REGULAR PAPER

Tighten the lasso: a convex hull volume-based anomaly detection method

Uri Itai[1] · Asael Bar Ilan[2] · Teddy Lazebnik[3,4]

Abstract

Detecting out-of-distribution (OOD) data is a critical task for maintaining model reliability and robustness. In this study, we propose a novel anomaly detection algorithm that leverages the convex hull (CH) property of a dataset by exploiting the observation that OOD samples marginally increase the CH's volume compared to in-distribution samples. Thus, we establish a decision boundary between OOD and in-distribution data by iteratively computing the CH's volume as samples are removed, stopping when such removal does not significantly alter the CH's volume. The proposed algorithm is evaluated against seven widely used anomaly detection methods across ten datasets, demonstrating performance comparable to state-of-the-art (SOTA) techniques. Furthermore, we introduce a computationally efficient criterion for identifying datasets where the proposed method outperforms existing SOTA approaches.

Keywords Outlier detection · Convex hull · Out-of-distribution · Geometry of a set · Hypervolume

1 Introduction

Anomaly detection (AD) plays a pivotal role across machine learning, data analysis, statistics, and other domains involving quantitative data [1]. It focuses on identifying patterns, observations, or events that markedly deviate from expected norms, often termed *anomalies* or *outliers* [2–4]. Such anomalies frequently correspond to rare yet important occurrences [5, 6] and have diverse applications, including the early detection of mechanical faults [2], financial fraud prevention [7], cybersecurity [8], enhanced decision-making [9], as well as powering real-time clinical alert systems [10].

In a data-driven context, AD is commonly regarded as a subfield of machine learning (ML) and can be mathematically framed as a binary classification problem, where data is categorized as either normal or anomalous [11–13]. However, due to the inherent rarity of anomalies, conventional "normal" classification models often underperform in this task, necessitating the development of specialized algorithms tailored for anomaly detection [14].

Indeed, various groups of anomaly detection (AD) algorithms have been developed, each tailored to different types of data and problem contexts [15–18]. These algorithms can be broadly categorized into three primary types: statistical-based methods, machine learning (ML)-based methods, and distance-based methods. These models focus on the hyperplane which is a flat $d-1$-dimensional surface in a d-dimensional space, separating normal from abnormal data by positioning a single linear boundary. Its advantages are low computational complexity, simplicity, and good performance for linearly separable data, but it cannot directly capture complex or curved data shapes without kernel transformations. In contrast, a hypervolume—often represented by the convex hull—forms the smallest convex region enclosing all normal data points, allowing it to model arbitrary convex shapes in the feature space. This provides higher flexibility and the ability to tightly wrap around the data distribution, which can improve detection accuracy for nonlinear patterns, while being more computationally expensive.

Uri Itai and Teddy Lazebnik have contributed equally to this work.

✉ Uri Itai
  uri.itai@gmail.com

1 Department of Mathematics, The Guangdong Technion-Israel Institute of Technology, Shantou, China

2 Department of Computer Science, Open University of Israel, Raanana, Israel

3 Department of Information Systems, University of Haifa, Haifa, Israel

4 Department of Computing, Jonkoping University, Jonkoping, Sweden

The traditional method of determining anomalies by taking k standard deviations away from the mean is extended using the Z-score (T statistics), in which the confidence interval is represented as an ellipsoid. This advancement paves the way for more refined, statistically-based techniques, including Gaussian mixture models [19], Mahalanobis distance [20], and hypothesis testing [21]. This method concerning the convex hull was done in [22]. These methods rely on the presumption that the data adheres to a specific statistical distribution, thereby identifying any points that significantly diverge from this distribution as anomalies.

