Data Science and Engineering (2022) 7:120–135
https://doi.org/10.1007/s41019-022-00185-5

RESEARCH PAPERS

A New Dimensionality-Unbiased Score for Efficient and Effective Outlying Aspect Mining

Durgesh Samariya[1] · Jiangang Ma[1]

Received: 13 December 2021 / Revised: 9 March 2022 / Accepted: 5 April 2022 / Published online: 29 April 2022

Abstract
The main aim of the outlying aspect mining algorithm is to automatically detect the subspace(s) (a.k.a. aspect(s)), where a given data point is dramatically different than the rest of the data in each of those subspace(s) (aspect(s)). To rank the subspaces for a given data point, a scoring measure is required to compute the outlying degree of the given data in each subspace. In this paper, we introduce a new measure to compute outlying degree, called *Simple Isolation score using Nearest Neighbor Ensemble* (SiNNE), which not only detects the outliers but also provides an explanation on why the selected point is an outlier. SiNNE is a dimensionally unbias measure in its raw form, which means the scores produced by SiNNE are compared directly with subspaces having different dimensions. Thus, it does not require any normalization to make the score unbiased. Our experimental results on synthetic and publicly available real-world datasets revealed that (i) SiNNE produces better or at least the same results as existing scores. (ii) It improves the run time of the existing outlying aspect mining algorithm based on beam search by at least two orders of magnitude. SiNNE allows the existing outlying aspect mining algorithm to run in datasets with hundreds of thousands of instances and thousands of dimensions which was not possible before.

Keywords Outlying aspect mining · Isolation based · Outlying degree · Subspace search

1 Introduction

Outliers (a.k.a anomalies) are data points that show dramatically different behavior from the remainder of data points in the dataset. The process of finding such data points is known as *Outlier Detection* (OD). In the era of big data, OD is considered as one of the vital task of data mining with a wide range of application domains [21], i.e., (i) fraud detection—in this domain, outlier refers to the fraud that includes credit card frauds [6], insurance claim frauds [4]; (ii) Medical or public health—in this domain, outlier refers to an unusual health condition of patients that happens due to instrumental error or disease symptoms [14].

Recently, researchers have been interested in the explanation of why the data point is considered as an outlier. The problem of finding these explanations leads to the *Outlying Aspect Mining* (OAM) [8, 22, 27, 28]. OAM is the task of identifying feature subset(s), in which a given data point is dramatically inconsistent with the rest of the data. In literature, the problem of OAM is also referred as *outlying subspace detection* [31], *outlier explanation* [9, 17, 18], *outlier interpretation* [7, 16, 29], *outlying property detection* [1] and *outlying aspect mining* [8, 22, 23, 26–28, 30].

In many application scenarios, it is required to find out in which set of feature(s), a given point is different than others. For example, in a bank, a fraud analyst collects information about various aspects of credit card fraud, and he/she is interested to know in which aspects the fraud does not conform with the remainder of that set of data. Moreover, when evaluating job applications, a panel member wants to know the job applicant's unique features. Another exciting application of OAM is in the medical domain [20]. Assume that you are a doctor and while treating a specific patient, you want to know, how this patient is different than others. Existing OD methods cannot answer all these questions.

To detect outlying aspects, OAM algorithms require a scoring measure to rank subspaces based on the outlying degrees of the given query. Existing OAM algorithms such

✉ Durgesh Samariya
d.samariya@federation.edu.au

Jiangang Ma
j.ma@federation.edu.au

1 School of Engineering, Information Technology and Physical Sciences, Federation University, Berwick, VIC, Australia

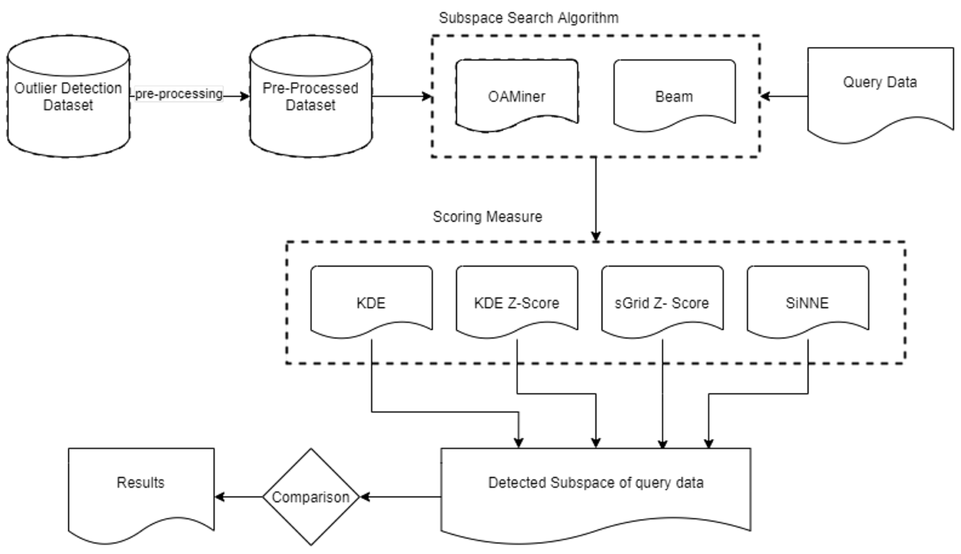


Fig. 1 The high-level process pipeline

as HOSMiner [31], OAMiner [8], Density *Z*-Score [27] and sGrid [28] use a traditional distance or density-based outlier score as the ranking measure. Because distance or density-based outlier scores depend on the dimensionality of subspaces, they cannot be compared directly to rank subspaces. [27] proposed to use *Z*-Score normalization to make them comparable. It requires computing the outlier scores of all the data points in each subspace. It adds significant computational overhead making OAM algorithms infeasible to run in large and/or high-dimensional datasets. Also, we discover that *Z*-Score normalization is not appropriate for OAM in some cases.

In this paper, we focus on the two issues of existing scores used in OAM: (i) dimensionality unbiasedness, and (ii) computational complexity. It is worth noting that another computational issue in OAM is to deal with the exponentially large number of subspaces. Current OAM methods perform a systematic search; which is computationally prohibitive when the number of dimensions is high. This paper does not deal with this computational issue. It still uses the existing systematic search approach but deals with computing the score in each subspace efficiently.

This paper makes the following contributions:

– Identify an issue of using *Z*-Score normalization of density-based outlier scores to rank subspaces and shows that it is biased towards a subspace having high-density variance.
– Propose a new simple measure called *Simple Isolation score using Nearest Neighbor Ensemble* (SiNNE), which is useful for detecting outliers from the dataset and outlying aspects of the given outlier points.
– Provide an objective measure to assess the quality of discovered outlying subspaces.
– Validate the effectiveness and efficiency of SiNNE in OAM. Our empirical results show that SiNNE can detect more interesting outlying aspects than the existing score, and it allows the OAM algorithm to run orders of magnitude faster than the existing scoring measure.

The rest of the paper is organized as follows. Section 2 provides a summary of previous work on outlying aspect mining. The proposed outlier detector scoring measure is presented in Sect. 3. Experimental settings are provided in Sect. 4, and empirical evaluation results are provided in Sect. 5. Finally, conclusions are provided in Sect. 6.

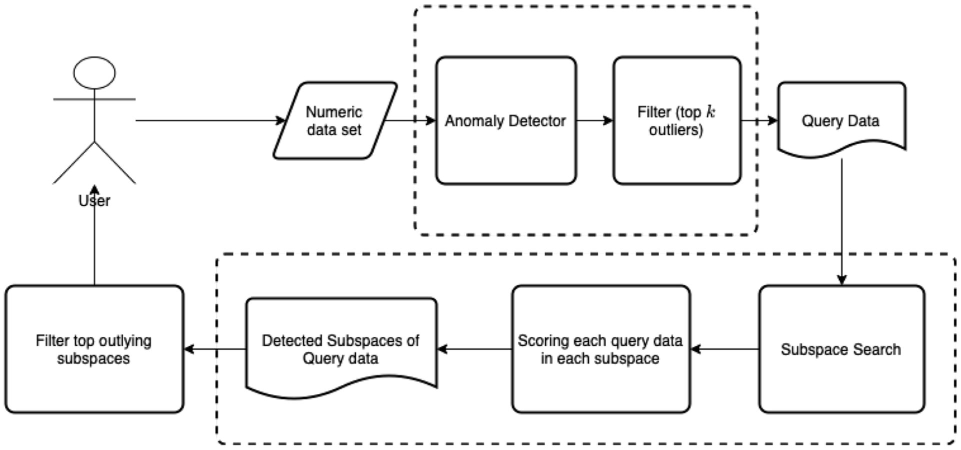
2 Related Works

In this section, first, we fixed some notations for the rest of the paper, provided some basic definitions, and then discussed recent outlying aspect mining methods. The high-level process pipeline of OAM is shown in Fig. 1.

Table 1 Key symbols and notations

Symbol	Definition
$\mathcal{X}$	A dataset with d attributes, where $\lvert\mathcal{X}\rvert = n$
x	A data point in $\mathcal{X}$
$\mathcal{F}$	Set of features, where $\mathcal{F} = \{F_1, F_2, \dots, F_d\}$
$\mathbf{q}$	A query point
$\mathbb{S}$	The set of all possible subspaces
ψ	The number of sub-samples
t	The number of sets
$\mathcal{D}$	A subset of $\mathcal{X}$, $\mathcal{D} \subset \mathcal{X}, \lvert\mathcal{D}\rvert = \psi$

Fig. 2 The flowchart


User
Numeric data set
Anomaly Detector
Filter (top k outliers)
Query Data
Subspace Search
Scoring each query data in each subspace
Detected Subspaces of Query data
Filter top outlying subspaces

2.1 Basic Notations and Definitions

Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a collection of n data points in an d-dimensional space $\mathfrak{R}$, where $\mathfrak{R}$ is a real domain. Each data point x is represented as an d dimensional vector $\langle x^{(1)}, x^{(2)}, \dots, x^{(d)} \rangle$. Let $\mathcal{F}$ be a full feature space and $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_\eth\}$ be a set of all possible subspaces, where $\eth = 2^d - 1$ is the number of possible subspaces. The key symbols and notations used in this paper are provided in Table 1.

