then, we can construct a deterministic function $\tilde{d} : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ that is an ε-semimetric and satisfies $\tilde{d}(\mathbf{y}_i, \mathbf{y}_j) = \delta_{ij}$ for all distinct pairs $\mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^\ell$.

We remark that this result, unlike Theorem 2, is not probabilistic, as it asserts the existence of $\tilde{d}$ for sure (not only with probability 1). It is worth remarking that $\tilde{d}$ cannot in general satisfy the triangle inequality with $\varepsilon = 0$, as a counter-example situation shown in Figure 2 shows.

VI. Experimental Demonstration

We implemented the construction of Theorem 2 in Octave [60] as a script that does the following:[4]

- We randomly drew points in $\mathbb{R}^\ell$ from a standard Gaussian distribution (with zero mean and unit variance), corresponding to hypothetical data points with (only) ℓ features. For the following experiment, we had $m = 10$ and $\ell = 2$, to avoid overloaded plots of point clouds and lengthy tables.
- We iterated pairwise through the $h = \binom{m}{2}$ distinct pairs of data points, assigning another random yet positive distance uniformly chosen as $\delta \sim \mathcal{U}[1, 3]$.
- Then, we constructed an ε-semimetric as described in the proofs of Theorems 1, 2, and 3, and verified whether the

[4]The code for all scripts is available from https://github.com/stefan-rass/clustering-manipulations.

desired distances came up as we randomly chose them. Table VII in Appendix D shows the scaled distances versus the values of the quadratic form q (from the proof of Theorem 1), thus experimentally confirming that the distances come up as we wanted them. The embedding into $\mathbb{R}^h = \mathbb{R}^{45}$ is done by adding a random noise vector with 43 dimensions, scaled in magnitude by a factor of $\varepsilon/\sqrt{h}$ with $\varepsilon = 0.1$, to accomplish an Euclidean distance of the $\mathbf{z}$-neighbor to the point $\mathbf{y}$ within $\leq \varepsilon$; cf. Fig. 1.

A. Attacking Clustering: General Outline

An adversarial use of Theorem 2 to forge a clustering may then proceed along steps described as workflow 1. We implemented this in Octave [60], with the following basic setup: the number m and dimension ℓ of points can be chosen arbitrarily at the beginning. The program proceeds by computing all pairwise difference vectors, and assigning random lengths $\delta_{ij} \sim \mathcal{U}[1, 3]$ to them as target distances. It then proceeds by enlarging the dimension from ℓ to h and padding the additional coordinates with yet more pseudo-random values, drawn from a Gaussian distribution such that the 5σ-range falls within an interval of $(-\varepsilon, +\varepsilon)$, with ε set to $\varepsilon = 0.1$ at the beginning (but changeable). Lemma 1 is then verified by computing the rank of the resulting matrix $\mathbf{M}$ and checking it to be maximal.

After that, the code constructs the quadratic forms using singular value decomposition, adds up the resulting matrices, and scales them down by the largest eigenvalue of the total. For a final check, we computed the distances according to the metric Q as defined by (7) and verified the values against the (scaled) random distances $\delta_{ij}/\lambda_{\max}$, with δ_{ij} as chosen at the beginning. The equality of the desired and computed distances was satisfied, and this observation was repeatable. This verified the correctness of the construction and both, Theorem 1 and Theorem 2, experimentally. Enlarging the dimensions or number of points, however, quickly shows that the construction scales badly since the dimension of the larger space grows quadratically with the number of points. Thus, the construction is practically feasible at best for a small number of points and, hence, clusters.

A practical difficulty comes up with the dependence of the proportionality factor α line 15 of the workflow, since this value depends on both, ε and the set of chosen neighbors $\mathbf{z}$ around the given data points (lines 6…11). That is, we cannot take the distances among the $\mathbf{z}$-vectors independently of ε, since these distances will scale by a factor that indirectly depends on ε and is outside our full control. In that sense, the parameterization of the algorithm needs care. We will come back to the parameterization in Section VI-B and Section VI-C when we describe how we chose the parameters to drive the algorithms towards the desired results. Since this is a one-to-one association of virtual cluster centers to points, one may need to parameterize the algorithms accordingly, or put the points $\mathbf{z}_{i,j}$ into close neighborhood of one another (for distinct i) or far away, such that the algorithm returns the desired results. We will demonstrate this for DBSCAN and k-Means.

Remark 1: The pseudorandom displacements that we require to define the data point's neighbors in workflow 1 can in practical instances (which includes our implementation), come from a conventional pseudorandom number generator (PRNG) that produces Gaussian or uniformly distributed values. This PRNG is seeded *with an initial value that, for the input pair* $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2\ell}$ *is different from the seed for the (reversed) pair* $(\mathbf{y}, \mathbf{x})$*, which will be the function* g *that algorithm 1 takes as input. Theorem 3 even defines a deterministic function* f *to create perturbations that make the neighbors provably linearly independent. Practically, however, a standard PRNG will be sufficient. This comes at the "price" of the high-dimensional neighbors* $\mathbf{z}'_i, \mathbf{z}'_j$ *for* $\mathbf{y}'_i, \mathbf{y}'_j$ *(see Figure 2) to be linearly independent only with probability 1.*

The experimental verification of Theorem 3 is done along using its constructed ε-semimetric to manipulate standard clustering algorithms. The next sections are devoted to a demonstration of this. In addition, the verification of Theorem 3 went successful along the same lines as outlined above for Theorem 1 and Theorem 2, with the additional verification of the triangle inequality to hold up to the additive ε-error. This was successfully verified in another Octave script.

B. Manipulating k-Means

We tested the construction towards manipulating the well-known k-Means clustering. The experiment followed these steps, all implemented in an Octave script:

1) Generation of a set of m points $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathbb{R}^\ell$ uniformly at random (as in Section VI-A).
2) Run through the list $\mathbf{y}_1, \ldots, \mathbf{y}_m$, and to each $\mathbf{y}_i$, assign a random but fixed class$(\mathbf{y}_i) \in \mathcal{U}(\{1, 2, 3\})$, *irrespectively* of any features of $\mathbf{y}_i$ (i.e., independent of its location relative to other points).
   Figure 3 shows an example of the inputs to the algorithm as constructed in this step, already assigning the desired class (here chosen at random). To visualize the input, the plot of the point cloud uses different symbols to represent the desired classes. Visually this confirms that the classes, since chosen at random, have nothing to do with any geometric proximity (at least if it were the Euclidean distance).
3) To assign the distances between points of the same cluster, versus distances between points in distinct clusters, we determined the closest pair of points among the set Y at distance $\delta_0 = \min_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|_2$. We likewise computed the largest separation as $\delta_1 = \max_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|_2$. Based on δ_0, δ_1, we defined the "small distance" to be $\delta_{\text{small}} = \frac{1}{200} \cdot \delta_0$, and a "large distance" to be $\delta_{\text{large}} = 200 \cdot \delta_1$.
4) Iterating over all pairs $\mathbf{y}_i, \mathbf{y}_j$ (with $i \neq j$), we assigned the desired distances between them as follows:

$$\delta_{ij} = \begin{cases} \delta_{\text{small}} & \text{if class}(\mathbf{y}_i) = \text{class}(\mathbf{y}_j), \\ \delta_{\text{large}} & \text{if class}(\mathbf{y}_i) \neq \text{class}(\mathbf{y}_j) \end{cases} \tag{4}$$

   The inputs for workflow 1 are then the data points $\{\mathbf{y}_i\}_{i=1}^n$ and the set of distances $\{\delta_{ij}\}_{1 \leq i < j \leq n}$ according to (4). The further inputs are $\varepsilon \leftarrow \delta_{\text{small}}$ and PRNG() was Octave's built-in pseudorandom number generator

Workflow 1 Forging a Clustering With Theorem 2

Require:

- data points $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m \in \mathbb{R}^\ell$
- arbitrary values $0 < \delta_{ij} \leq \|\mathbf{y}_i - \mathbf{y}_j\|_2$ with $\delta_{ij} = \delta_{ji}$ for all distinct $i, j \in \{1, 2, \ldots, m\}$, and.
- a nonzero positive value $\varepsilon < \frac{1}{2} \min_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|_2$
- a function $g : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ such that $g(\mathbf{x}, \mathbf{y}) \neq g(\mathbf{y}, \mathbf{x})$ for all distinct $\mathbf{x}, \mathbf{y} \in \mathbb{R}^\ell$ (see Remark 1)