ML-based methods can be classified into supervised, semi-supervised, and unsupervised approaches [23]. Supervised methods, such as neural networks or support vector machines, require labeled datasets containing normal and abnormal instances. However, obtaining these labels is often impractical in real-world scenarios due to the high cost and difficulty associated with labeling data [24]. In contrast, semi-supervised methods, such as one-class SVM [25], one-class deep neural networks [26], and autoencoders [27], as well as Local Outlier Factor [28], are trained exclusively on normal data and detect anomalies by identifying instances that do not conform to the learned patterns [29]. Additional methods include contracting a gain function [30] or constructing a distribution [22].

Another approach involves the use of Voronoi diagrams [31]. In this method, the data is partitioned into bins, and points that do not fit within these bins are flagged as anomalies. The computation of each cell in the diagram is based on the convex hull.

Distance-based methods, such as K-nearest neighbors (KNN) [32] and clustering techniques like DBSCAN [33], involve measuring the distance or similarity between data points. Data points far from clusters or with few neighboring points are considered anomalies. These methods are typically limited in scalability, as they struggle to handle large datasets due to their computational complexity [34]. Moreover, distance-based methods tend to perform poorly in non-uniform data distributions, as they assume anomalies are far from most data points.

An alternative approach to AD utilizes the concept of a convex hull, defined as the smallest convex set that contains a given set of points in a metric space [35]. Simply put, a convex hull is a set of outermost nails on a board stretching a rubber band around a set of nails when released. Computing the convex hull plays an important role in many fields of computer science, such as computer graphics [36] and computer-aided design [37]. This geometric framework leads to convex-hull-based algorithms for AD, which offer an efficient way to identify anomalies by leveraging geometric properties [38]. A convex hull, representing the boundary of the normal data distribution, delineates the smallest convex shape enclosing a set of points in multi-dimensional space [38]. Any data point outside this convex boundary is considered anomalous [22]. These algorithms are particularly powerful because they provide a clear, interpretable geometric definition of normal behavior, making it easy to visualize and understand the separation between normal and anomalous data. Moreover, convex-hull-based methods are adaptable to high-dimensional and complex datasets as they require only a well-defined distance metric, which is relatively straightforward in most real-world applications [39, 40]. Several studies already used convex-hull-based AD algorithms, showing promising results [41–43]. For instance, Liu at al. [44] proposed adding a discount factor to the convex hull to avoid overfitting on the training dataset. Casale et al. [45] reduce the computation time of the convex hull as an AD algorithm using the approximate polytope ensemble technique. In an applied context, He et al. [46] propose a novel classifier, the kernel flexible and displaceable convex-hull-based tensor machine, designed for gearbox fault diagnosis using multi-source signals. This classifier uses a convex hull approach in tensor feature space to classify feature tensors, demonstrating improved robustness and effectiveness in identifying gearbox faults with small sample sizes.

Although efficient, existing algorithms lack a definitive mechanism for determining the optimal convex hull shape for a given dataset without introducing additional assumptions. Furthermore, their practical applications remain ambiguous within the broader context of state-of-the-art anomaly detection (AD) algorithms.

In this study, we propose a novel parameter-free AD algorithm grounded in the convex hull family of methods. This algorithm balances two configurations: the convex hull encompassing the entire dataset and the convex hull computed for a subset of the data, excluding the most "distant" samples. By leveraging this approach, we introduce a topology-agnostic AD algorithm that maintains computational efficiency and resource requirements comparable to other state-of-the-art AD algorithms while exhibiting robustness to complex data geometries.

The proposed method not only identifies anomalies but also quantifies the extent to which a given instance deviates from the norm. This metric is instrumental in distinguishing between tail events and out-of-distribution instances, a distinction of particular importance in time series analysis [47] and online data processing [48]. While a detailed examination of this distinction lies beyond the scope of the present paper, it will be addressed in future research.