The problem of outlier detection is to identify all x_i which remarkably deviates from others in full feature set $\mathcal{F}$, whereas the problem of outlying aspect mining is to identify subspace $\mathcal{S}_i \in \mathbb{S}$, where the given data point $x_i \in \mathcal{X}$ is significantly different from the rest of the data. That given data point $x_i \in \mathcal{X}$ is referred as a query $\mathbf{q}$.

Definition 1 (*Outlier*) An outlier is a data instance that significantly deviates from others in the full feature set $\mathcal{F}$.

Definition 2 (*Subspace*) A subspace is a subset of the dimensions d of dataset $\mathcal{X}$.

Definition 3 (*Query point*) A query $\mathbf{q}$ is a data point of interest, which is used to find outlying aspects.

Definition 4 (*Problem definition*) Given a set of n instances $\mathcal{X}$ ($|\mathcal{X}| = n$) in d dimensional space, a query $\mathbf{q} \in \mathcal{X}$, a subspace $\mathcal{S}$ is called outlying aspect of $\mathbf{q}$ iff,

– outlying degree of $\mathbf{q}$ in subspace $\mathcal{S}$ is higher than other subspaces, and there is no other subspace with same or higher outlying degree.

2.2 Outlying Aspect Mining

To the best of our knowledge, [31] is the earliest work that defines the problem of OAM. They introduced a framework to detect an outlying subspace called HOS-Miner (stands for High-dimensional Outlying Subspace Miner). Therein, the author used a distance-based measure called Outlying Degree (*OutD* in short). The *OutD* of query $\mathbf{q}$ in subspace $\mathcal{S}$ is computed as:

$$OutD_{\mathcal{S}}(\mathbf{q}) = \sum_{x \in \aleph^k_{\mathcal{S}}(\mathbf{q})} d_{\mathcal{S}}(\mathbf{q}, x)$$

where $\aleph^k_{\mathcal{S}}(\mathbf{q})$ is a set of k-nearest neighbors of $\mathbf{q}$ in subspace $\mathcal{S}$, $d_{\mathcal{S}}(a, b)$ is an euclidean distance between a and b in subspace $\mathcal{S}$, which is computed as $d_{\mathcal{S}}(a, b) = \sqrt{\sum_{i \in \mathcal{S}} (a_i - b_i)^2}$.

In 2015, [8] introduced Outlying Aspect Miner (OAMiner in short). Instead of using distance, therein, authors employed a kernel density estimation [24]-based scoring measure to compute the outlyingness of query $\mathbf{q}$ in subspace $\mathcal{S}$:

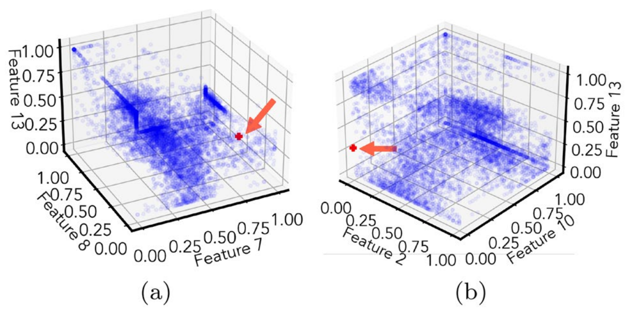


Fig. 3 Data distribution in two three-dimensional subspaces of the Pendigits dataset. **a** $\tilde{f}_{\mathcal{S}_i}(\mathbf{q}) = 21.30, Z(\tilde{f}_{\mathcal{S}_i}(\mathbf{q})) = -2.10$; **b** $\tilde{f}_{\mathcal{S}_j}(\mathbf{q}) = 1.20, Z(\tilde{f}_{\mathcal{S}_j}(\mathbf{q})) = -1.25$

$$\tilde{f}_{\mathcal{S}}(\mathbf{q}) = \frac{1}{n(2\pi)^{\frac{m}{2}} \prod_{i\in\mathcal{S}} h_i} \sum_{x\in\mathcal{X}} e^{-\sum_{i\in\mathcal{S}} \frac{(\mathbf{q}_i - x_i)^2}{2h_i^2}}$$

where $\tilde{f}_{\mathcal{S}}(\mathbf{q})$ is a kernel density estimation of $\mathbf{q}$ in subspace $\mathcal{S}$, m is the dimensionality of subspace $\mathcal{S}$ ($m = |\mathcal{S}|$), h_i is the kernel bandwidth in dimension i.

[8] stated that $\tilde{f}_{\mathcal{S}}$ is bias towards high-dimensional subspaces—density tends to decrease as dimension increases. Thus, to remove the effect of dimensionality biasedness, they proposed to use the density rank of the query as a measure of outlyingness.

[27] proposed two outlying scoring metrics (i) density Z-Score and (ii) iPath score (stands for isolation Path).

Therein, the density Z-Score is defined as follows:

$$\text{Z-Score}(\tilde{f}_{\mathcal{S}}(\mathbf{q})) \triangleq \frac{\tilde{f}_{\mathcal{S}}(\mathbf{q}) - \mu_{\tilde{f}_{\mathcal{S}}}}{\sigma_{\tilde{f}_{\mathcal{S}}}}$$

where $\mu_{\tilde{f}_{\mathcal{S}}}$ and $\sigma_{\tilde{f}_{\mathcal{S}}}$ are the mean and standard deviation of the density of all data instances in subspace $\mathcal{S}$, respectively.

The iPath score is motivated by Isolation Forest (iForest) anomaly detection approach [15]. The process of calculating the iPath score in subspace $\mathcal{S}$ of query $\mathbf{q}$ w.r.t. sub-samples ψ of the data is:

$$iPath_{\mathcal{S}}(\mathbf{q}) = \frac{1}{t}\sum_{i=1}^{t} l^i_{\mathcal{S}}(\mathbf{q})$$

where $l^i_{\mathcal{S}}(\mathbf{q})$ is path length of $\mathbf{q}$ in i^{th} tree and subspace $\mathcal{S}$.

[27] were the first to coin the term dimensionality unbiasedness, i.e., "A dimensionality unbiased outlyingness measure (OM) is a measure of which the baseline value, i.e., average value for any data sample $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ drawn from a uniform distribution, is a quantity independent of the dimension of the subspace $\mathcal{S}$."

[28] introduced a simple grid-based density estimator called sGrid. sGrid is a smoothed variant of a grid-based density estimator [24]. Let $\mathcal{X}$ be a collection of n data objects in d-dimensional space, $x.\mathcal{S}$ be a projection of a data object $x \in \mathcal{X}$ in subspace $\mathcal{S}$. The sGrid density of point $\mathbf{q}$ is computed as the number of points that falls into a bin that covers point $\mathbf{q}$ and its surrounding neighbors. In their work, they show that the proposed density estimator has advantages over the existing kernel density estimator in outlying aspect mining by replacing the kernel density estimator with sGrid.

In recent work, [30] proposed a reconstruction-based method using completely random trees (RecForest in short). Therein, reconstruction has been done using the intersection of the bounding boxes in the completely random forest for each data point. The outlying score OS of each feature $i = 1, 2, \dots, d$ for query $\mathbf{q}$ is defined as:

$$OS_i = \frac{\exp(\mathbf{q}_i - \mathbf{q}_i^{rec})^2}{\sum_{j=1}^{d} \exp(\mathbf{q}_j - \mathbf{q}_j^{rec})^2}$$

where $\mathbf{q}^{rec}$ is a reconstructed sample of $\mathbf{q}$.

[29] proposed an Attention-guided Triplet deviation network for Outlier interpretatioN (ATON). Instead of searching subspaces, ATON learns an embedding space and learns how each dimension is contributing to the outlyingness of the query.

3 The Framework

We first outline the motivation for our method, followed by the details of SiNNE. Figure 2 presents the flowchart of the complete framework.

3.1 Issue of Using *Z*-Score

Because Z-Score normalization uses mean and variance of density values of all data instances in a subspace ($\mu_{\tilde{f}_{\mathcal{S}_i}}$ and $\sigma_{\tilde{f}_{\mathcal{S}_i}}$), it can be biased towards a subspace having high variation of density values (i.e., high $\sigma_{\tilde{f}_{\mathcal{S}_i}}$).

Let's take a simple example to demonstrate this. Assume that $\mathcal{S}_i$ and $\mathcal{S}_j$ ($i \neq j$), be two different subspaces of the same dimensionality (i.e., $|\mathcal{S}_i| = |\mathcal{S}_j|$). Intuitively, because they have the same dimensionality, they can be ranked based on the raw density (unnormalized) values of a query $\mathbf{q}$. Assuming $\mu_{\tilde{f}_{\mathcal{S}_i}} = \mu_{\tilde{f}_{\mathcal{S}_j}}$, we can have $Z(\tilde{f}_{\mathcal{S}_i}(\mathbf{q})) < Z(\tilde{f}_{\mathcal{S}_j}(\mathbf{q}))$ even though $\tilde{f}_{\mathcal{S}_i}(\mathbf{q}) = \tilde{f}_{\mathcal{S}_j}(\mathbf{q})$ if $\sigma_{\tilde{f}_{\mathcal{S}_i}} > \sigma_{\tilde{f}_{\mathcal{S}_j}}$ (i.e., $\mathcal{S}_i$ is ranked higher than $\mathcal{S}_j$ based on density Z-Score normalization just because of higher $\sigma_{\tilde{f}_{\mathcal{S}_i}}$).