1: $h \leftarrow \binom{m}{2}$ ▷ dimension of the space in which distances will be computed
2: fix a value s with $0 < s < \varepsilon/\sqrt{h}$
3: $D \leftarrow \emptyset$ ▷ to collect pairwise difference vectors
4: **for** $i = 1, \ldots, m$ **do**
5: **for** $j = i + 1, \ldots, m$ **do**
6: Seed a pseudorandom generator PRNG() with the value $g(\mathbf{y}_i, \mathbf{y}_j)$, and let PRNG() output uniformly distributed values in $[0, 1]$.
7: Draw a lot of $h - \ell$ pseudorandom values $r_1, \ldots r_{h-\ell} \leftarrow s \cdot \text{PRNG}()$
8: Define $\mathbf{z}_{i,i} \leftarrow (\mathbf{y}_i, r_1, \ldots, r_{h-\ell}) \in \mathbb{R}^h$.
9: Re-seed PRNG() with the value $g(\mathbf{y}_j, \mathbf{y}_i)$
10: Draw fresh values $r_1, \ldots r_{h-\ell} \leftarrow s \cdot \text{PRNG}()$
11: Define $\mathbf{z}_{i,j} \leftarrow (\mathbf{y}_j, r_1, \ldots, r_{h-\ell}) \in \mathbb{R}^h$.
12: $D \leftarrow D \cup \{\mathbf{z}_{i,i} - \mathbf{z}_{i,j}\}$ ▷ almost surely, these will be all linearly independent (Lemma 1)
13: **end for**
14: **end for**
15: Construct the norm $\|\cdot\|_Q$ on the points in D such that $\|z_{i,j} - z_{j,i}\|_Q \propto \delta_{ij}$ for all $i \neq j$ ▷ steps in the proof of Theorem 1; see Table VII for an example
16: Define the ε-semimetric $\widetilde{d} : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ on the input points $\mathbf{y}_i, \mathbf{y}_j$ as

$$\widetilde{d}(\mathbf{y}_i, \mathbf{y}_j) := \|\mathbf{z}_{i,j} - \mathbf{z}_{j,i}\|_Q,$$

with the vectors $\mathbf{z}_{i,j}, \mathbf{z}_{j,i}$ computed by padding with pseudorandom noise (as in lines 6…11).
Ensured after this step (by Theorem 2):
$\widetilde{d}$ is an ε-semimetric that satisfies $\widetilde{d}(\mathbf{y}_i, \mathbf{y}_j) \propto \delta_{ij}$ for all $i \neq j$, which will make a similarity-based clustering give results consistent with the chosen distances (next step).
17: Run a clustering algorithm (e.g., k-Means, DBSCAN, …) using $\widetilde{d}$ as the distance function to classify the points $\mathbf{y}_1, \ldots, \mathbf{y}_m$ with distances proportional to δ_{ij} between them (towards the desired result).

`rand()` with a seed value computed nonlinearly from both, $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$, under constraints explained in Appendix C.

The quadratic form q exactly gives the desired distances, as shown in Table VII for an example run of the construction, and we refrain from repeating another table showing similar results.

To verify our setup, we evaluated the intra-cluster and inter-cluster distances. Table V shows a snapshot of one (out of many runs), confirming that the distances are small inside a cluster and large between any two clusters. To avoid very small and very large numbers causing numeric issues, we used the unscaled version (i.e., without the downscaling by the largest eigenvalue as in (6)) of the quadratic form here, which does no change to the desired relations of which points are close to one another and which are far away from each other. The experiments nicely show how the clusters are separated by a large distance at least (proportional to δ_{large}), while points within the same cluster are separated by a small distance at most (proportional to δ_{small}).

Point no.	y_1	y_2	desired class
1	18,4875	-7,4766	2
2	9,1751	2,223	3
3	-0,5026	9,942	2
4	-4,2728	-1,1626	1
5	-8,7678	5,5768	3
6	-13,2171	-6,7889	1
7	4,8107	-4,9768	3
8	-5,5598	4,0152	2
9	-7,7993	4,3651	2
10	21,8574	2,3408	3

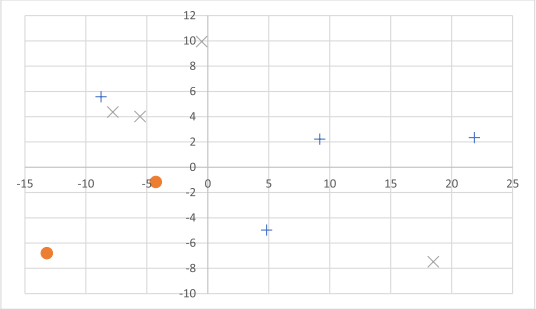

Fig. 3. 10 example points with spatial locations, and markers representing the different *desired* classes.

TABLE V
SEPARATION WITHIN AND BETWEEN CLUSTERS

(a) Cluster centers (= averages of all desired cluster points)

Cluster	Center (x, y)
1	(-8.7450, -3.9758)
2	(1.1565, 2.7114)
3	(6.7688, 1.2909)

(b) Maximum distance within a cluster

Cluster	1	2	3
distance within	6.33×10^{-11}	6.31×10^{-11}	6.37×10^{-11}

(c) Minimum distance between two clusters

Cluster	1	2	3
1		5.89×10^{-5}	5.89×10^{-5}
2	5.89×10^{-5}		5.89×10^{-5}
3	5.89×10^{-5}	5.89×10^{-5}	

With this setup, we ran a slightly adapted version of the k-Means implementation provided by [61]. The change we made was twofold, namely:

- We adapted the function to take our ε-semimetric to compute distances,
- and we kept the number of iterations in the algorithm fixed by not letting it terminate earlier than before reaching the maximum number of iterations.

The k-Means algorithm then received the following input: we computed the desired cluster centers by grouping and averaging the points $\mathbf{y}_i$ according to their cluster. The cluster centers computed (also reproducible from the table in Figure 3), are shown in Table Va.

That is, the center for the class c cluster is the arithmetic mean of all points $\mathbf{y}_i$ to which we assigned class$(\mathbf{y}_i) = c$, and this was done for all classes. The resulting cluster centers were

then included in the data points (see Table VI, columns to the left of the vertical separator line) on which we constructed the metrics following the setup steps above, and put them in front of the list of input points to the k-Means algorithm. This covers two versions of how k-Means is usable: we can either directly supply it with the cluster centers (this is what our implementation does), and let it run on the given data points (which then include the cluster centers accordingly). Alternatively, we can refrain from supplying cluster centers, in which case k-Means (in the given implementation) would take the first k points as cluster centers to start with if there are k classes. Since we have put the cluster centers upfront in the list, the algorithm will in both cases be initialized identically.

1) Results: We found that the construction is numerically relatively unstable since the eigenvalues of the matrix $\mathbf{A}$ to construct the quadratic form in some cases fell out of the numerical scope of the machine precision. Whenever the experiment finished without over- or underflows, however, the clustering of k-Means came up *exactly as we desired*. Table VI shows the results of k-Means in the column "by k-Means". It is directly visible that k-Means produced just the random classes assigned. The experiment is repeatable with a freshly seeded random number generator each time and confirmed the findings also over many independent repetitions. We also let the algorithm run on cluster centers that were simply chosen as arbitrary z-neighbors of some point within the desired cluster, but not the direct arithmetic mean of the cluster as such. This made the algorithm frequently update (shift) the cluster center, which due to the construction of our quadratic form, led to strong changes in the distances as were computed. The resulting cluster assignments were, in that case, far off what we desired them to be. The choice of the arithmetic mean of the desired points to be the cluster centers, on the contrary, will effectively let the algorithm re-compute the same cluster center over and over again so that the distances that we designed really come up as desired. This leads to the results reported in Table VI.

Other work that tackles the problem of cluster center determination (such as we require for k-Means for example), is compatible with our constructions; noting that the cluster centers can be computed by any means, including sophisticated methods as in [62] and [63], for example. In our case, whether the clustering algorithm will work well with the user-supplied cluster centers depends on how much the cluster centers may become adapted by the algorithm. We tested this version of k-Means, finding that letting the algorithm change the cluster centers, destroys the designed effects and outcomes and the experiments all failed.

C. Manipulating DBSCAN

With the same setup as for k-Means, we let DBSCAN do a clustering for us, with the following differences to section VI-B:

- No inclusion of pre-computed cluster centers; the data was given to DBSCAN just as it came out of the random generator.
- DBSCAN's parameters "neighborhood size ϵ" and "minimum number $minPts$" of points within the ϵ-neighborhood were set to $\epsilon = \frac{\delta_{\text{small}}+\delta_{\text{large}}}{2}$ and $minPts = 2$.