We conducted extensive experiments using ten real-world datasets spanning various domains to evaluate the proposed method. We benchmarked the performance of our model against seven state-of-the-art AD algorithms representing all three principal categories of AD methods. The results demonstrate. The convex hull algorithm demonstrates strong performance, surpassing all competing methods except for

Isolation Forest and Local Outlier Factor. However, Isolation Forest does not account for distances and angles and remains invariant under shearing transformations, which may not always be desirable. In contrast, the Local Outlier Factor is a local algorithm, meaning it primarily identifies local anomalies rather than global ones. As a result, it may fail to detect global anomalies. Consequently, the convex-hull-based anomaly detection algorithm proves to be effective in scenarios that require a global geometric perspective.

The rest of the paper is organized as follows. Section 2 reviews the state-of-the-art AD algorithms as well as the convex hull family of algorithms and provides examples of geometry-based AD algorithms for one and two dimensions. Section 3 formally introduces the proposed convex-hull-based AD algorithm with an analytical analysis of the algorithm. Section 4 outlines the experiments on real-world data comparing the proposed algorithms with current state-of-the-art AD algorithms and also the sensitivity analysis of the proposed algorithm. Finally, Section 5 discusses the applicative outcomes of this study and suggests possible future research.

2 Related work

This section presents a comprehensive review of the definition of the convex hull, followed by an exploration of the algorithms commonly utilized for its numerical computation. Subsequently, an in-depth examination of the current state-of-the-art anomaly detection (AD) algorithms is provided, focusing on their methodologies, applications, and performance characteristics.

2.1 Convex hull

The convex hull holds significant importance across various research domains, including computer graphics [49], computational geometry [50], optimization [51], and numerous other fields.

The convex hull is the smallest convex set that completely encloses a given set of points, minimizing the hypervolume enclosed by the set. Formally, for a set of points $\{X_i\}_{i=1}^n \subset R^d$, the convex hull is the minimal convex polytope containing all points, and it is expressed as [52]:

$$CH(S) = \left(\sum_{i=1}^n \alpha_i X_i \,|\, \alpha_i \geq 0 \wedge \sum_{i=1}^n \alpha_i = 1 \right), \tag{1}$$

where α_i represents the non-negative scalar coefficients associated with each point x_i in the set S.

2.1.1 Computing convex hull

Various methods have been proposed to compute the convex hull of an arbitrary set of points, leveraging the convex hull's inherent properties of convexity. Among these, three widely utilized algorithms are noteworthy: the gift wrap algorithm [53], the Graham scan [54], and the Quickhull algorithm [36].

The gift wrap algorithm [53], also known as Jarvis March, constructs the convex hull incrementally by identifying boundary points one at a time. Starting with the leftmost point, which is guaranteed to be part of the convex hull, the algorithm iteratively selects the point that forms the smallest counterclockwise angle with the line segment formed by the current hull boundary. This selection ensures that the boundary progresses in a "wrapping" motion, ultimately enclosing all points. The hull is complete once the algorithm loops back to the starting point. The simplicity of the gift wrap algorithm makes it a natural choice for small datasets or cases where the number of points on the hull is small. However, its time complexity, O(nh), where n is the total number of points and h is the number of points on the hull, can become prohibitive for larger datasets with dense point distributions.

Graham's scan algorithm [54] begins by identifying a reference point, typically the point with the smallest y-coordinate (or the leftmost point in case of ties). The algorithm then sorts all points by their polar angle relative to this reference point, ensuring a natural order for constructing the hull. Using a stack, Graham's scan processes these points sequentially, adding points to the stack while ensuring that each addition maintains the convexity of the boundary. If a new point causes the boundary to form a concave angle, points are removed from the stack until convexity is restored. The reliance on sorting gives Graham's scan a time complexity of O($n \log n$), making it particularly well-suited for static datasets where sorting overhead can be amortized.

The Quickhull algorithm employs a divide-and-conquer approach to compute the convex hull of a given set of points. The process begins by identifying the points with minimum and maximum x-coordinates defining the initial boundary segment. The algorithm then determines the point farthest from this segment, effectively partitioning the dataset into two subsets: points located to the left and right of the segment. This procedure is applied recursively to each subset, iteratively identifying the points that constitute the convex hull. Although Quickhull is well-suited for datasets with non-uniform spatial distributions, its worst-case computational complexity is O(n^2), which occurs when the point distribution leads to excessive recursion.