To show this effect in a real-world dataset, let's take an example of the *pendigits*[1] dataset ($n = 9868$ and $d = 16$). Figure 3 shows the distribution of data in two three-dimensional subspaces $\mathcal{S}_i = \{7, 8, 13\}$ and $\mathcal{S}_j = \{2, 10, 13\}$. Visually, the query $\mathbf{q}$ represented by the red square appears to be more outlier in $\mathcal{S}_j$ than in $\mathcal{S}_i$. This is consistent with its raw density values in the two subspaces, $\tilde{f}_{\mathcal{S}_j}(\mathbf{q}) = 1.20 < \tilde{f}_{\mathcal{S}_i}(\mathbf{q}) = 21.30$. However, the ranking is reversed after the Z-Score normalization, ($Z(\tilde{f}_{\mathcal{S}_j}(\mathbf{q})) = -1.25 > Z(\tilde{f}_{\mathcal{S}_i}(\mathbf{q})) = -2.10$). This is due to the higher $\sigma_{\tilde{f}_{\mathcal{S}_i}} = 57.3 > \sigma_{\tilde{f}_{\mathcal{S}_j}} = 34.2$.

Apart from these, existing OAM scoring measures have two limitations:

– they are dimensionally biased and they require normalization; and

[1] Available at https://elki-project.github.io/datasets/outlier.

– they are expensive to compute in each subspace.

Being motivated by these limitations of density-based scores in OAM, we introduce a new measure which is dimensionally unbias in its raw form and can be computed efficiently.

3.2 Outlierness Computation

We now introduce a new scoring measure called simple isolation using nearest-neighbor ensembles (**SiNNE** in short). This scoring function is inspired by the isolation-based anomaly detection using nearest-neighbor ensembles [2, 3].

The proposed scoring function has two major steps:

- Building hyperspheres: The process of building hyperspheres in each subspace. The hyperspheres are build using nearest neighbors.
- Scoring query: The current model is used to score the query.

3.2.1 Build Model

Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a dataset $x_i \in \mathfrak{R}^d$, where $i \in n$ represents the position of data point x in $\mathcal{X}$, n is the number of data points in the dataset and d is the number of dimensions. We randomly choose ψ data samples from $\mathcal{X}$, t times in each subspace.

Our proposed scoring function follows same procedure as the iNNE [2] to build ensemble of hyperspheres. However, in context of OAM, the difference is that we create ensembles in subspaces instead of full feature space.

Basically, SiNNE creates an ensemble of hyperspheres. Ensemble is defined as t sets of hyperspheres, where each set consists of ψ hyperspheres.

Definition 5 (*Hyperspheres*) Given data subset $\mathcal{D}_i^{(\psi)}$, a hypersphere $\mathsf{H}(c)$ centered at c with radii $\tau(c) = ||c - \eta_c||$ is defined as $\{x : ||x - c|| \leq \tau(c)\}$, where $x \in \mathfrak{R}^d$ and $c, \eta_c \in \mathcal{D}_i^{(\psi)}$; η_c is the nearest neighbor of c in $\mathcal{D}_i^{(\psi)}$.

Definition 6 Given ψ sub-samples, an ensemble $\mathcal{H}$ contains t sets and each set consists of ψ hyperspheres. $\mathcal{H}$ is defined as:

$$\mathcal{H} = \{\{\mathsf{H}(c) : c \in \mathcal{D}_i^{(\psi)}\} : i = 1, 2, \dots, t\}$$

Note that the training process of SiNNE and iNNE is same, however, they differ in the computation of outlier score (cf. Sect. 3.5 for more differences).

Definition 7 (*Simple isolation score*) The simple isolation score of $\mathbf{q}$ in subspace $\mathcal{S}$ based on sub-sample $\mathcal{D}$ is defined as:

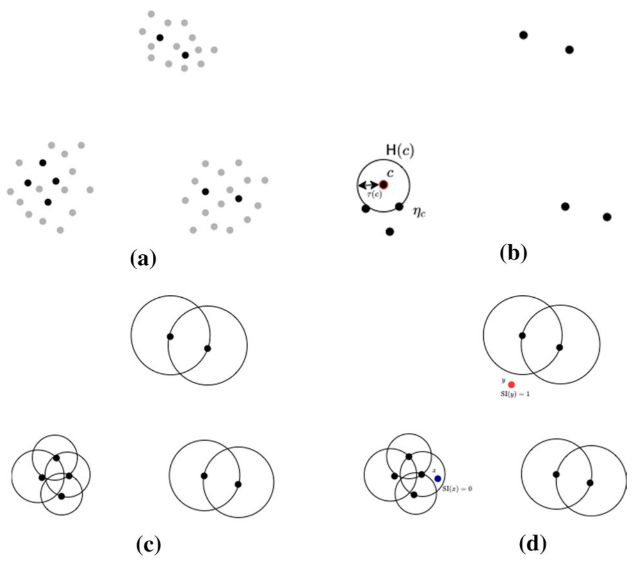


Fig. 4 **a** Randomly selected sub-samples $\mathcal{D}$ of size $\psi = 8$; **b** build hypersphere for data point c; **c** set of hyperspheres from $\mathcal{D}$; **d** simple isolation score for data point x and y using isolation model

$$\mathrm{SI}_{\mathcal{S}}(\mathbf{q}) = \mathbb{I}[\mathbf{q} \in \bigcup_{c \in \mathcal{D}} \mathsf{H}(c)] \tag{1}$$

where $\mathbb{I}[B]$ denotes the indicator function which gives the output 0 if B is true; otherwise $\mathbb{I}[B] = 1$.

SI takes the value either 0 or 1. When $\mathbf{q}$ is covered by any of the hypersphere, it assigns 0 and if it is not covered by any of the hypersphere then SiNNE assumes that point is far away from the data and assigns 1.

Definition 8 The outlier score for $\mathbf{q}$ in subspace $\mathcal{S}$ based on SiNNE is defined as the average of simple isolation score over t sets.

$$\mathrm{SiNNE}_{\mathcal{S}}(\mathbf{q}) = \frac{1}{t} \sum_{i=1}^{t} \mathrm{SI}_{\mathcal{S}}^{i}(\mathbf{q}) \tag{2}$$

As SI takes 0 or 1 score only, SiNNE($\mathbf{q}$) have score values in the range [0, 1].

Because the area covered by each hypersphere decreases as the dimensionality of the space increases and so is the actual data space covered by normal instances. Therefore, SiNNE is independent of the dimensionality of space in its raw form without any normalization making it ideal for OAM. It adapts to the local data density in the space because the sizes of the hyperspheres depend on the local density. It can be computed a lot faster than the k-NN distance or density. Also, it does not require to compute outlier scores of all n instances in each subspace (which is required with existing score for Z-Score normalization) which gives it a significant advantage in terms of run time.

The procedures to build an ensemble of models and using them to compute outlyingness of the given query data in subspace $\mathcal{S}$ are provided in Algorithms 1 and 2.

```
Algorithm 1: Build Hyperspheres ($\mathcal{X}, t, \psi$)
  Input: $\mathcal{X}$ - given data set; $t$ - number of sets, $\psi$ -
         number of sub-samples
  Output: $\mathcal{H}$ - An ensemble of $t$ sets of $\psi$
          hyperspheres
1 initialize $\mathcal{H} = \Phi$ ;
2 for $i \leftarrow 1$ to $t$ do
3     Generate $\mathcal{D}_i$ by randomly selecting $\psi$ data points
       from $\mathcal{X}$ without replacement ;
4     initialize $\mathcal{H}_i = 0$ ;
5     for $c \in \mathcal{D}_i$ do
6         $\eta(c|\mathcal{D}_i) \leftarrow$ The nearest neighbor of $c$ in $\mathcal{D}_i$ ;
7         $\mathsf{H}(c) \leftarrow$ Build a hypersphere centered at $c$
           with radius $||c - \eta(c|\mathcal{D}_i)||_2$ ;
8         $\mathcal{H}_i = \mathcal{H}_i \cup \mathsf{H}(c)$ ;
9     end
10    $\mathcal{H} = \mathcal{H} \cup \mathcal{H}_i$ ;
11 end
12 return $\mathcal{H}$;
```

```
Algorithm 2: SiNNE: Computing outlying-
ness of query
  Input: $\mathbf{q}$ - query point, $\mathcal{H}$ - $\{\mathcal{H}^i | i = 1, \ldots, t\}$
  Output: SiNNE($\mathbf{q}$)
1 initialize $si = 0$ ;
2 for $i \leftarrow 1$ to $t$ do
3     $si$ += search($\mathcal{H}_i, \mathbf{q}$) {return 0 if there is a
       hypersphere $\mathsf{H}$ that covers $\mathbf{q}$ in $\mathcal{H}_i$ else 1} ;
4 end
5 SiNNE($\mathbf{q}$) = $si/t$ ;
6 return SiNNE($\mathbf{q}$) ;
```

Time complexity The time complexity of creating SiNNE model is $O(t\psi^2)$ and in scoring stage, for query data point, it needs to find whether it falls in any hyperspheres or not, which takes $O(t\psi)$. Total time complexity of SiNNE is $O(t\psi^2 + t\psi)$.