TABLE VI

EXPERIMENTAL VERIFICATION OF PROPORTIONAL PAIRWISE DISTANCES, RANDOMLY ASSIGNED PAIRWISE BETWEEN 10 POINTS IN $\mathbb{R}^2$. THE RIGHT COLUMNS SHOW WHAT k-MEANS AND DBSCAN RECOVERED USING THE INPUT DATA ON THE LEFT, AND OUR DESIGNED ε-SEMIMETRIC TO MEASURE DISTANCES

Point no.	y_1	y_2	random class	by k-means	by DBSCAN
1	18,4875	-7,4766	2	2	1
2	9,1751	2,223	3	3	2
3	-0,5026	9,942	2	2	1
4	-4,2728	-1,1626	1	1	3
5	-8,7678	5,5768	3	3	2
6	-13,2171	-6,7889	1	1	3
7	4,8107	-4,9768	3	3	2
8	-5,5598	4,0152	2	2	1
9	-7,7993	4,3651	2	2	1
10	21,8574	2,3408	3	3	2

1) Results: As far as the scalability issue concerns the construction of the norm $\|\cdot\|_Q$ on $\mathbb{R}^h$, this issue exists in all experiments and relates to the method in general. However, unlike k-Means, the experiments on DBSCAN ran without any numerical issues and always terminated, in all cases delivering an "isomorphic" clustering. That is, DBSCAN used a different naming (enumeration) of the classes, but there was always a one-to-one correspondence between the class c that DBSCAN assigned to the point y_i, and the (randomly desired) classification class($\mathbf{y}_i$). Table VI shows the results in the column "by DBSCAN", where it is visible that the cluster naming does not match the original clusters, but remains "the same" via the one-to-one correspondence $\{1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 2\}$, between the randomly assigned desired classes and those that DBSCAN recovered.

D. Manipulated DBSCAN on MNIST Data

Let us now test the manipulations of DBSCAN on elements from the classical MNIST dataset on handwritten digits. To keep the example simple, let us aim at the recognition of primes in the set $\{0, 1, \ldots, 9\}$. We let DBSCAN run first in its "original" form, and then manipulate it to recognize primes accurately, or inaccurately, including or excluding numbers as we wish.

Feature engineering is often done to enhance the accuracy and quality of predictions. Hence, we also do it here to keep the example feasible and to rule out any manipulations by (visually incomprehensible) noise on the picture (as is used elsewhere in adversarial machine learning).

Our example features are partly manually, partly automatically assignable, and are fourfold: (1) is the number positive? (0/1 indicator), (2) is it an integer? (0/1 indicator), (3) does the number have nontrivial divisors? (0/1 indicator) and (4) how "similar" is the number to one of the primes $\{2, 3, 5, 7\}$, where similarity is here measured by the earth-mover distance to a fixed set of reference digits (in our case the first ones of each kind to appear in the dataset). This feature is directly derived from the visual appearance of the MNIST picture. We remark that these choices are just examples and the overall method

						
Features	Positive?	0	1	1	1	1
	Integer?	1	1	1	1	1
	No nontriv. divisors	0	0	1	1	0
	Similarity	2.109906	1.600969	1.798213	1.470953	1.605889
Exp1	DBSCAN with Eucl. Distance	1	4	2	2	4
Exp2	DBSCAN with ε-semimetric: "just primes"	1	1	2	2	1
Exp3	DBSCAN with ε-semimetric: "just primes incl. 1"	1	2	2	2	1
Exp4	DBSCAN with ε-semimetric: "all odd = prime"	1	2	1	2	1
Exp5	DBSCAN with interpolator dist.: "just primes"	1	1	2	2	1

						
Features	Positive?	0	1	1	1	1
	Integer?	1	1	1	1	1
	No nontriv. divisors	1	0	1	0	0
	Similarity	1.134939	1.212047	0.4297715	1.591323	0.7591084
Exp1	DBSCAN with Eucl. Distance	3	4	3	4	4
Exp2	DBSCAN with ε-semimetric: "just primes"	2	1	2	1	1
Exp3	DBSCAN with ε-semimetric: "just primes incl. 1"	2	1	2	1	1
Exp4	DBSCAN with ε-semimetric: "all odd = prime"	2	1	2	1	2
Exp5	DBSCAN with interpolator dist.: "just primes"	2	1	2	1	1

Fig. 4. Running DBSCAN under different manipulations.

does not depend on how the features are made. In a practical setting, the features would, more likely, be extracted from the images by more sophisticated means (by AI techniques) before entering a classification. We keep our example simple here, maintaining both, manual and automatically derived features from the public dataset. Figure 4 displays the digit images, with assigned features and a series of experimental runs of DBSCAN with different underlying distance functions:

Experiment 1: DBSCAN with Euclidean distance, $minPts = 2$ and ε set to the middle of the range in which the pairwise (Euclidean) distances between the feature vectors lie. That is if $d_{\min}$ and $d_{\max}$ are the pairwise minimum and maximum distances between the feature vectors, we put $\varepsilon := \frac{1}{2}(d_{\min} + d_{\max})$ to get some nontrivial classification. DBSCAN's results are shown in row "Exp1" of Figure 4. It becomes evident that, despite the sought result being already among the features, the classification is far from correct (if it were about recognizing primes); additionally, the earth mover's distance is here a particularly bad feature to rely on visual appearance. However, let us, for illustration only, that these features were meaningful and cannot be adapted. In that case, manual correction may be necessary. This is our next experiment 2.

Experiment 2: Given that the classification of DBSCAN based on the given features is not satisfying (based on given features that we may be unable to change) we can "manually fix" the desired results by assigning class 1 for the numbers $\{0, 1, 4, 6, 8\}$ and class 2 for the (primes) $\{2, 3, 5, 7\}$. By defining the desired classes in this way, and designing an ε-semimetric for it, the experiment's results were *perfectly aligned* with what they should be, assigning class "2" to exactly the primes. This is a *non-adversarial* application of the concept, although its performance on new data (for which the ε-semimetric was not constructed) remains generally undefined. Observed numerical round-off errors suggest that some stabilization of the construction is needed, which is left as an open problem for future research here.

Experiment 3: some people may ask why the number 1 is not a prime, and mathematics has various good reasons to exclude it from the primes. However, we can easily make our prime classifier from the previous experiment 2 accept "1" as a prime: all we have to do is define the class that the respective picture (represented by its feature vector) should have to be the "primes", and after constructing the ε-semimetric, DBSCAN will include it.

Experiment 4: extending the manipulation further, let us now make all odd digits primes (against that 1 is by definition, not a prime, 2 is a prime, and $9 = 3 \times 3$ is composite). The matter is again only to define the classes for "1" and "9" to be "prime", constructing the ε-semimetric accordingly, and letting DBSCAN run with it to confirm what we desire.

1) Comparison to Alternative Manipulation Methods:
Experiment 5: as some sort of comparison to alternatives, we remark that it is indeed possible to manipulate DBSCAN *without* resorting to our particular construction. The most sophisticated form of this alternative would be Siamese networks, but we can illustrate it easier: we constructed a simple inverse-distance weighting interpolating function $d(\mathbf{x}, \mathbf{y})$ that gives exactly the desired distances between feature vectors $\mathbf{x}, \mathbf{y}$, in the very same way as shown for the construction of an ε-semimetric. Quite expected, DBSCAN likewise reproduces the desired classification (here, the ones from Experiment 2). While this works well in terms of results, the plausibility of the underlying distance is not ensured: indeed, the interpolator, but probably most methods to approximate functions or number sets, can mostly reproduce values at high accuracy, but, unless constructed for it, will fail to also reproduce analytic properties. In our case, it is a simple matter of "black-box-testing" the underlying distance function on pairs and triples to verify (in fact refute) the properties of a metric, thus making the manipulation recognizable. On the contrary, an ε-semimetric not only interpolates a given set of values but also does so respecting analytic properties, which within a numeric accuracy of $\geq \varepsilon$ cannot be recognized by black-box testing. This feature substantiates its "plausibility" and distinguishes it from alternatives like deep machine learning models or similar for the same manipulation goals.

2) Observations: The construction suffers from numeric round-off errors when the dimensionality becomes too high, i.e., if we construct the distance to match more items, or if the number of features becomes very high. Likewise, the seeding function g should be chosen to give nonzero seeds in all cases (again observing numerical issues otherwise). DBSCAN was generally more stable than k-Means in this respect, whose manipulation in the above sense failed for exactly (and only) numerical round-off errors. If a clustering,

however, is based on features extracted from the data at hand, then a manipulation like shown for the MNIST images becomes feasible again.