These algorithms form the basis for efficient convex hull computation and have been extended to handle dynamic datasets. For instance, updating the convex hull after removing a single point can often be achieved in O($n + k$), where k is the number of points on the hull boundary, avoiding the

need for a complete recomputation. Their adaptability and efficiency ensure these methods remain central to computational geometry and its applications in higher dimensions.

Chan's algorithm [53] computes the convex hull with a time complexity of $O(n \log n + n^{\lfloor d/2 \rfloor})$, making it computationally efficient for large datasets. An alternative method is described by Nielsen and Nock [55], which integrates the efficiency of divide-and-conquer strategies with the precision of geometric methods. This ensures that the convex hull can be computed in optimal time for various applications.

2.1.2 Convex hull for anomaly detection

Identifying anomalies using the convex hull is based on the observation that anomalous points tend to expand the convex hull's boundaries. From an energy perspective, this approach can be interpreted as a trade-off analysis, wherein the inclusion of additional data points results in a transition from a set with a minimal convex hull to an expanded configuration encompassing these new points. This shift highlights the balance between maintaining a compact representation and accommodating potential anomalies within the dataset. Using a support vector machine for constructing boundaries hyperplane was done by Zhang and Gu [56] and Wand et al. [38]. Using the convex hull to improve the calculation of the mean and variance to detect anomalies was done by Costa et al. [22]. Blaise et al. [31] suggest a method to detect a group of anomalies using the Voronoi diagram and the convex hull.

2.2 Anomaly detection

Anomaly detection (AD) plays a critical role in database projects. However, due to the predominantly un-supervised nature of AD, there is often no definitive solution to AD tasks in many real-world applications [57]. It can be formally defined as the task of identifying data instances $x \in X$ whose characteristics deviate significantly from those of the majority of the data, according to a predefined or learned notion of normality. AD is often defined as follows. Given a dataset $D = \{x_1, x_2, \ldots, x_n\}$ sampled from an unknown distribution $P(X)$. The goal is to learn a scoring function $s: X \to R$ such that higher values of $s(x)$ indicate a higher likelihood of x being an anomaly. Notably, this definition is not computationally strict and reveals the adoptive nature of the definition in different applied settings. As a result, a variety of methods have been developed over the years, each employing distinct strategies to identify anomalies based on specific assumptions regarding the properties of the data, the anomalies, or both. Among these methods, several algorithms have gained widespread adoption.

Isolation Forest [28] isolates individual data points through recursive partitioning of the data space, identifying anomalies based on the speed at which they can be separated. It constructs an ensemble of isolation trees, where data points that are isolated with shorter average path lengths are flagged as anomalies. *Isolation Forest* is particularly effective when anomalies are sparse and well-separated from normal data points, especially in high-dimensional spaces. However, its performance may be compromised when anomalies are densely clustered or exhibit intricate patterns, as random splits may fail to capture these underlying structures. Moreover, this method relies solely on the ordinal ranking of each variable, disregarding distances and inter-variable relationships. As a result, its effectiveness may be diminished in situations where these factors are critical. The method operates based on the topology of the data, meaning that geometrical transformations, such as stretching or shrinking, do not affect the results. Nonetheless, in many practical applications, the geometry of the data plays an important role, and omitting this consideration may lead to false discoveries of abnormalities.

Single-class SVM [58] detects anomalies by learning a boundary around normal data. Trained on data from a single class, these methods assume points outside the learned boundary are anomalous. *Single-class SVMs* construct a hyperplane that maximally separates normal data from the origin.

Gaussian mixture models (GMM) [19] detect anomalies by modeling data as a mixture of Gaussian distributions, identifying points with low likelihood under the model as anomalies. *GMM* is suitable for data clusters that approximate Gaussian shapes but may struggle with non-Gaussian clusters or complex distributions. This can be a generalization of the *Z*-score. Instead of a single bell in the Gaussian distribution, there exist a few. In real-life data, this is not common.