3.3 Subspace Search

Apart from scoring measure, OAM framework requires subspace search method. In this work, we will be using Beam [27] search method, because it is the latest search method and used in literature. We replicate the procedure of beam search in Algorithm 3 for ease of reference. The overall time complexity of beam search is $O(d^2 + W \cdot d \cdot \ell)$, where W is beam width and ℓ maximum dimension of subspace.

```
Algorithm 3: Beam($\mathbf{q}, \ell, \mathcal{X}, W, T, d$)
  Input: $\mathcal{X}$ - given data set, $\mathbf{q}$ - query, $\ell$ - maximum
         dimension, $W$ - beam width, $T$ - number of
         top subspaces, $d$ - number of dimension
  Output: set of outlying features for query $\mathbf{q}$
1 generate $2D$ subspaces ;
2 Add the top $T$ subspaces to $Ans$;
3 for $\ell = 3$ to $\ell$ do
4     initialize $L_{(\ell)} = \Phi$ ;
5     for each subspace $\mathcal{S} \in L_{(\ell-1)}$ do
6         for each Attribute $F_i \in \mathcal{F}$ do
7             if $\mathcal{S} \cup F_i$ not considered yet then
8                 compute outlying-score $\{\mathcal{S} \cup F_i\}$;
9                 if the worst subspace score in $\mathcal{S}$ is
                    worse than $\{\mathcal{S} \cup F_i\}$ then
10                    replace;
11                end
12                if $|L_{(\ell)} < W|$ then
13                    append $\{\mathcal{S} \cup F_i\}$ to $L_{(\ell)}$ ;
14                end
15                else if the worst scored subspace in
                    $L_{(\ell)}$ is worst than $\{\mathcal{S} \cup F_i\}$ then
16                    replace;
17                end
18            end
19        end
20    end
21 end
22 return set of outlying features
```

3.4 An Example of Proposed Method

In this section, we present an illustrative example of proposed method. Figure 4a shows a randomly selected 8 sub-samples (highlighted in black color) from dataset with $n = 50$ in 2-d subspace. Figure 4b shows an example of how $\mathsf{H}(c)$ hypershpere is build at centered c with radii $\tau(c)$. Figure 4c shows all 8 hyperspheres created using 8 sub-samples, which is used to compute outlying degree of the data point. As shown in Fig. 4d, to compute outlying degree of point x, the hypershpere that covers x needs to be determined. The SI (x) = 0 as x falls in hypershpere while data point y does not fall in any hypersphere, and thus outlying degree of y is obtained as 1.

3.5 Key Differences with Closely Related Work

In this subsection, we discuss the difference between SiNNE and iNNE.

Although having similar training process, SiNNE and iNNE employ different scoring mechanism. Specifically, iNNE employs local isolation-based score which is computed as follows:

Table 2 Dataset statistics

Dataset	#datasize (n)	#dimension (d)
synthetic datasets	1000	10–100
wilt	4839	5
pageblocks	5473	10
mnist	20,444	96
u2r	60,821	33
mulcross	262,144	4
covertype	286,144	10

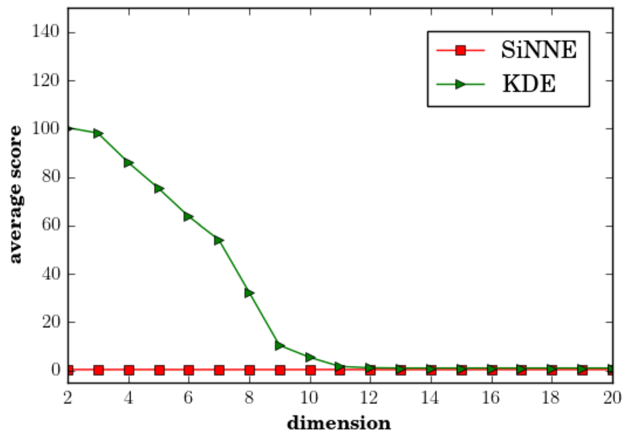


Fig. 5 Dimensionality unbiasedness

$$I_i(\mathbf{q}) = \begin{cases} 1 - \frac{\tau(\eta_{cnn(\mathbf{q})})}{\tau(cnn(\mathbf{q}))}, & \text{if } \mathbf{q} \in \bigcup_{c\in\mathcal{D}_i} \mathcal{H}(c) \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

where $cnn(\mathbf{q}) = \underset{c\in\mathcal{D}}{\arg\min}\{\tau(c) : \mathbf{q} \in \mathcal{H}(c)\}$, $\mathcal{D}$ is set of randomly selected sub-samples without replacement, $|\mathcal{D}| = \psi$, $\mathcal{H}(c)$ is a hypersphere centered at c with radius $\tau(c) = d_{\mathcal{S}}(c, \eta_c)$, where η_c is nearest neighbor of c.

In contrast, SiNNE uses a new simple isolation-based score (cf. Eq. (1)) which assigns 0 if point falls in any hypersphere otherwise 1.

Apart from this, iNNE creates a model in full feature space since it has single sole purpose of detecting outliers from the full feature space $\mathcal{F}$ while the purpose of SiNNE is to detect subspace for the given data point, and thus it creates a model in subspace. Although iNNE [2] was previously used as a outlier detector, its use in OAM context is new.

Table 3 Comparison of SiBeam, RBeam, Beam, and sGBeam in term of exact matches on *synth_10D*. Discovered subspaces with the exact matches with the ground truths are bold-faced. **q**-id represent query point index; **GT** represents ground truth; the numbers in the bracket (subspace) are attribute indices

q-id	GT	SiBeam	RBeam	Beam	sGBeam
172	{8, 9}	**{8, 9}**	{1, 8, 9}	**{8, 9}**	**{8, 9}**
183	{0, 1}	**{0, 1}**	**{0, 1}**	**{0, 1}**	**{0, 1}**
184	{6, 7}	**{6, 7}**	{4, 6, 7}	**{6, 7}**	**{6, 7}**
207	{0, 1}	**{0, 1}**	{0, 1, 7}	**{0, 1}**	**{0, 1}**
220	{2, 3, 4, 5}	**{2, 3, 4, 5}**	{2, 3, 4, 5, 7}	**{2, 3, 4, 5}**	**{2, 3, 4, 5}**
245	{2, 3, 4, 5}	**{2, 3, 4, 5}**	**{2, 3, 4, 5}**	**{2, 3, 4, 5}**	{3, 4, 5}
315	{0, 1}	**{0, 1}**	{0, 1, 9}	**{0, 1}**	**{0, 1}**
	{6, 7}	**{6, 7}**	{0, 6, 7}	**{6, 7}**	**{6, 7}**
323	{8, 9}	**{8, 9}**	{2, 8, 9}	**{8, 9}**	**{8, 9}**
477	{0, 1}	**{0, 1}**	{0, 1, 2}	**{0, 1}**	**{0, 1}**
510	{0, 1}	**{0, 1}**	{0, 1, 5}	**{0, 1}**	**{0, 1}**
577	{2, 3, 4, 5}	**{2, 3, 4, 5}**	{0, 3, 7}	{6, 7}	**{2, 3, 4, 5}**
654	{2, 3, 4, 5}	**{2, 3, 4, 5}**	{1, 2, 3, 4, 5}	**{2, 3, 4, 5}**	**{2, 3, 4, 5}**
704	{8, 9}	**{8, 9}**	{0, 8, 9}	**{8, 9}**	**{8, 9}**
723	{2, 3, 4, 5}	**{2, 3, 4, 5}**	{0, 2, 3, 4, 5}	**{2, 3, 4, 5}**	**{2, 3, 4,5}**
754	{6, 7}	**{6, 7}**	**{6, 7}**	**{6, 7}**	**{6, 7}**
765	{6, 7}	**{6, 7}**	{1, 6, 7}	**{6, 7}**	**{6, 7}**
781	{6, 7}	**{6, 7}**	**{6, 7}**	**{6, 7}**	**{6, 7}**
824	{8, 9}	**{8, 9}**	{6, 8, 9}	**{8, 9}**	**{8, 9}**
975	{8, 9}	**{8, 9}**	**{8, 9}**	**{8, 9}**	**{8, 9}**

Theorem 1 *The isolation score using iNNE with sub-sample size $\psi = 2$ is equivalent to SiNNE.*

Proof Given a iNNE model $\mathcal{H}$ and sample size $\psi = 2$, each set contains two hypersphere with same radius (cf. Definition 5). Thus, $\tau(\eta_{cnn(\mathbf{q})}) = \tau(cnn(\mathbf{q}))$. For sample size ($\psi = 2$) isolation score is as follows:

$$I_i(\mathbf{q}) = \begin{cases} 0, & \text{if } \mathbf{q} \in \bigcup_{c\in\mathcal{D}_i} \mathcal{H}(c), \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

which is same as Eq. 1. □

In terms of performance, SiNNE detects the ground truth for each query while iNNE only detects the ground truth for 11 out of 15 queries (details are presented in "Appendix"). In addition to that, SiNNE is faster than iNNE, this is because SiNNE does not require to find smallest hypersphere and its neighboring hypersphere for score.

Table 4 Comparison of outlying aspects discovered by SiBeam, RBeam, Beam, and sGBeam on four synthetic datasets and average run time of 10 queries from each dataset. Discovered subspaces with the exact matches with the ground truths are bold-faced. **q**-id represent query point index; **GT** represents ground truth; the numbers in the bracket (subspace) are attribute indices