VII. Discussion

A. Complexity

The complexity of workflow 1 is governed by the computation of nullspaces in line 15. This is done for h matrices given by (5) of dimension $h \times h$, with $h \in O(m^2)$ for m data points. Using Gaussian elimination or singular value decomposition (numerically more stable), the null spaces are computable in $O(h^3)$ steps per vector. This computation can be parallelized for all h vectors. The second most expensive operation is the computation of eigenvalues in equation (6), but this is done only once. Since we can replace the value $\lambda_{\max}(\mathbf{A})$ in (6) also by a bound obtained from Gershgorin's circle theorem [64], this step does not add significantly to the runtime. With parallelization, the overall complexity is $O(h^3) = O(m^6)$ (sequentially, it would be $h \cdot O(h^3) = O(m^8)$). Practically, it nonetheless pays to resort to numeric techniques and dimensionality reduction (e.g., by spectral clustering techniques).

B. Applicability to Spectral and Hierarchical Clustering

ε-semimetrics are also applicable in spectral clustering methods [28]. There, two functions are in use: one is a similarity function (for example, using a Gaussian kernel) to prepare a graph, and the other is a standard distance used in a second step for the actual clustering on the graph nodes. In rough terms, with details well explained in [28], the basic outline of spectral clustering is the following: 1) arrange the data points as nodes of a complete undirected graph, and label the edge from the data point $\mathbf{x}$ to $\mathbf{y}$ with the similarity value $s(\mathbf{x}, \mathbf{y})$. Then, remove some edges according to a fixed rule from several possibilities:

- Fix some threshold $\varepsilon > 0$ and remove an edge if its similarity is $> \varepsilon$,
- Fix some value $k \in \mathbb{N}$ and delete all edges, except those with the k lowest similarity values
- Keep the graph complete (delete no edges at all), and only rely on similarity values that are adjustable by choosing the function s appropriately; for example, a Gaussian kernel with zero mean and variance σ, which then plays a similar role as the ε-threshold above (setting $\sigma < \varepsilon/3$ will result in the same effect as the first heuristic for about 99.8% of nodes)

Once the graph is prepared, the clustering proceeds by constructing its Laplacian matrix, computing the eigenvectors, and running a conventional clustering on the matrix of eigenvectors (only). At this point, the second of the aforementioned functions comes in since the last step is still done with a standard technique, such as k-Means (which, for example, [28] explicitly mentions). Since our method is independent of the actual nature or dimension of the feature vectors, it works, in the same way, to manipulate the last step only in a spectral clustering, but this is sufficient to forge the results likewise. However, since spectral clustering essentially transforms n data points to a sequence of k eigenvectors of potentially much lower dimension, the numeric stability of our forgery method can be better than in a direct application of k-Means or DBSCAN.

For *divisive hierarchical clustering*, our construction can be used to design metrics to separate the points into desired clusters in each stage, possibly using different metrics for each step. Similarly, for *agglomerative clustering*, we can let points be singleton clusters at the beginning, and design an ε-semimetric to merge clusters as we desire in each step. Both cases boil down to a systematic choice of distances to make the clustering produce the desired results, which is a separate issue of this work's contribution.

C. Scalability and Practicality

Numeric round-off errors induce the practical limitation of our method; specifically, we will need to compute singular value decompositions and eigenvalues of matrices whose dimension is quadratic in the number of data points to a cluster. Hence, practical manipulations on real-life size datasets may require an implementation with arbitrary-precision arithmetic.[5] Exploring this direction, especially in terms of the added computational complexity for high-precision arithmetic, is beyond the scope of this article. Applying the construction in the context of spectral clustering in a thereby smaller set (consisting of only a few, even though large-dimensional, eigenvectors [28]; cf. Section VII-B), may thus be a beneficial subject of future studies.

Although experimentally verifiable, the procedure does not scale well as the experiments with k-Means have shown (cf. also Section VI-D2). The algorithms used here were selected for their simplicity and availability as "bare bones implementation", void of complicated overheads for integration in larger libraries. This choice was aimed at easing matters of verification and reproduction of results in this work. Future studies may thus try to use off-the-shelf implementations of the same or more powerful clustering algorithms in other programming languages such as Python or similar. This work is intended as a mere first step to establish the theory. Extending the experiments to more advanced clustering algorithms than the two basic ones that we have used here, is a further direction for future work.

VIII. Conclusion

A natural question relates to how plausible the use of a metric crafted like ours is in practice. Unless the clustering is audited at the level of its source code, a numeric "black-box testing" of the distance function's properties may efficiently and effectively indicate if the distance is *not* a topological metric. Our construction would resist such a purely functional verification, since it has the same properties as a true metric, up to the ε-error in the triangle inequality. However, if numbers are displayed to users with a numeric accuracy less than ε, while the inner calculations are done with higher precision,

[5] Wikipedia provides a comprehensive list of suitable software libraries at https://en.wikipedia.org/wiki/List_of_arbitrary-precision_arithmetic_software.

the violation of the triangle inequality will become practically indistinguishable from natural round-off errors.

Our findings have a qualitative impact on how clustering should be made to be *trustworthy*. Essentially, the advice is the same as in cryptographic security, namely to immutably fix all configuration data of the clustering algorithm, in particular the norm that it uses. If there is a verifiable commitment to any (fixed) norm or metric, such as the Euclidean, is made, then the results of the clustering will be modifiable only by violations of the (input) data integrity, such as by poisoning. Otherwise, if the metric is not fixed and hence changeable by an adversary (e.g., by externally supplying its own function instead of the default), clustering can be manipulated without touching any of the input data, while preserving the input data integrity. This is important in cases of adversarial denial of a clustering result: suppose a person has, by some clustering algorithm, been assigned to a certain group suffering discrimination (of any kind). If the clustering is a posteriori manipulable, this assignment becomes demonstrably deniable, if the adversary designs the metric to produce the same output as before, but puts the person in question into a different cluster than before. If the metric is verifiably fixed from the beginning, the clustering becomes undeniable (at least by our techniques).

The lesson we learned from our study and experiments is the inevitable need to enforce transparency and a prior commitment to algorithms, parameters, metrics, and all details of a clustering algorithm before applying it to data. Our "attacks" depend on the possibility of a posterior manipulation of the algorithms to justify a desired result. This possibility can include reproducing a past set of legitimate clustering decisions and may only affect the manipulation of a single or a few data points (set the distances accordingly to match an honest setup, and only set some values to distances of malicious desire).

The operational principles to derive from these insights are well known and standard in cryptography, known as Kerckhoffs' principle, and elsewhere also called a "nothing-up-my-sleeve" design choice.[6] The idea is to explain where design choices, even if they refer to seemingly arbitrary constants, come from, and are not due to some hidden agenda of the designer ("nothing up my sleeve").

The principle is the same here: the implementer and user of the clustering algorithm should be obliged to explain all design choices and transparently and publicly commit to them before letting the algorithms become operational. If not, then a posteriori manipulations, just as we have demonstrated, become possible.

APPENDIX

A. Proof of Lemma 1

We give the proof for uniform distributions, but remark that the argument only depends on the continuity of the distribution (w.r.t. the Lebesgue measure), but this more general claim is not required in the following.

[6]This term stems from the field of symmetric cryptography, where constants in algorithms are chosen in a way that allows to re-calculate the constant (e.g., use a hexadecimal representation of $\sqrt{2}$ as is done in some hashing algorithms or similar).

We consider a selection of the first k column-vectors (counted from the left) in $\mathbf{M}$. By induction over k, we show linear independence up to $k = n$, which then means full rank of $\mathbf{M}$. The case $k = 1$ is trivial, so let us assume that up to $k < n$ columns $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^n$ have been sampled linearly independent as $\mathbf{x}_1 \sim F_1, \ldots, \mathbf{x}_k \sim F_k$. The new vector $\mathbf{x}_{k+1} \in \mathbb{R}^n$, to be a linear combination of $\mathbf{x}_1, \ldots, \mathbf{x}_k$, would need to lie in the k-dimensional subspace of $\mathbb{R}^n$. This subspace has dimension $k < n$ and is therefore a nullset w.r.t. the Lebesgue measure on $\mathbb{R}^n$. Since we sample from absolutely continuous distributions on $\mathbb{R}^n$ w.r.t. the n-dimensional (Lebesgue) measure, we have $\Pr_{F_{k+1}}(\mathbf{x}_{k+1} \in \operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}) = 0$ too under the distribution F_{k+1}, thus completing the induction step. □

B. Proof of Theorem 2

We will place a "virtual neighbor" near each data point $\mathbf{y}_1, \ldots, \mathbf{y}_m$, such that the distances between the virtual neighbors of each data point are freely specifiable. To bypass the $O(\sqrt{\ell})$-bound on m given by (3), we will respectively increase the dimension towards $h = \binom{m}{2}$ to have a sufficient number of pairs embeddable with linearly independent differences, and in close proximity to the given data points $\mathbf{y}_1, \ldots, \mathbf{y}_m$.