Local Outlier Factor (LOF) [59] measures the local density deviation of a data point relative to its neighbors, identifying anomalies in areas of significantly lower density. *LOF* is useful for data with local clusters or varying densities, but may struggle with uniform density data or when clear neighborhood structures are absent. This method detects anomalies without considering the global structure of the data. Nonetheless, detecting abnormalities globally is very important in many cases.

Density-based spatial clustering of applications with noise (DBSCAN) [33] groups data into dense regions, identifying points in sparse regions as anomalies. *DBSCAN* is effective for datasets with varying densities and distinct clusters but may perform poorly on data with uniform density or overlapping clusters.

K-means [60] partitions data into a set number of clusters, flagging points with high distances from the nearest cluster center as anomalies. This method assumes spherical clusters and works best when clusters are compact and anomalies are
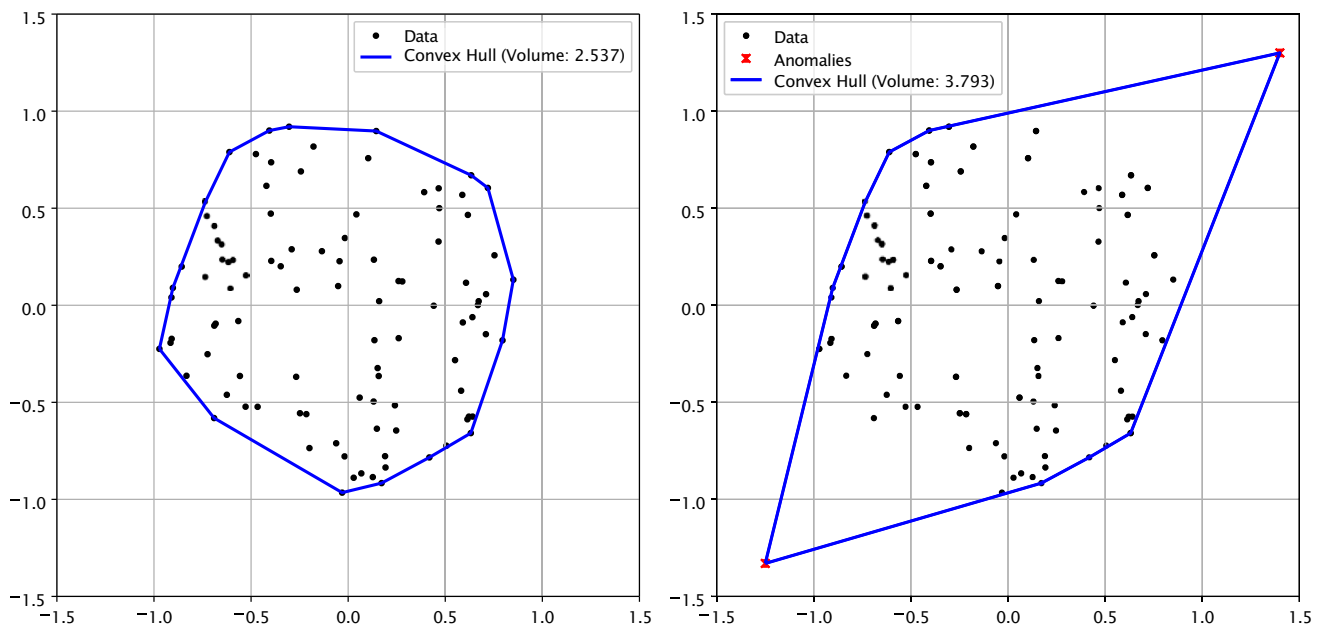
Fig. 1 A simplistic example of a 2D dataset with an anomaly and its influence on the CH's volume. On the left, there is the dataset without anomalies with a volume of 2.537, while as shown on the right, two anomaly samples increase the CH's volume to 3.793, almost 50% increase

far from cluster centers but are limited by varying cluster shapes or densities.