	q-id	**GT**	SiBeam	RBeam	Beam	sGBeam
synth_20D	43	{0, 1, 2}	**{0, 1, 2}**	{0, 14, 17}	**{0, 1, 2}**	**{0, 1, 2}**
	86	{18, 19}	**{18, 19}**	**{18, 19}**	**{18, 19}**	**{18, 19}**
	157	{0, 1, 2}	**{0, 1, 2}**	{1, 4}	**{0, 1, 2}**	**{0, 1, 2}**
	288	{0, 1, 2}	**{0, 1, 2}**	{0, 1, 18}	**{0, 1, 2}**	**{0, 1, 2}**
	451	{18, 19}	**{18, 19}**	{18, 19, 1}	**{18, 19}**	**{18, 19}**
	665	{0, 1, 2}	**{0, 1, 2}**	{2, 4, 17}	**{0, 1, 2}**	**{0, 1, 2}**
	705	{18, 19}	**{18, 19}**	{18, 19, 2}	**{18, 19}**	**{18, 19}**
	873	{18, 19}	**{18, 19}**	{4, 7}	**{18, 19}**	**{18, 19}**
	878	{0, 1, 2}	**{0, 1, 2}**	**{0, 1, 2}**	**{0, 1, 2}**	**{0, 1, 2}**
	942	{18, 19}	**{18, 19}**	{6, 18, 19}	**{18, 19}**	**{18, 19}**
Avg. Run time			0.49	216.68	248.58	**0.42**
synth_50D	106	{41, 42, 43}	**{41, 42, 43}**	**{41, 42, 43}**	**{41, 42, 43}**	**{41, 42, 43}**
	121	{21, 22, 23}	**{21, 22, 23}**	{10, 22, 30}	**{21, 22, 23}**	**{21, 22, 23}**
	200	{13, 14, 15}	**{13, 14, 15}**	{13, 15, 43}	**{13, 14, 15}**	**{13, 14,15}**
	269	{41, 42, 43}	**{41, 42, 43}**	{7, 13}	**{41, 42, 43}**	**{41, 42, 43}**
	427	{5, 6, 7, 8}	**{5, 6, 7, 8}**	{8, 9, 48}	{48, 49}	{44, 46, 47}
	461	{26, 27}	**{26, 27}**	{9, 26, 27}	**{26, 27}**	**{26, 27}**
	512	{24, 25}	**{24, 25}**	{10, 24, 25}	**{24, 25}**	**{24, 25}**
	678	{21, 22, 23}	**{21, 22, 23}**	{9, 23, 32}	**{21, 22, 23}**	**{21, 22, 23}**
	788	{41, 42, 43}	**{41, 42, 43}**	{12, 36, 47}	**{41, 42, 43}**	**{41, 42, 43}**
	885	{48,49}	**{48,49}**	{4, 48, 49}	**{48,49}**	**{48,49}**
Avg. Run time			1.74	1398.75	1492.77	**1.62**
synth_75D	3	{18, 19}	**{18, 19}**	{12, 47, 68}	**{18, 19}**	**{18, 19}**
	33	{11, 12}	**{11, 12}**	{11, 12, 33}	**{11, 12}**	**{11, 12}**
	69	{6, 7, 8}	**{6, 7, 8}**	{16, 53}	**{6, 7, 8}**	**{6, 7, 8}**
	145	{0, 1}	**{0, 1}**	{0, 1, 2}	**{0, 1}**	**{0, 1}**
	214	{9, 10}	**{9, 10}**	{10, 11}	**{9, 10}**	**{9, 10}**
	375	{72, 73, 74}	**{72, 73, 74}**	{19, 25}	**{72, 73, 74}**	**{72, 73, 74}**
	499	{43, 44}	**{43, 44}**	{14, 19}	**{43, 44}**	**{43, 44}**
	526	{40, 41, 42}	**{40, 41, 42}**	{25, 66}	**{40, 41, 42}**	**{40, 41, 42}**
	828	{6, 7, 8}	**{6, 7, 8}**	{5, 8}	**{6, 7, 8}**	**{6, 7, 8}**
	999	{0, 1}	**{0, 1}**	**{0, 1}**	**{0, 1}**	**{0, 1}**
Avg. Run time			3.43	2366.59	2487.15	**3.34**
synth_100D	45	{55, 56, 57}	**{55, 56, 57}**	{82, 98}	**{55, 56, 57}**	**{55, 56, 57}**
	80	{17, 18}	**{17, 18}**	**{17, 18}**	**{17, 18}**	**{17, 18}**
	105	{10, 11}	**{10, 11}**	{0, 10, 11}	**{10, 11}**	**{10, 11}**
	163	{55, 56, 57}	**{55, 56, 57}**	{13, 23, 83}	**{55, 56, 57}**	**{55, 56, 57}**
	258	{43, 44}	**{43, 44}**	{10, 47}	**{43, 44}**	**{43, 44}**
	437	{53, 54}	**{53, 54}**	{10, 21}	**{53, 54}**	**{53, 54}**
	608	{17, 18}	**{17, 18}**	{3, 17}	**{17, 18}**	**{66, 67}**
	771	{53, 54}	**{53, 54}**	{3, 9, 88}	**{53, 54}**	**{53, 54}**
	786	{10, 11}	**{10, 11}**	{7, 32}	**{10, 11}**	**{10, 11}**
	898	{10, 11}	**{10, 11}**	{70, 72}	**{10, 11}**	**{10, 11}**
Avg. Run time			11.51	3663.37	3809.92	**11.33**
Total (40)			40/40	5/40	39/40	39/40

Table 5 Comparison of outlying aspects discovered by SiBeam, RBeam, Beam, and sGBeam on six real-world datasets and average run time of five queries from each dataset. **q**-id represent query point index; the numbers in the bracket (subspace) are attribute indices

Data	**q**-id	SiBeam	RBeam	Beam	sGBeam	Data	**q**-id	SiBeam	RBeam	Beam	sGBeam
wilt	993	{0, 1, 3}	{0}	{0}	{0}	*pageblocks*	336	{2, 3, 8}	{9}	{5}	{0}
	1015	{0, 1, 2}	{0}	{0}	{0}		1488	{0, 1, 3}	{0}	{5}	{0}
	2313	{0, 1, 2}	{0}	{0}	{0}		3706	{3, 9}	{1, 4}	{4}	{3}
	4068	{1, 2, 4}	{4}	{4}	{1}		4582	{0, 3, 7}	{7}	{5}	{0}
	4798	{0, 1, 2}	{2}	{3}	{1}		5121	{0, 1, 6}	{4}	{4}	{6}
Avg. Run time		**0.12**	85.51	85.87	0.31			**0.53**	860.59	867.85	0.69
mnist	2561	{1, 76}	♦	♦	{95}	*u2r*	37,075	{0, 8}	♦	♦	{8}
	8127	{1, 91}			{4}		41,070	{0, 8}			{8}
	9604	{5, 28, 94}			{5}		52,423	{0}			{0}
	11,424	{3, 79}			{30}		56,047	{0, 29}			{8}
	12,032	{9, 15}			{31}		58,769	{0, 16}			{16}
Avg. Run time		**52.74**	>24 h	>24 h	155.86			**9.96**	>24 h	>24 h	270.33
mulcross	8504	{1, 2, 3}	♦	♦	{3}	*covertype*	143,662	{0, 7}	♦	♦	{7}
	17,742	{0, 1, 3}			{0}		143,934	{0, 7}			{7}
	23,545	{0, 1, 2}			{0}		246,578	{0, 2}			{7}
	133,002	{0}			{2}		248,303	{0, 7}			{7}
	228,099	{1, 2, 3}			{0}		248,622	{0, 7}			{7}
Avg. Run time		**0.15**	>24 h	>24 h	12.67			**2.47**	>24 h	>24 h	263.76

Table 6 Comparison of SiBeam, RBeam, Beam, and sGBeam on six real-world datasets in terms of quality of discovered subspace

Data	**q**-id	SiBeam	RBeam	Beam	sGBeam	Data	**q**-id	SiBeam	RBeam	Beam	sGBeam
wilt	993	**−0.48**	−0.39	−0.39	−0.39	*pageblocks*	336	**−1.66**	−0.65	−0.19	0.30
	1015	**−0.51**	−0.39	−0.39	−0.39		1488	**−1.24**	−0.93	0.26	−0.93
	2313	**−0.49**	−0.39	−0.39	−0.39		3706	−0.94	**−1.10**	0.22	−0.87
	4068	**−0.36**	0.40	0.40	0.71		4582	**−1.38**	−0.94	0.85	0.60
	4798	**−2.43**	−0.77	0.18	−0.65		5121	**−2.37**	0.50	0.50	−1.00
mnist	2561	**−0.74**	♦	♦	0.18	*u2r*	37,075	**−1.00**	♦	♦	**−1.00**
	8127	**−0.26**			0.69		41,070	**−1.00**			**−1.00**
	9604	**−0.70**			0.36		52,423	**−0.57**			**−0.57**
	11,424	**−0.56**			0.56		56,047	**−1.00**			**−1.00**
	12,032	**−0.28**			0.36		58,769	**−1.00**			**−1.00**
mulcross	8504	**−0.22**	♦	♦	0.39	*covertype*	143,662	**0.17**	♦	♦	0.42
	17,742	**−0.22**			0.70		143,934	**0.01**			0.16
	23,545	**0.11**			0.29		246,578	**−0.13**			0.01
	133,002	1.02			**0.68**		248,303	**−0.83**			−0.77
	228,099	**0.03**			0.82		248,622	**−0.83**			−0.77

4 Experimental Setting

4.1 Datasets

In this study, we used two types of datasets, i.e., synthetic and real-world. For synthetic[2] datasets, we adopted five datasets ([13]): *synth_10D*, *synth_20D*, *synth_50D*, *synth_75D*, and *synth_100D*.

For real-world[3] datasets, we adopted six datasets ([5]): *wilt*, *pageblocks*, *mnist*, *u2r*, *mulcross* and *covertype*.

The characteristics of datasets in terms of data size and the dimensionality of the original input space are provided in Table 2.

4.2 Contenders and Parameters

We compare SiNNE (**SiBeam**) with three contenders (a) kernel density rank (**RBeam**), (b) *Z*-Score normalized kernel density (**Beam**) and (c) *Z*-Score normalized sGrid density (**sGBeam**).

We used default parameters as suggested in respective papers unless specified otherwise. For SiBeam, we set $\psi = 8$ and $t = 100$. The Beam and RBeam employed KDE (kernel density estimator) to estimate density. KDE uses the Gaussian kernel with default bandwidth.[4] To calculate the Gaussian kernel, we use Euclidean distance. The parameter w block size for bit set operation in sGBeam was set to 64 as suggested by the authors [28]. Parameters beam width (W) and maximum dimensionality of subspace (ℓ) in Beam search procedure were set to 100 and 3, respectively, as done in [27].

4.3 Evaluation Metric

As far as we know, there is no such publicly available real-world dataset which offers ground truth to verify the quality of discovered subspaces. Therefore, in the absence of a better evaluation measure, we propose to use a mean kernel embedding [19] to evaluate the quality of discovered subspaces. The intuition behind the mean kernel embedding is, in the most outlying aspect, the query is far away from the distribution of the data, i.e., it has the minimum average similarity with rest of the data. The quality of discovered subspace $\mathcal{S}$ for a query $\mathbf{q}$ using a kernel mean embedding method [19] is computed as follows:

$$f_{\mathcal{S}}(\mathbf{q}, \mathcal{X}) = \frac{1}{n} \sum_{x \in \mathcal{X}} K_{\mathcal{S}}(\mathbf{q}, x) \tag{5}$$

where $K_{\mathcal{S}}(\mathbf{q}, x)$ is a kernel similarity of $\mathbf{q}$ and x in subspace $\mathcal{S}$.