Let us first put the data points into the higher-dimensional space $\mathbb{R}^h$ as $\mathbf{y}'_i = (\mathbf{y}_i^\top \ \mathbf{0}_{(h-\ell)\times 1})^\top \in \mathbb{R}^h$. Then, fix some positive $\varepsilon < \frac{1}{2}\min_{i\neq j} \left\|\mathbf{y}'_i - \mathbf{y}'_j\right\|_2 = \frac{1}{2}\min_{i\neq j} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2$, and to each point $\mathbf{y}'_i$, associate a set of $m-1$ random points $\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,m-1}$ within an Euclidean distance $< \varepsilon$ towards $\mathbf{y}_i$. These points will serve to measure the pairwise distances between $\mathbf{y}_i$ and $\mathbf{y}_j$ using higher-dimensional "substitutes". Note that we cannot just use $\mathbf{y}'_i, \mathbf{y}'_j$ directly, since the pairwise differences between them could not give us linearly independent vectors as we require. To see this, recall that we have m such vectors $\mathbf{y}'_1, \ldots, \mathbf{y}'_m$, and computing pairwise differences is a linear operation. Hence, the result cannot be linearly independent vectors. For that to happen, we need points that are "independent" of $\mathbf{y}_i, \mathbf{y}_j$ to give linearly independent differences, but still close enough to $\mathbf{y}_i, \mathbf{y}_j$ to "represent" their spatial location. Therefore, we sample from an ε-neighborhood that is small enough to ensure disjointness between any two points $\mathbf{y}'_i, \mathbf{y}'_j$, and sample a fresh random point near $\mathbf{y}_i$ whenever we need a distance vector between $\mathbf{y}_i$ and one of the other points. Figure 1 shows this.

To see why this establishes the required linear independence, let us systematically iterate through the pairwise differences to see why linear independence is accomplished by the construction as described.

Start with $\mathbf{y}_1 \in \mathbb{R}^\ell$, respectively $\mathbf{y}'_1 \in \mathbb{R}^h$: for all other points $\mathbf{y}'_j$ for $j = 2, \ldots, m$, we sample a fresh neighbor $\mathbf{z}_{i,j} \in \mathcal{U}(B_{2,\varepsilon}(\mathbf{y}'_i))$ and $\mathbf{z}_{j,1} \in \mathcal{U}(B_{2,\varepsilon}(\mathbf{y}'_j))$ uniformly at random. We then compute the distance vector $\mathbf{x}_k = \mathbf{z}_{i,j} - \mathbf{z}_{j,1}$ (with the index $k \simeq ij$). This procedure is likewise repeated for $i = 2, \ldots, m$, giving the difference vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\binom{m}{2}} \in \mathbb{R}^h$, all collected in a matrix

$$\mathbf{M} = (\mathbf{x}_1, \ldots, \mathbf{x}_h) \in \mathbb{R}^{h\times h}. \tag{5}$$

Claim 1: Let $\mathbf{y}_1, \ldots, \mathbf{y}_m$ be given, The matrix $\mathbf{M}$ constructed from pairwise differences of uniformly random points

$\mathbf{x}_i \sim \mathcal{U}(B_{2,\varepsilon}(\mathbf{y}_i))$ *within neighborhoods centered at the points* $\mathbf{y}'_1, \ldots, \mathbf{y}'_m$*, has almost surely full rank, i.e.,* $\Pr(rank(\mathbf{M}) = h) = 1$.

Proof of Claim 1: This is a mere application of Lemma 1 with the distribution for $\mathbf{x}_i$ being $\mathcal{U}(B_{2,\varepsilon}(\mathbf{y}_i))$. Since these are all absolutely continuous, the coordinates of each $\mathbf{x}_k$ are differences of independent uniformly distributed random variables, and as such have a trapezoidal shaped distribution, which is again absolutely continuous, and Lemma 1 applies.□

Claim 1 assures the conditions needed for the construction in Section IV-A are met with probability 1, and it just remains to assign the distances between the substitute points $\mathbf{z}_{i,s}, \mathbf{z}_{j,t}$ (for $1 \leq s, t \leq m-1$) in the neighborhood of $\mathbf{y}_i$ (resp. $\mathbf{y}'_i$) and $\mathbf{y}_j$ (resp. $\mathbf{y}'_j$) to be as we desire.

Now, repeating the construction from Section IV-A on the pairwise difference vectors (that are now all linearly independent, since they were computed from distinct z-neighbors of the original points), we get a quadratic form q_h on the higher-dimensional space $\mathbb{R}^h$. The resulting norm $\|\mathbf{x}\|_q := \sqrt{q_h(\mathbf{x})}$ is equivalent to the Euclidean norm, and the respective constants are given by the eigenvalues of the matrix $\mathbf{A}$ within $q_h(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x}$, since every quadratic form satisfies

$$\lambda_{\min}(\mathbf{A}) \cdot \|\mathbf{x}\|_2^2 \leq \mathbf{x}^\top \mathbf{A}\mathbf{x} \leq \lambda_{\max}(\mathbf{A}) \cdot \|\mathbf{x}\|_2^2,$$

where $\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})$ are the smallest and largest eigenvalues of $\mathbf{A}$. Experimentally, one quickly discovers that there is not much control over the eigenvalues of $\mathbf{A}$ directly, and this is not surprising, since the norm induced by q_h cannot diverge arbitrarily from the Euclidean distances (due to its topological equivalence). For our purposes, however, it will suffice to scale all eigenvalues by the same factor and put

$$\alpha := \frac{1}{\lambda_{\max}(\mathbf{A})} \tag{6}$$

to define the scaled quadratic form

$$Q(\mathbf{x}) := \mathbf{x}^\top (\alpha \cdot \mathbf{A})\mathbf{x} \tag{7}$$

whose induced norm satisfies

$$\|\mathbf{x}\|_Q = \sqrt{Q(\mathbf{x})} \leq \|\mathbf{x}\|_2 \tag{8}$$

since its largest eigenvalue was just scaled down to 1. The scaling is necessary for the subsequent construction of an ε-semimetric, whose violation of the triangle inequality should remain within an ε-bound. This tolerance ε is set based on Euclidean distances and our norm should hence be bounded by the 2-norm.

We show that $\|\mathbf{x}\|_Q$ induces distances that are consistent with the closeness relations induced by the set of desired distances $\{\delta_{ij} : 1 \leq i < j \leq m\}$ in the following sense: let i, j, k be a triple selection from the set of given points, for which we want $\mathbf{y}_i$ to be closer to/equal/or farther away from $\mathbf{y}_j$ than $\mathbf{y}_k$. The possible relation can be that the distance from point i to point j is less than, equal to, or larger than the distance to the third point k, which we jointly express as

$$\delta_{ij} \lessgtr \delta_{ik}. \tag{9}$$

The norm $\sqrt{Q(x)}$ by construction is consistent with (9) for all possible triples of neighboring points $\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k$ (dropping the double index to simplify notation and to signify that these points are arbitrary ones from the list of existing), since

$$\|\mathbf{z}_i - \mathbf{z}_j\|_Q = \sqrt{\alpha} \cdot \delta_{ij} \lessgtr \sqrt{\alpha} \cdot \delta_{jk} = \|\mathbf{z}_i - \mathbf{z}_k\|_Q, \tag{10}$$

in which the $<$, $=$ or $>$ relations are in all cases preserved since $\alpha > 0$, as $\mathbf{A}$ is positive definite.