Mean shift [61] iteratively shifts data points toward high-density regions, identifying clusters as density peaks and labeling points outside these clusters as anomalies. *Mean shift* is effective when data has distinct density peaks but may be less effective in uniformly distributed data or data lacking prominent clusters.

The primary limitation of the clustering process is its inherent instability, which is highly sensitive to the initial conditions.

3 Convex hull method for anomaly detection

A volume-based convex hull method provides a robust framework for anomaly detection by leveraging the geometric properties of the data distribution to identify points that deviate significantly from the majority. The convex hull's volume measures the dataset's spatial extent within *n*-dimensional space. Anomalies are often located at the periphery of the data distribution, where their separation from the dense data core disproportionately inflates the convex hull's volume. Figure 1 presents a simplistic example of a 2D dataset with an anomaly and its influence on the CH's volume.

The proposed method identifies compact and densely distributed subsets of data by minimizing the convex hull volume while maximizing the number of enclosed points. This approach effectively excludes anomalous points. However, balancing these conflicting objectives—maximizing

data inclusion while minimizing the convex hull volume—poses significant challenges, akin to the sorites paradox [62] when considered in extreme scenarios.

To address these challenges, a well-defined stopping condition must be established for the iterative removal or addition of points. Alternatively, appropriate weights can be assigned to these competing objectives within an optimization framework to achieve a balance between the two goals.

Formally, to define an algorithm based on the above motivation, we first define S_p as the subset of S, a set $S \subseteq R^n$ that maximizes the volume-based objective. The problem can be stated as the following combinatorial optimization task:

$$\max_{S_p \subseteq S} f(S_p) = |S_p| - \lambda \operatorname{vol} CH(S_p), \quad \lambda > 0, \qquad (2)$$

where $CH(S_p)$ represents the convex hull of the subset S_p, and vol($CH(S_p)$) denotes its volume. The objective in Eq. (2) quantifies the trade-off between the size of the selected subset $|S_p|$ and the compactness of its convex hull, with λ acting as a sensitivity parameter. Larger λ values prioritize minimizing the convex hull volume, aiding anomaly isolation, while smaller λ values tolerate broader variations in the data distribution.

Algorithm 1 shows a pseudo-code of the proposed algorithm which accepts a dataset (S), stopping criteria (SC), and an algorithm to compute the convex hull CH; and returns the subset of samples in the dataset which is not anomaly-free. At each iteration, the convex hull $CH(S_p)$ is fully recomputed from the current subset S_p after the removal of each candidate point p. No incremental hull update strategy is used; the

recomputation is performed in full to ensure correctness and to maintain consistency with the volume evaluation.

particularly useful for comparing models fitted to the same dataset.

The computational complexity of the proposed algorithm is determined by the iterative nature of its optimization pro-

---

Algorithm 1 Modified Volume-Based Convex Hull Anomaly Detection

---

Require: Dataset $S \subseteq \mathsf{R}^n$, stopping criteria SC, convex hull method CH
1: Initialize $S_p \leftarrow S, r \leftarrow \varnothing$
2: $S_h \leftarrow \mathrm{CH}(S_p)$
3: $vol_c \leftarrow volume(S_h)$
4: $vol_m \leftarrow vol_c$
5: $S_h^{new} \leftarrow S_h$
6: $r_n \leftarrow \varnothing$
7: **repeat**
8:   **for** $p \in S_h$ **do**
9:     $S_p^n \leftarrow S_p \setminus \{p\}$
10:    $S_h^n \leftarrow \mathrm{CH}(S_p^n)$
11:    $vol_n \leftarrow volume(S_h^n)$
12:    **if** $vol_n \leq vol_m$ **then**
13:      $S_h^{new} \leftarrow S_h^n$
14:      $vol_m \leftarrow vol_n$
15:      $r_n \leftarrow p$
16:    **end if**
17:   **end for**
18:   $S_h \leftarrow S_h^{new}$
19:   $r \leftarrow r \cup r_n$
20:   **if** $SC \vee |r| = |S|$ **then**
21:     **return** $S \setminus r$
22:   **end if**
23: **until** True