We use Chi-square kernel [32] because it is parameter-free and widely used by the computer vision research community. The Chi-square kernel $K_{\mathcal{S}}(\mathbf{q}, x)$ is computed as follows:

$$K_{\mathcal{S}}(\mathbf{q}, x) = 1 - \sum_{i \in \mathcal{S}} 2 \frac{(\mathbf{q}_i - x_i)^2}{(\mathbf{q}_i + x_i)}$$

In OAM, $\mathbf{q}$ is considered to be more outlier in $\mathcal{S}_i$ than $\mathcal{S}_j$ if $f_{\mathcal{S}_i}(\mathbf{q}, \mathcal{X}) < f_{\mathcal{S}_j}(\mathbf{q}, \mathcal{X})$.

4.4 Implementation

All measures and experimental setup were implemented in Java using WEKA platform [10]. We made the required changes in the Java implementation of iNNE[5] provided by the authors to implement SiNNE. We used the Java implementations of sGrid made available by the authors [28].

All experiments were conducted on a machine with Intel 8-core i9 CPU and 16 GB main memory, running on macOS Monterey version 12.0.1.

We run each jobs on multiple single CPU treads, which is done using GNU parallel [25]. All jobs were performed upto 24 h, and incomplete jobs were killed and marked as '♦'.

5 Empirical Evaluation

In this section, we compare SiNNE and three contenders in four set of experiments: (a) Experiment 1—dimensionality unbiasedness; (b) Experiment 2—performance on synthetic datasets; (c) Experiment 3—performance on real-world datasets; and (d) Experiment 4—run-time comparisons.

5.1 Experiment 1: Dimensionality Unbiasedness

We generated 19 synthetic datasets using NumPy [12] library. Each dataset contains 1000 data points from uniform distribution $\mathcal{U}([0,1]^d)$, where d varied from 2 to 20. We computed the average score of all instances using SiNNE and KDE. The results are presented in Fig. 5. The flat line for SiNNE shows that it is dimensionality unbiased, whereas KDE (without Z-Score normalization) is not. Note that [27] shows that ranks and *Z*-Score normalization make any score dimensionally unbias. Hence, we did not include them in our experiment.

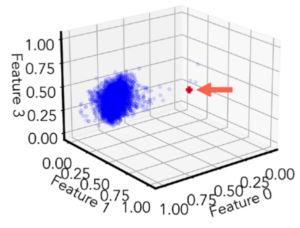
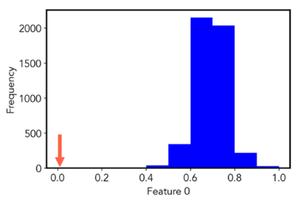
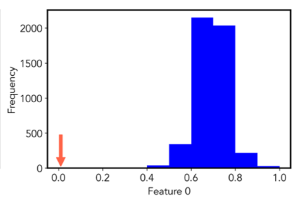
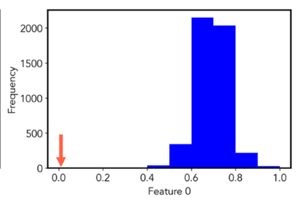
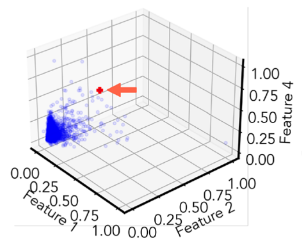
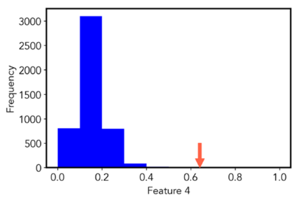
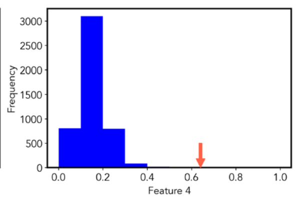
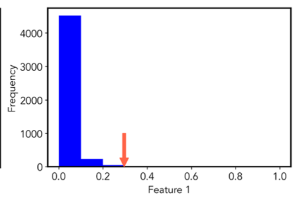
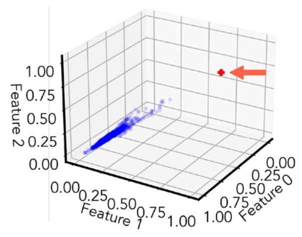
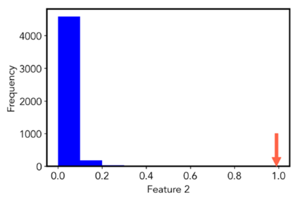
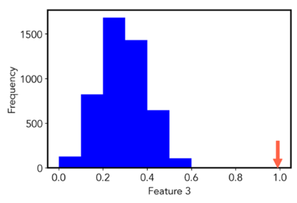
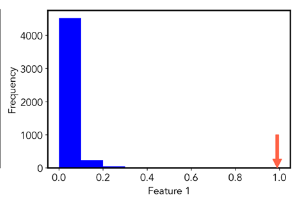
[2] Available at https://www.ipd.kit.edu/ muellere/HiCS/.

[3] Available at https://elki-project.github.io/datasets/outlier.

[4] Note that a better rule of thumb [11] was used to set bandwidth h as: $h = 1.06\, min\left\{\sigma, \frac{R}{1.34}\right\} n^{-\frac{1}{5}}$ where $R = \mathcal{X}_{[0.75n]} - \mathcal{X}_{[0.25n]}$, where $\mathcal{X}_{[0.25n]}$ and $\mathcal{X}_{[0.75n]}$ are the first and third quartiles of data $\mathcal{X}$, respectively.

[5] Available at https://github.com/tharindurb/iNNE.

Table 7 Visualization of discovered subspaces by SiBeam, RBeam, Beam, and sGBeam in the *wilt* dataset

q-id	SiBeam	RBeam	Beam	sGBeam
993				
4068				
4798				

5.2 Experiment 2: Performance on Synthetic Datasets

[13] provided several synthetic datasets, which are used in previous studies [8, 22, 27, 28]. The collection of these synthetic datasets have 1000 data points and dimensions are 10, 20, 50, 75, and 100. Each dataset has a fixed number of outliers for which outlying subspaces are known (ground truth).

synth_10D has 19 outliers, we passed all outliers one at a time as a query. Table 3 summarize the subspace discovered by SiBeam, RBeam, Beam, and sGBeam for all 19 queries. In terms of exact matches, SiBeam is the best performing measure which detects the ground truth as a top outlying aspect of each query. Beam and sGBeam perform similar by producing 19 exact matches. RBeam is the worst performing measure, which produces only five exact matches.

Table 4 summarizes the mining results of SiBeam, RBeam, Beam, and sGBeam on four synthetic datasets, i.e., *synth_20D*, *synth_50D*, *synth_75D* and *synth_100D*. SiBeam finds the ground truth as a top outlying subspace for each query (ten queries from each datasets). Beam and sGBeam perform similar by producing 39 exact matches out of 40. RBeam is the worst performing measure, which produces exact matches for 5 queries out of 40.

5.3 Experiment 3: Performance on Real-World Datasets

In real-world datasets, outliers and their outlying aspects are not available. Thus, we used the state-of-the-art outlier detector called iForest[6] [15] to find top k ($k = 5$) outliers and they were used as queries. We then use the f_S score (cf. Eq. 5) in the top-ranked subspace to measure the quality of discovered subspace—the lower the value, the more likely the subspace is outlying aspect of a given query.

It is worth noting that SiBeam and sGBeam are the only methods which are able to finish the process for each query, while RBeam and Beam finish the process for only 10 queries.

Table 5 shows subspaces discovered by four OAM methods (i.e., SiBeam, RBeam, Beam, and sGBeam) on six real-world datasets.

Table 6 shows the quality of discovered subspaces by SiBeam, RBeam, Beam, and sGBeam. High-quality subspaces of each query is highlighted in bold. SiBeam is best

[6] We used default parameter of ψ and t which are 256 and 100, respectively.

Table 8 Visualization of discovered subspaces by SiBeam, RBeam, Beam and sGBeam in the *pageblock* dataset

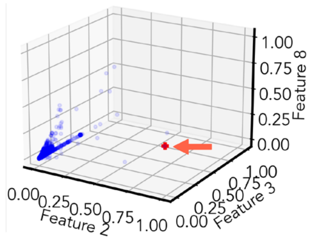
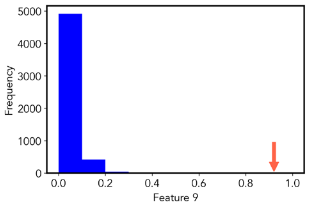
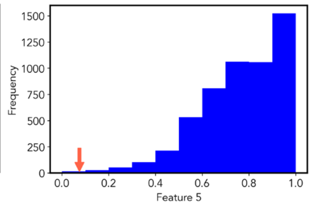
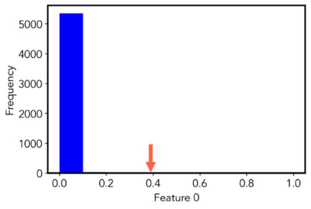
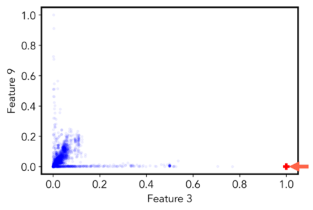
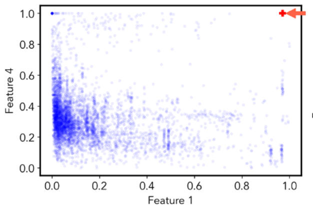
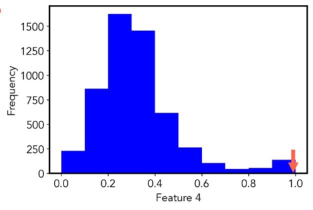
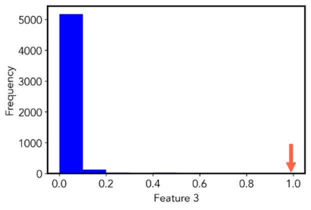
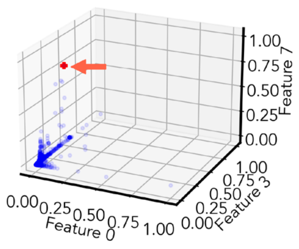
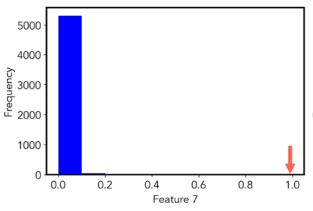
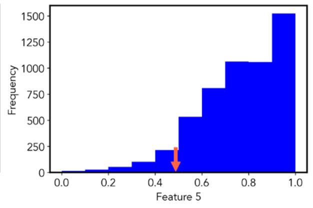
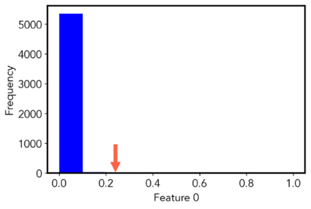
q-id	SiBeam	RBeam	Beam	sGBeam
336				
3706				
4582				