The Euclidean ε-neighborhoods of each given data point $\mathbf{y}_i$ will be contained in the neighborhoods that our constructed norm induces, and the same $\|\cdot\|_Q$-ball will contain points that do have the desired distances between them, up to the constant multiple α. The proof is complete. □

C. Proof of Theorem 3

The proof structure is essentially the description preceding Theorem 3 but only made rigorous. To this end, let $\varepsilon > 0$ be given, and choose some deterministic function $f : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ with two properties:

$$|f| \leq \frac{\varepsilon}{6} \tag{11}$$

$$f(\mathbf{x}, \mathbf{y}) \neq f(\mathbf{y}, \mathbf{x}) \text{ for all } \mathbf{x} \neq \mathbf{y}. \tag{12}$$

The mapping of $\mathbf{y} \in \mathbb{R}^\ell \mapsto \mathbf{z} \in \mathbb{R}^h$, with $h = \binom{m}{2}$, is done by using the function f to add the "random noise". Note that the purpose of this random noise was, however, just to create linear independence, and we can accomplish this also in entirely deterministic ways: Lemma 1 implies that each pair $(\mathbf{y}_i, \mathbf{y}_j)$ admits, with probability 1, a selection of two neighbors $\mathbf{z}_{i,j} \in B_{Q,\varepsilon/6}(\mathbf{y}_i)$ and $\mathbf{z}_{j,i} \in B_{Q,\varepsilon/6}(\mathbf{y}_j)$ such that the pairwise differences $\{\mathbf{z}_{i,j} - \mathbf{z}_{j,i}\}_{ij}$ are linearly independent. Since this selection can be done with probability 1, suitable neighbor points *exist* and we can fix them as constants. The "noise" is then an additive term $f(\mathbf{y}_i, \mathbf{y}_j) = \varepsilon_{ij}$ to give $\mathbf{y}'_i + \varepsilon_{ij} = \mathbf{z}_{i,j}$ (likewise gives $\mathbf{y}'_j = \mathbf{y}_j + f(\mathbf{y}_i, \mathbf{y}_j) = \mathbf{z}_{j,i}$), where $\mathbf{y}'_i = (\underbrace{\mathbf{y}_i}_{\in \mathbb{R}^\ell}, 0, \ldots, 0) \in \mathbb{R}^h$ and $\mathbf{y}'_j$ are the canonic embeddings of $\mathbf{y}_i, \mathbf{y}_j$ into $\mathbb{R}^h$. To keep the noise magnitude under the allowed ε-bound for the triangle inequality, we scale the noise by a factor $s > 0$ to make $|s \cdot f(\mathbf{y}_i, \mathbf{y}_j)| = \|s \cdot \varepsilon_{ij}\|_Q < \varepsilon/6$. Since f adds a fixed noise vector in all cases, the noise function itself is deterministic. For pairs $(\mathbf{x}, \mathbf{y}) \notin Y \times Y$, we let f return a zero vector (i.e., add no noise to the points).

On the chosen neighboring points, we can define the quadratic form and induced norm that satisfies $\|\mathbf{z}_{i,j} - \mathbf{z}_{j,i}\|_Q = \delta_{ij}$. This construction is possible, since by hypothesis, the distances desired are all less or equal to the Euclidean distances, so we do not have problems with scaling. The ε-semimetric is then defined as

$$\tilde{d}(\mathbf{x}, \mathbf{y}) := \|(\mathbf{x}, f(\mathbf{x}, \mathbf{y})) - (\mathbf{y}, f(\mathbf{y}, \mathbf{x}))\|_Q.$$

It remains to show that this is indeed an ε-semimetric, so we check the corresponding properties:

Identity of Indiscernibles: since f is a deterministic mapping, we get the noise $f(\mathbf{x}, \mathbf{x})$ identically in both terms, giving $\|(\mathbf{x}, f(\mathbf{x}, \mathbf{x})) - (\mathbf{x}, f(\mathbf{x}, \mathbf{x}))\|_Q = 0$, since we have a norm on $\mathbb{R}^h$.

Positivity: for $\mathbf{x} \neq \mathbf{y}$, we have $(\mathbf{x}, f(\mathbf{x}, \mathbf{y})) \neq (\mathbf{y}, f(\mathbf{y}, \mathbf{x}))$, and since $\|\cdot\|_Q$ is a norm on $\mathbb{R}^h$, positivity follows directly.

TABLE VII
EXPERIMENTAL VERIFICATION OF PROPORTIONAL PAIRWISE DISTANCES, RANDOMLY ASSIGNED PAIRWISE BETWEEN 10 POINTS IN $\mathbb{R}^2$

point pair $k \simeq ij$	desired distance unscaled δ_{ij}	distance as given by $\sqrt{q(\mathbf{x}_k)}$ with $\mathbf{x}_k = \mathbf{y}_i - \mathbf{y}_j$; see (2)
1	1.9383	1.9383
2	2.9016	2.9016
3	2.2079	2.2079
4	1.2656	1.2656
5	2.8503	2.8503
6	1.2466	1.2466
7	2.6005	2.6005
8	1.6730	1.6730
9	2.0714	2.0714
10	2.3622	2.3622
11	2.6508	2.6508
12	2.8490	2.8490
13	1.4008	1.4008
14	2.4533	2.4533
15	2.5970	2.5970
16	1.0175	1.0175
17	1.8562	1.8562
18	2.5771	2.5771
19	1.2072	1.2072
20	1.7114	1.7114
21	2.1238	2.1238
22	1.4545	1.4545
23	2.5350	2.5350
24	2.4131	2.4131
25	1.3432	1.3432
26	1.3689	1.3689
27	1.2163	1.2163
28	1.6719	1.6719
29	2.1236	2.1236
30	1.7685	1.7685
31	2.5814	2.5814
32	2.5736	2.5736
33	2.6235	2.6235
34	1.6553	1.6553
35	2.6174	2.6174
36	1.2535	1.2535
37	2.5714	2.5714
38	1.3655	1.3655
39	2.8465	2.8465
40	2.9228	2.9228
41	2.8777	2.8777
42	1.8183	1.8183
43	2.8736	2.8736
44	2.6435	2.6435
45	1.6349	1.6349

Symmetry: directly by construction, since $\tilde{d}(\mathbf{x}, \mathbf{y}) = \|(\mathbf{x}, f(\mathbf{x}, \mathbf{y})) - (\mathbf{y}, f(\mathbf{y}, \mathbf{x}))\|_Q = \|(\mathbf{y}, f(\mathbf{y}, \mathbf{x})) - (\mathbf{x}, f(\mathbf{x}, \mathbf{y}))\|_Q = \tilde{d}(\mathbf{y}, \mathbf{x})$.

Approximate Triangle Inequality: for this, we first prove the following claim, saying that $\tilde{d}$ and d differ by at most $\frac{\varepsilon}{3}$ by construction: to see this, let the noise part $f(\mathbf{x}, \mathbf{y})$, $f(\mathbf{y}, \mathbf{x})$ be the vectors $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2$. Then (using canonic embeddings of $\mathbf{x}, \mathbf{y} \in \mathbb{R}^\ell$ inside $\mathbb{R}^h$), $\tilde{d}(\mathbf{x}, \mathbf{y}) = \|(\mathbf{x}, f(\mathbf{x}, \mathbf{y})) - (\mathbf{y}, f(\mathbf{y}, \mathbf{x}))\|_Q = \|\mathbf{x} - \mathbf{y} + (\boldsymbol{\epsilon}_1 - \boldsymbol{\epsilon}_2)\|_Q$ and by the converse triangle inequality, we get

$$\begin{aligned} |\tilde{d}(\mathbf{x}, \mathbf{y}) - \|\mathbf{x} - \mathbf{y}\|_Q| &\leq \left\|\tilde{d}(\mathbf{x}, \mathbf{y}) - (\mathbf{x} - \mathbf{y})\right\|_Q \\ &= \|(\mathbf{x} - \mathbf{y}) + (\boldsymbol{\epsilon}_1 - \boldsymbol{\epsilon}_2) - (\mathbf{x} - \mathbf{y})\|_Q \\ &= \|\boldsymbol{\epsilon}_1 - \boldsymbol{\epsilon}_2\|_Q \leq \|\boldsymbol{\epsilon}_1 - \boldsymbol{\epsilon}_2\|_2 \leq 2\frac{\varepsilon}{6} = \frac{\varepsilon}{3}, \end{aligned} \tag{13}$$

using the bound $\|\boldsymbol{\epsilon}_1\|_2 = \|f(\mathbf{x}, \mathbf{y})\|_2 = |f(\mathbf{x}, \mathbf{y})| \leq \varepsilon/3$ that likewise holds for $\boldsymbol{\epsilon}_2$. Consider the metric $d(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\|_Q$ on $\mathbb{R}^\ell$, using the canonic embedding the points $\mathbf{x}, \mathbf{y}$ into $\mathbb{R}^h$. This metric satisfies the triangle inequality and is no more than $\varepsilon/3$ different from $\tilde{d}$, which will establish the claim:

$$\begin{aligned} \tilde{d}(\mathbf{x}, \mathbf{y}) &\overset{(13)}{\leq} d(\mathbf{x}, \mathbf{y}) + \frac{\varepsilon}{3} \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) + \frac{\varepsilon}{3} \\ &\overset{(13)}{\leq} \tilde{d}(\mathbf{x}, \mathbf{z}) + \frac{\varepsilon}{3} + \tilde{d}(\mathbf{z}, \mathbf{y}) + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} \\ &= \tilde{d}(\mathbf{x}, \mathbf{z}) + \tilde{d}(\mathbf{z}, \mathbf{y}) + \varepsilon. \end{aligned}$$