Practically, one can use multiple stopping conditions, including the elbow point method [63], which locates the inflection point in a cost function curve, and the Akaike Information Criterion (AIC) [64]. The elbow point method is a heuristic technique for determining the optimal number of components (e.g., clusters) in a model by identifying the inflection point in a cost function curve, such as the within-cluster sum of squares. The method relies on plotting the cost function against the number of components and selecting the point beyond which additional components yield diminishing returns in performance improvement. This point, resembling an "elbow" in the curve, represents a balance between model complexity and fit. The Akaike Information Criterion (AIC) [64] is a statistical measure for model selection that balances goodness-of-fit and model complexity. It is defined as $\mathrm{AIC} = 2k - 2\ln(\hat{L})$, where k is the number of estimated parameters in the model and $\hat{L}$ is the maximum likelihood value. Lower AIC values indicate models that better trade-off explanatory power with parsimony, making the criterion

cess and the operations performed within each iteration. The algorithm comprises an outer loop that iterates until the stopping condition SC is satisfied or all points are removed from the dataset ($|r| = |S|$). Let T denote the total number of iterations.

Within each iteration, the algorithm computes the convex hull S_h of the remaining dataset S_p and evaluates all points within the current convex hull. For a dataset of size n, computing the convex hull requires $O(|S_p|^2)$ operations in the worst case, where $|S_p| \leq n$. The algorithm evaluates each point p in the convex hull S_h by temporarily removing it to calculate the volume of the resulting convex hull S_h^n. Each convex hull computation for $S^n = S_p \setminus \{p\}$ also has a complexity of $O(|S_p|^2)$.

Given that $|S_h|$, the number of points in the convex hull, can be as large as n, evaluating all points in S_h requires $O(|S_h| \cdot |S_p|^2)$ operations per iteration. In the worst-case scenario where $|S_h| = n$ and $|S_p| = n$, this results in $O(n \cdot n^2) =$

$O(n^3)$ operations for one iteration. The total number of iterations T is bounded by n in the worst case, as at least one point is removed from S_p in each iteration. Thus, the overall time complexity of the algorithm becomes $O(T \cdot n^3) = O(n^4)$.

The memory requirements of the algorithm are primarily determined by the need to store the input dataset, the convex hull S_h, and intermediate results. The input dataset $S \subseteq R^d$, consisting of n points in d-dimensional space, requires $O(n \cdot d)$ memory.

The convex hull S_h requires storage proportional to the number of vertices in the hull. In the worst-case scenario, where all n points are part of S_h, this also necessitates $O(n \cdot d)$ memory. Temporary storage is required for subsets such as $S^n = S_p \backslash \{p\}$, which similarly demands $O(n \cdot d)$ memory.

The computation of the volume introduces a negligible constant memory overhead. Therefore, the memory complexity is primarily driven by the storage requirements for the dataset and the convex hull, resulting in an overall memory complexity of $O(n \cdot d)$.

4 Experiments

In this section, we present the experiments conducted to evaluate the proposed CH anomaly detection algorithm. All experiments were executed on a dedicated server running Windows 10 Pro (64-bit), equipped with an Intel Core i7-10700 CPU (8 cores, 16 threads, 2.90 GHz base frequency), 32 GB of DDR4 RAM, and a 1 TB NVMe SSD for storage.

4.1 Datasets

Table 1 presents the datasets used in this study with their number of rows, cols, and portion of tagged anomalies. The datasets range from small ones with as few as 148 rows and up to medium ones with over 60 thousand rows. The number of columns also ranges from seven to forty-one columns, representing a relatively wide range of configurations. In addition, we also present the datasets that are considered appropriate to the CH algorithms, denoted by *Yes* in the last column, as these show a large reduction in the CH's volume over the first 10 steps of the algorithm.