Table 9 Visualization of discovered subspaces by SiBeam, RBeam, Beam and sGBeam in the *mnist* dataset

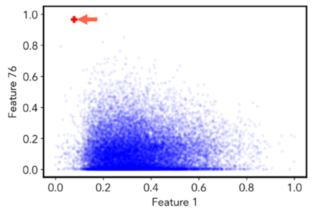
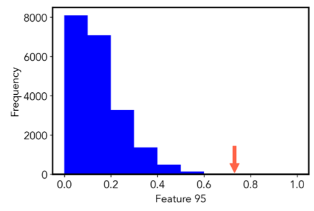
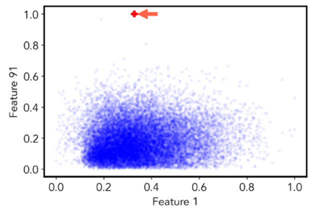
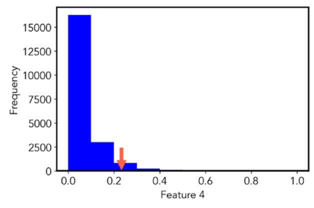
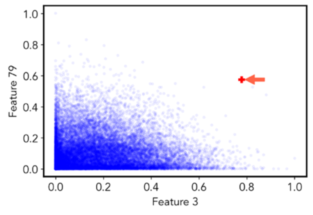
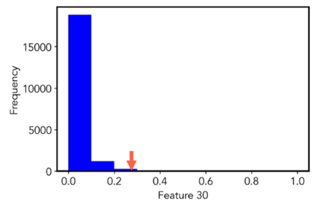
q-id	SiBeam	RBeam	Beam	sGBeam
2561		♦	♦	
8127		♦	♦	
11424		♦	♦	

Table 10 Visualization of discovered subspaces by SiBeam, RBeam, Beam and sGBeam in the *u2r* dataset

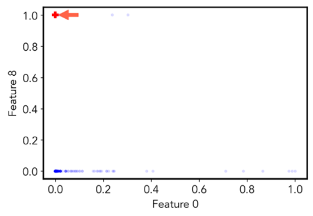
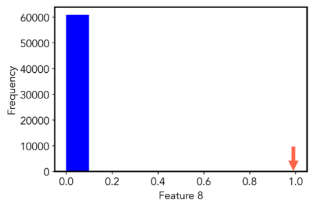
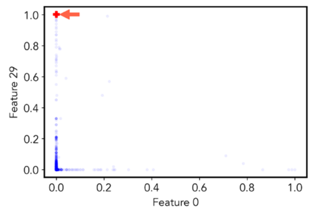
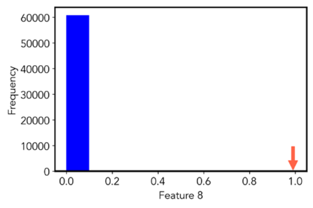
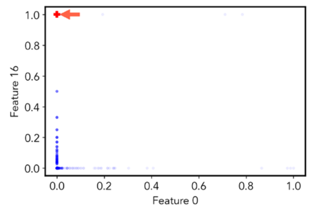
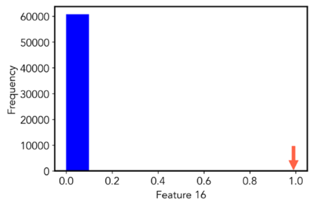
q-id	SiBeam	RBeam	Beam	sGBeam
37075		♦	♦	
56047		♦	♦	
58769		♦	♦	

Table 11 Visualization of discovered subspaces by SiBeam, RBeam, Beam and sGBeam in the *mulcross* dataset

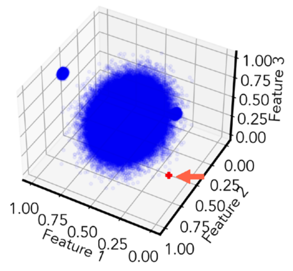
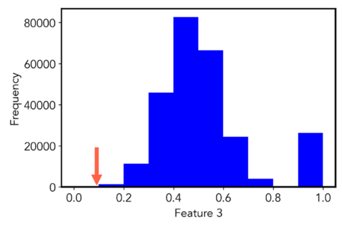
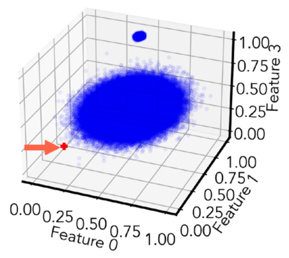
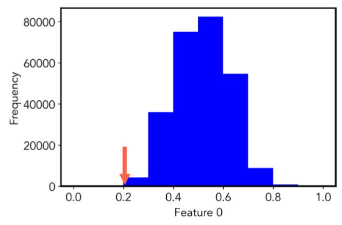
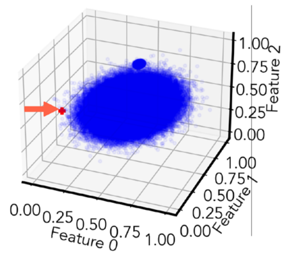
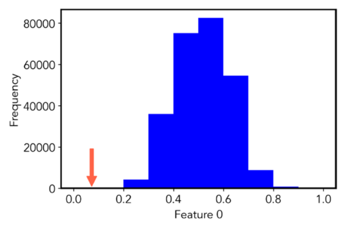
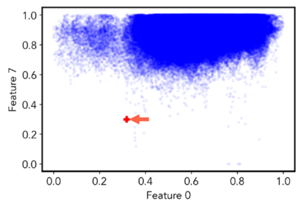
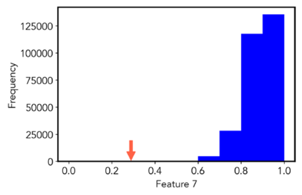
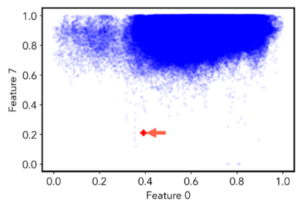
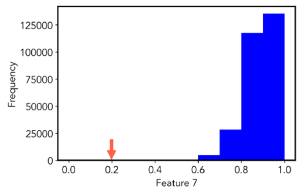
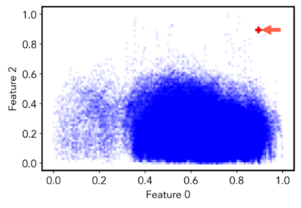
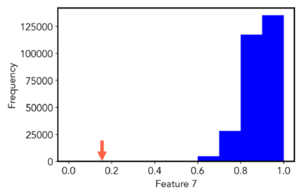
q-id	SiBeam	RBeam	Beam	sGBeam
8504		♦	♦	
17742		♦	♦	
23545		♦	♦	

Table 12 Visualization of discovered subspaces by SiBeam, RBeam, Beam and sGBeam in the *covertype* dataset

q-id	SiBeam	RBeam	Beam	sGBeam
143662		♦	♦	
143934		♦	♦	
246578		♦	♦	

performer on 28 out of 30 according to proposed quality measure. sGBeam discovered high-quality subspace for only 5 queries out of 30. On the other hand, RBeam discovered high-quality subspace for only one query out of ten, whereas Beam was unable to detect high-quality subspace even for a single query.

The average run time of five queries for each dataset is presented in Table 5. Next, we visually compare discovered subspaces by each measure for top query from each datasets.

Tables 7, 8, 9, 10, 11 and 12 shows the subspace discovered by SiBeam and contending measures on *wilt*, *pageblock*, *mnist*, *u2r*, *mulcross*, and *covertype*, respectively. Visually, we can say that SiBeam detects better subspace than its 3 contenders.

5.4 Experiment 4: Run-Time Comparison

Table 7 shows average run time for randomly chosen 10 queries from each real-world datasets of the SiBeam and its three contending measures. SiBeam and sGBeam were able to finish for all datasets, whereas RBeam and beam only able to finish on *wilt*, and *pageblock* datasets within 24 h. These results shows that the proposed scoring measure enables the existing OAM approach based on beam search to run orders of magnitude faster in large datasets. Specifically, SiBeam runs at least two and three magnitude faster than RBeam and Beam on *wilt* and *pageblocks* datasets, respectively. SiBeam runs at least two order of magnitude faster than sGBeam on large datasets ($n > 50$K).