□

Our practical evaluation in section VI combined the choice of neighboring points with the definition of the noise function f, by seeding an available pseudorandom number generator with a value that depends non-symmetrically on both $\mathbf{y}_i, \mathbf{y}_j$ (i.e., provides different seeds for $(\mathbf{y}_i, \mathbf{y}_j)$ than for $(\mathbf{y}_j, \mathbf{y}_i)$), and which produced a pseudorandom uniform distribution in $[0, 1]$. This delivers the desired neighbors $\mathbf{z}_{i,j}$ and $\mathbf{z}_{j,i}$ while achieving linear independence with high probability. Scaling the pseudorandom noise down by a factor $0 < s < 0.9 \cdot \frac{\varepsilon}{\cdot\sqrt{h}}$ accomplished bound (11).

D. Experimental Verification of Distances

Table VII shows the results of constructing a norm as in the proof of Theorem 1 for 10 arbitrarily chosen points in $\mathbb{R}^4$ with distances that were assigned stochastically independent of the point locations

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their invaluable and inspiring comments on earlier versions of the manuscript.

REFERENCES

[1] Eur. Comission. (Nov. 2023). *Regulatory Framework Proposal on Artificial Intelligence—Shaping Europe's Digital Future*. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

[2] Eur. Comission. (Nov. 2023). *The Digital Services Act Package—Shaping Europe's Digital Future*. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/digital-services-act-package

[3] B. Biggio, I. Pillai, S. R. Bulò, D. Ariu, M. Pelillo, and F. Roli, "Is data clustering in adversarial settings secure?" in *Proc. ACM Workshop Artif. Intell. Secur.* New York, NY, USA: ACM, Nov. 2013, pp. 87–98.

[4] A. E. Cinà, A. Torcinovich, and M. Pelillo, "A black-box adversarial attack for poisoning clustering," *Pattern Recognit.*, vol. 122, Feb. 2022, Art. no. 108306. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320321004866

[5] Y. Xu, X. Wei, P. Dai, and X. Cao, "A2SC: Adversarial attacks on subspace clustering," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 19, no. 6, pp. 191:1–191:23, Jul. 2023, doi: 10.1145/3587097.

[6] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.* New York, NY, USA: ACM, 2011, pp. 43–58, doi: 10.1145/2046684.2046692.

[7] Y. Chen et al., "Practical attacks against graph-based clustering," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Oct. 2017, pp. 1125–1142, doi: 10.1145/3133956.3134083.

[8] X. Li, B. Kao, S. Luo, and M. Ester, "ROSC: Robust spectral clustering on multi-scale data," in *Proc. World Wide Web Conf. World Wide Web*, Apr. 2018, pp. 157–166, doi: 0.1145/3178876.3185993.

[9] E. Hohma, C. M. M. Frey, A. Beer, and T. Seidl, "SCAR: Spectral clustering accelerated and robustified," *Proc. VLDB Endowment*, vol. 15, no. 11, pp. 3031–3044, Jul. 2022, doi: 10.14778/3551793.3551850.

[10] A. Bojchevski, Y. Matkovic, and S. Günnemann, "Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2017, pp. 737–746, doi: 10.1145/3097983.3098156.

[11] D. Chicco, "Siamese neural networks: An overview," in *Artificial Neural Networks* (Methods in Molecular Biology), H. Cartwright, Ed., New York, NY, USA: Springer, 2021, pp. 73–94, doi: 10.1007/978-1-0716-0826-5_3.

[12] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," 2017, *arXiv:1712.07107*.

[13] S. Rass, S. König, S. Ahmad, and M. Goman, "Metricizing the Euclidean space towards desired distance relations in point clouds," 2022, *arXiv:2211.03674*.

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 5. Berkeley, CA, USA: Univ. of California Press, Jan. 1967, pp. 281–298.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*. Portland, OR, USA: AAAI Press, Aug. 1996, pp. 226–231.

[16] N. Rekabsaz, S. Kopeinik, and M. Schedl, "Societal biases in retrieved contents: Measurement framework and adversarial mitigation of BERT rankers," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 306–316.

[17] M. Schedl, D. Hauger, K. Farrahi, and M. Tkalčič, "On the influence of user characteristics on music recommendation algorithms," in *Advances in Information Retrieval*. Wiesbaden, Germany: Springer, 2015, pp. 339–345.

[18] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation," in *Proc. Workshop Recommendation Multi-Stakeholder Environments RecSys*, vol. 2440. Aachen, Germany, 2019. [Online]. Available: http://ceur-ws.org/Vol-2440/paper4.pdf

[19] (2023). *Stop the AMS Algorithm*. [Online]. Available: https://amsalgorithmus.at/en/

[20] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–35, Jul. 2022.

[21] X.-H. Wu, "Noise clustering using a new distance," in *Proc. IEEE Int. Conf. Inf. Acquisition*, Aug. 2006, pp. 1015–1020.

[22] J. Gao, R. E. Carrillo, and K. E. Barner, "LLp metric based robust clustering," in *Proc. 43rd Annu. Conf. Inf. Sci. Syst.*, Mar. 2009, pp. 747–750.

[23] R. Zhang, H. Tong, Y. Xia, and Y. Zhu, "Robust embedded deep K-means clustering," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: ACM, Nov. 2019, pp. 1181–1190.

[24] S. B. Shah, P. Raval, H. Khakhi, and M. S. Raval, "Frequency centric defense mechanisms against adversarial examples," in *Proc. 1st Int. Workshop Adversarial Learn. Multimedia*. New York, NY, USA: ACM, Oct. 2021, pp. 62–67.

[25] Y. Chen, L. Cai, W. Cheng, and H. Wang, "Super-resolution coding defense against adversarial examples," in *Proc. Int. Conf. Multimedia Retr.* New York, NY, USA: ACM, Jun. 2020, pp. 189–197.

[26] A. Aydin, D. Sen, B. T. Karli, O. Hanoglu, and A. Temizel, "Imperceptible adversarial examples by spatial chroma-shift," in *Proc. 1st Int. Workshop Adversarial Learn. Multimedia*. New York, NY, USA: ACM, Oct. 2021, pp. 8–14.

[27] A. Beer, D. Kazempour, L. Stephan, and T. Seidl, "LUCK- linear correlation clustering using cluster algorithms and a kNN based distance function," in *Proc. 31st Int. Conf. Scientific Stat. Database Manage.*, Jul. 2019, pp. 181–184.

[28] U. von Luxburg, "A tutorial on spectral clustering," 2007, *arXiv:0711.0189*.

[29] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2004, pp. 59–68, doi: 10.1145/1014052.1014062.

[30] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA: ACM, 2004, p. 11, doi: 10.1145/1015330.1015360.

[31] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering," in *Proc. 19th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann, Jul. 2002, pp. 307–314.

[32] V. Chatziafratis, R. Niazadeh, and M. Charikar, "Hierarchical clustering with structural constraints," 2018, *arXiv:1805.09476*.

[33] C. Böhm and C. Plant, "HISSCLU: A hierarchical density-based method for semi-supervised clustering," in *Proc. 11th Int. Conf. Extending Database Technol., Adv. Database Technol.* New York, NY, USA: ACM, Mar. 2008, pp. 440–451, doi: 10.1145/1353343.1353398.

[34] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-means clustering with background knowledge," in *Proc. Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 577–584.

[35] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2003, pp. 39–48, doi: 10.1145/956750.956759.