6 Conclusion

In this paper, we have introduced an efficient and effective scoring measure Simple Isolation score using Nearest Neighbor **E**nsemble (SiNNE), which is dimensionally unbias. By replacing the existing scoring measure to proposed scoring measure, we gain three benefits. The first benefit is that

Table 13 Average run time (in CPU seconds) for 10 queries of SiBeam, RBeam, Beam, and sGBeam on six real-world datasets

Dataset	SiBeam	RBeam	Beam	sGBeam
wilt	0.18 ± 0.09	110.69 ± 21.74	111.39 ± 21.96	0.42 ± 0.11
pageblocks	0.66 ± 0.27	1226.62 ± 298.52	1234.37 ± 295.76	0.89 ± 0.38
mnist	73.09 ± 22.27	>24 h	>24 h	189.69 ± 28.6
u2r	7.91 ± 1.95	>24 h	>24 h	175.78 ± 18.25
mulcross	0.41 ± 0.12	>24 h	>24 h	18.5 ± 1.08
covertype	3.56 ± 1.55	>24 h	>24 h	317.51 ± 13.03

Table 14 Comparison of SiNNE (SiBeam) and iNNE (iBeam) on five synthetic datasets. Discovered subspaces with the exact matches with ground truths are bold-faced. **q**-id represent query point index; the numbers in the bracket (subspace) are attribute indices

	q-id	GT	SiBeam	iBeam
synth_10D	172	{8, 9}	**{8, 9}**	**{8, 9}**
	207	{0, 1}	**{0, 1}**	**{0, 1}**
	723	{2, 3, 4, 5}	**{2, 3, 4, 5}**	{6}
synth_20D	43	{0, 1, 2}	**{0, 1, 2}**	**{0, 1, 2}**
	86	{18, 19}	**{18, 19}**	**{18, 19}**
	288	{0, 1, 2}	**{0, 1, 2}**	**{0, 1, 2}**
synth_50D	106	{41, 42, 43}	**{41, 42, 43}**	{24}
	121	{21, 22, 23}	**{21, 22, 23}**	**{21, 22, 23}**
	885	{48,49}	**{48,49}**	**{48,49}**
synth_75D	214	{9, 10}	**{9, 10}**	**{9, 10}**
	375	{72, 73, 74}	**{72, 73, 74}**	{38, 39}
	828	{45, 46, 47}	**{45, 46, 47}**	{6, 7, 8}
synth_100D	258	{43, 44}	**{43, 44}**	**{43, 44}**
	437	{53, 54}	**{53, 54}**	**{53, 54}**
	771	{53, 54}	**{53, 54}**	**{53, 54}**

SiNNE is dimensionally unbiased measure, which does not rely on any normalization means it can be used directly to compare subspaces with different dimensionality. The second benefit is that SiNNE allows existing OAM (i.e., Beam) to run orders of magnitude faster compared to three state-of-the-art scoring measures. Thus it is more suitable for mining huge datasets with thousands of dimensions. The third benefit is now we can identify more interesting outlying subspace for a given query. This is confirmed by considerably better performance of SiNNE, compared to three state-of-the-art scoring measures in empirical evaluation. In addition to that, we introduced a new performance measure for outlying aspect mining. Our experimental results on real-world datasets show that SiNNE perform comparatively better than state-of-the-art measures.

Appendix: SiNNE Versus iNNE

This appendix provides the additional results of SiNNE and iNNE comparison from the following Sect. 3.5.

Table 14 presents the subspaces discovered by SiNNE and iNNE on synthetic datasets. In term of exact matches, SiNNE detects ground truth of each query as outlying aspects, whereas iNNE only detects ground truth of 11 queries out of 15 as most outlying aspects.

Acknowledgements The preliminary version of this paper is published in Proceedings of the 21st International Conference on Web Information Systems Engineering (WISE) 2020 [22]. This work is supported by Federation University Research Priority Area (RPA) scholarship, awarded to Durgesh Samariya.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Angiulli F, Fassetti F, Manco G, Palopoli L (2017) Outlying property detection with numerical attributes. Data Min Knowl Disc 31(1):134–163
2. Bandaragoda TR, Ting KM, Albrecht D, Liu FT, Wells JR (2014) Efficient anomaly detection by isolation using nearest neighbour ensemble. In: 2014 IEEE international conference on data mining workshop, pp 698–705
3. Bandaragoda TR, Ting KM, Albrecht D, Liu FT, Zhu Y, Wells JR (2018) Isolation-based anomaly detection using nearest-neighbor ensembles. Comput Intell 34(4):968–998. https://doi.org/10.1111/coin.12156
4. Brockett PL, Xia X, Derrig RA (1998) Using Kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. J Risk Insur 65(2):245–274. http://www.jstor.org/stable/253535
5. Campos GO, Zimek A, Sander J, Campello RJGB, Micenková B, Schubert E, Assent I, Houle ME (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min Knowl Disc 30(4):891–927. https://doi.org/10.1007/s10618-015-0444-8
6. Chan PK, Fan W, Prodromidis AL, Stolfo SJ (1999) Distributed data mining in credit card fraud detection. IEEE Intell Syst Appl 14(6):67–74
7. Dang XH, Micenková B, Assent I, Ng RT (2013) Local outlier detection with interpretation. In: Blockeel H, Kersting K, Nijssen S, Železný F (eds) Machine learning and knowledge discovery in databases. Springer Berlin Heidelberg, Berlin, pp 304–320
8. Duan L, Tang G, Pei J, Bailey J, Campbell A, Tang C (2015) Mining outlying aspects on numeric data. Data Min Knowl Disc 29(5):1116–1151. https://doi.org/10.1007/s10618-014-0398-2
9. Gupta N, Eswaran D, Shah N, Akoglu L, Faloutsos C (2019) Beyond outlier detection: lookout for pictorial explanation. In: Berlingerio M, Bonchi F, Gärtner T, Hurley N, Ifrim G (eds) Machine learning and knowledge discovery in databases. Springer, Cham, pp 122–138
10. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explor Newsl 11(1):10–18. https://doi.org/10.1145/1656274.1656278
11. Härdle W (2012) Smoothing techniques: with implementation in S. Springer, New York
12. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R,

Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. Nature 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2

13. Keller F, Muller E, Bohm K (2012) Hics: high contrast subspaces for density-based outlier ranking. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, ICDE'12, pp 1037–1048, https://doi.org/10.1109/ICDE.2012.88
14. Lin J, Keogh E, Ada Fu, Van Herle H (2005) Approximations to magic: finding unusual medical time series. In: 18th IEEE symposium on computer-based medical systems (CBMS'05), pp 329–334
15. Liu FT, Ting KM, Zhou Z (2008) Isolation forest. In: 2008 Eighth IEEE international conference on data mining, pp 413–422
16. Liu N, Shin D, Hu X (2018) Contextual outlier interpretation. In: Proceedings of the 27th international joint conference on artificial intelligence. AAAI Press, IJCAI'18, pp 2461–2467
17. Mejía-Lavalle M, Sánchez Vivar A (2009) Outlier detection with explanation facility. In: Perner P (ed) Machine learning and data mining in pattern recognition. Springer Berlin Heidelberg, Berlin, pp 454–464
18. Micenková B, Ng RT, Dang X, Assent I (2013) Explaining outliers by subspace separability. In: 2013 IEEE 13th international conference on data mining, pp 518–527, https://doi.org/10.1109/ICDM.2013.132
19. Muandet K, Fukumizu K, Sriperumbudur B, Schölkopf B (2017) Kernel mean embedding of distributions: a review and beyond. Found Trends Mach Learn 10(1–2):1–141
20. Samariya D, Ma J (2021) Mining outlying aspects on healthcare data. In: Siuly S, Wang H, Chen L, Guo Y, Xing C (eds) Health information science. Springer, Cham, pp 160–170
21. Samariya D, Thakkar A (2021) A comprehensive survey of anomaly detection algorithms. Ann Data Sci. https://doi.org/10.1007/s40745-021-00362-9
22. Samariya D, Aryal S, Ting KM, Ma J (2020) A new effective and efficient measure for outlying aspect mining. In: Huang Z, Beek W, Wang H, Zhou R, Zhang Y (eds) Web information systems engineering—WISE 2020. Springer, Cham, pp 463–474
23. Samariya D, Ma J, Aryal S (2020b) A comprehensive survey on outlying aspect mining methods. arXiv preprint arXiv:2005.02637
24. Silverman BW (1986) Density estimation for statistics and data analysis. Chapman & Hall, London
25. Tange O (2020) Gnu parallel 20201022 ('samuelpaty'). Zenodo. https://doi.org/10.5281/zenodo.4118697
26. Vinh NX, Chan J, Bailey J, Leckie C, Ramamohanarao K, Pei J (2015) Scalable outlying-inlying aspects discovery via feature ranking. In: Cao T, Lim EP, Zhou ZH, Ho TB, Cheung D, Motoda H (eds) Advances in knowledge discovery and data mining. Springer, Cham, pp 422–434
27. Vinh NX, Chan J, Romano S, Bailey J, Leckie C, Ramamohanarao K, Pei J (2016) Discovering outlying aspects in large datasets. Data Min Knowl Disc 30(6):1520–1555. https://doi.org/10.1007/s10618-016-0453-2
28. Wells JR, Ting KM (2019) A new simple and efficient density estimator that enables fast systematic search. Pattern Recognit Lett 122:92–98. https://doi.org/10.1016/j.patrec.2018.12.020
29. Xu H, Wang Y, Jian S, Huang Z, Wang Y, Liu N, Li F (2021) Beyond outlier detection: Outlier interpretation by attention-guided triplet deviation network. In: Proceedings of the web conference 2021, association for computing machinery, New York, NY, USA, WWW'21, pp 1328–1339, https://doi.org/10.1145/3442381.3449868
30. Xu YX, Pang M, Feng J, Ting KM, Jiang Y, Zhou ZH (2021) Reconstruction-based anomaly detection with completely random forest. In: Proceedings of the 2021 SIAM international conference on data mining (SDM), SIAM, pp 127–135
31. Zhang J, Lou M, Ling TW, Wang H (2004) Hos-miner: a system for detecting outlying subspaces of high-dimensional data. In: Proceedings of the thirtieth international conference on very large data bases—volume 30, VLDB endowment, Toronto, Canada, VLDB'04, pp 1265–1268, http://dl.acm.org/citation.cfm?id=1316689.1316810
32. Zhang J, Marszałek M, Lazebnik S, Schmid C (2007) Local features and kernels for classification of texture and object categories: a comprehensive study. Int J Comput Vis 73(2):213–238. https://doi.org/10.1007/s11263-006-9794-4