[36] E. Xing, M. Jordan, S. J. Russell, and A. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15, S. Becker, S. Thrun, and K. Obermayer, Eds., Cambridge, MA, USA: MIT Press, 2002. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2002/file/c3e4035af2a1cde9f21e1ae1951ac80b-Paper.pdf

[37] W. Wei, B. Xi, and M. Kantarcioglu, "Adversarial clustering: A grid based clustering algorithm against active adversaries," 2018, *arXiv:1804.04780*.

[38] J. Cai, J. Hao, H. Yang, X. Zhao, and Y. Yang, "A review on semi-supervised clustering," *Inf. Sci.*, vol. 632, pp. 164–200, Jun. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025523002840

[39] H. Wang, W. Yang, J. Wang, R. Wang, L. Lan, and M. Geng, "Pairwise similarity regularization for adversarial domain adaptation," in *Proc. 28th ACM Int. Conf. Multimedia*. New York, NY, USA: ACM, Oct. 2020, pp. 2409–2418, doi: 10.1145/3394171.3413516.

[40] H. Tang, Y. Wang, and K. Jia, "Unsupervised domain adaptation via distilled discriminative clustering," 2023, *arXiv:2302.11984*.

[41] J. Li, G. Li, Y. Shi, and Y. Yu, "Cross-domain adaptive clustering for semi-supervised domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2505–2514. [Online]. Available: https://www.computer.org/csdl/proceedings-article/cvpr/2021/450900c505/1yeKkecruGQ

[42] S. Pasteris, F. Vitale, C. Gentile, and M. Herbster, "On similarity prediction and pairwise clustering," in *Proc. 29th Int. Conf. Algorithmic Learn. Theory*, Apr. 2018, pp. 1–28.

[43] T. Di Noia, N. Tintarev, P. Fatourou, and M. Schedl, "Recommender systems under European AI regulations," *Commun. ACM*, vol. 65, no. 4, pp. 69–73, Mar. 2022, doi: 10.1145/3512728.

[44] M. Schedl, N. Rekabsaz, E. Lex, T. Grosz, and E. Greif, "Multiperspective and multidisciplinary treatment of fairness in recommender systems research," in *Adjunct Proc. 30th ACM Conf. User Model., Adaptation Personalization*, Jul. 2022, pp. 90–94, doi: 10.1145/3511047.3536400.

[45] R. Wang, F. M. Harper, and H. Zhu, "Factors influencing perceived fairness in algorithmic decision-making: Algorithm outcomes, development procedures, and individual differences," in *Proc. CHI Conf. Human Factors Comput. Syst.*, Apr. 2020, pp. 1–14, doi: 10.1145/3313831.3376813.

[46] G. Dai, J. Xie, and Y. Fang, "Metric-based generative adversarial network," in *Proc. 25th ACM Int. Conf. Multimedia*. New York, NY, USA: ACM, Oct. 2017, pp. 672–680.

[47] T.-T. Zhang and B. Yuan, "Density-based multiscale analysis for clustering in strong noise settings with varying densities," *IEEE Access*, vol. 6, pp. 25861–25873, 2018.

[48] M. I. Ostrovskii, "Metric embeddings: Bilipschitz and coarse embeddings into Banach spaces," in *Metric Embeddings*. Berlin, De Gruyter, Jun. 2013. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/9783110264012/html?lang=de

[49] Y. Deldjoo, T. D. Noia, and F. A. Merra, "A survey on adversarial recommender systems: From attack/defense strategies to generative adversarial networks," *ACM Comput. Surv.*, vol. 54, no. 2, p. 35, Mar. 2021.

[50] Y. Xie, S. Shekhar, and Y. Li, "Statistically-robust clustering techniques for mapping spatial hotspots: A survey," *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–38, Jan. 2022.

[51] G. Liu, I. Khalil, and A. Khreishah, "Using single-step adversarial training to defend iterative adversarial examples," in *Proc. ACM Conf. Data Appl. Security Privacy*. New York, NY, USA: ACM, Apr. 2021, pp. 17–27.

[52] Q. Zhang, Y. Ding, Y. Tian, J. Guo, M. Yuan, and Y. Jiang, "AdvDoor: Adversarial backdoor attack of deep learning system," in *Proc. 30th ACM SIGSOFT Int. Symp. Softw. Test. Anal.* New York, NY, USA: ACM, Jul. 2021, pp. 127–138.

[53] A. José-García and W. Gómez-Flores, "A survey of cluster validity indices for automatic data clustering using differential evolution," in *Proc. Genetic Evol. Comput. Conf.* New York, NY, USA: ACM, Jun. 2021, pp. 314–322.
[54] R. Krishnamoorthy and S. S. Kumar, "A new inter cluster validation method for unsupervised clustering techniques," in *Proc. Int. Conf. Commun. Comput. Vis. (ICCCV)*, Dec. 2013, pp. 1–5.
[55] Y. Wang, J. Zheng, X. Liu, and B. Lang, "Robust clustering for social annotated images," in *Proc. 5th Int. Conf. Internet Multimedia Comput. Service*. New York, NY, USA: ACM, Aug. 2013, pp. 87–92.
[56] L. McInnes, J. Healy, and S. Astels. (2016). *Comparing Python Clustering Algorithms—HDBScan 0.8.1 Documentation*. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html
[57] I. J. Schoenberg, "Remarks to maurice Frechet's article 'sur la definition axiomatique d'une classe d'espace distances vectoriellement applicable sur l'espace de Hilbert," *Ann. Math.*, vol. 36, no. 3, p. 724, Jul. 1935.
[58] L. A. Steen and J. A. Seebach, *Counterexamples in Topology*. New York, NY, USA: Dover, 1995.
[59] B. Mendelson, *Introduction to Topology*, 3rd ed. New York, NY, USA: Dover, 1990.
[60] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. (2022). *GNU Octave Version 7.1.0 Manual: A High-Level Interactive Language for Numerical Computations*. [Online]. Available: https://www.gnu.org/software/octave/doc/v7.1.0/
[61] S. Ghatak. (Feb. 2017). *DBSCAN*. Accessed: Oct. 4, 2022. [Online]. Available: https://github.com/devil1993/DBSCAN;snapshot downloaded/saved
[62] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for K-means clustering," *Pattern Recognit. Lett.*, vol. 25, no. 11, pp. 1293–1302, Aug. 2004.
[63] K. K. Saab, "Estimation of cluster centroids in presence of noisy observations," in *Proc. IEEE MIT Undergraduate Res. Technol. Conf. (URTC)*, Nov. 2016, pp. 1–4.
[64] E. W. Weisstein, *Gershgorin Circle Theorem*. Champaign, IL, USA: Wolfram Research, 2023. [Online]. Available: https://mathworld.wolfram.com/

Stefan Rass (Member, IEEE) received the degree in mathematics and computer science from Universitaet Klagenfurt (AAU). Currently, he is a Full Professor with Johannes Kepler University (JKU) Linz, Austria, and a member of the Secure and Correct Systems Laboratory. He has participated in various nationally and internationally funded research projects, being a contributing researcher in many EU projects, and offering consultancy services to the industry. He has authored numerous articles related to practical security, security infrastructures, robot security, and applied statistics and decision theory in security. His research interests include decision theory and game theory with applications in system security, especially robotics security, complexity theory, statistics, and information-theoretic security.

Sandra König received the bachelor's and master's degrees in mathematics from ETH Zürich and the Ph.D. degree (Hons.) in mathematics with Alpen-Adria-Universität Klagenfurt. She is currently a Researcher with Research Driven Solutions (RDS) and a Lecturer with the University of Zürich. She has been involved in several Austrian and EU projects dealing with the protection of critical infrastructures and the safety and security of autonomous systems. She authored numerous articles related to risk estimation and risk mitigation in security, applied statistics, machine learning, and game theory. Her research interests include probabilistic risk models and game theoretic risk mitigation methods. Beyond this, she is interested in machine learning techniques and application in domains, such as logistics and life cycle assessment.

Shahzad Ahmad received the bachelor's degree in electronics engineering from the Harcourt Butler Technological Institute, Kanpur, India, in 2016, and the master's degree in electronics engineering (communication and information systems) from AMU, Aligarh, India, in 2021. He is currently pursuing the Ph.D. degree in computer science from the LIT Secure and Correct Systems Laboratory, Johannes Kepler University Linz, Austria. His research interests include plausible deniability, cloud security, and signal processing.

Maksim Goman received the master's degree in computer science from TU Ilmenau and the Ph.D. degree in business informatics from JKU Linz. He is currently a Researcher with JKU Linz, with a focus on IT security and audit and risk management. His research interests include the quantitative evaluation of technological risk, the analysis of stochastic networks, and the methods of decision support for management.