4.2 Convex hull anomaly detection implementation

In order to explore different implementations of the proposed CH anomaly detection algorithm, we consider two properties: a dimension reduction algorithm and a stop condition. For the dimension reduction, we adopted the popular principal component analysis (PCA) [65] and the t-distributed stochastic neighbor embedding (t-SNE) [66] algorithms. As a default, we assume both reduce to a two-dimensional space. We used this step to obtain a feasible computational time for

the proposed algorithm. For the stop condition, we adopted three strategies: naive, elbow, and optimal. For the naive strategy, we stopped the process once the change in the CH's volume was less than one percent of the original change (i.e., the change in the CH's volume between the original dataset and after removing the first data point). For the elbow strategy, we performed the entire CH's computation, up to three data points, and took the elbow point of the CH's volume profile [67]. Finally, the optimal strategy is an unrealistic one and is used only to explore the model's performance, in which the number of anomaly points is known in advance, and the CH stops after the same number of steps. Importantly, we used the Euclidean distance [68] for all the experiments.

4.3 Performance

Table 2 presents the performance of all six CH configurations (for two-dimensionality reduction methods over three stooping strategies) and seven baseline anomaly detection algorithms in terms of their accuracy, F_1 score, recall, precision, area under the receiver operating characteristic (ROC) curve (AUC), and computation time (CT). Notably, the proposed convex hull algorithm performs comparably to the Isolation Forest model, which is considered the current state-of-the-art and obtains the highest F_1 score of 0.2619. More precisely, the convex hull algorithm with the optimal stopping strategy and t-SNE obtained the highest precision (and accuracy), followed by the other versions of the convex hull, and only then by the Isolation Forest algorithm. The DBSCAN algorithm obtains an almost perfect recall of 0.9968 but overall produces extremely poor results with an F_1 score of 0.0528. Expectedly, the optimal, elbow, and naive stop strategies result in a monotonic decrease in performance in terms of all metrics. In addition, the t-SNE produces better results in terms of precision and F_1 score but worse in terms of recall compared to the PCA method. Notably, the proposed CH algorithm is implemented in Python programming language without performance optimization, while the baseline algorithms are implemented in the C program language with performance optimization.

Table 3 presents a similar analysis to Table 2 but focuses only on the four datasets that are deemed convex hull friendly (see Table 1). In such settings, the convex hull with an optimal stooping strategy with the t-SNE obtains the highest F_1 score and precision of 0.1698 and 0.1491, respectively. Even the more realistic configuration of the algorithm with the elbow point stooping condition outperforms all other algorithms in terms of F_1 score and precision of 0.1657 and 0.1457, respectively. Like before, optimal, elbow, and naive stooping strategies result in a monotonic decrease in performance in terms of all metrics. In addition, the t-SNE produces better results in terms of precision and F_1 score but worse in terms of recall compared to the PCA method.

Table 1 An overview of the datasets used in this study

Name	Description	# Samples	# Cols	Portion of anomalies (%)	CH-friendly
Glass	Glass identification data	214	7	4.2	No
Ionosphere	Radar data for ionosphere detection	351	32	35.9	No
Lymphography	Medical lymphography data	148	19	4.1	No
PenDigits	Handwritten digit recognition data	9868	16	0.2	No
Shuttle	NASA shuttle anomaly data	1013	9	1.3	No
WBC	Wisconsin breast cancer data	454	9	2.2	No
Waveform	Synthetic waveform data	3443	21	2.9	Yes
KDDCup99	Network intrusion detection dataset	60,632	41	0.4	Yes
WDBC	Wisconsin diagnostic breast cancer	367	30	2.7	Yes
WPBC	Wisconsin prognostic breast cancer	198	33	23.7	Yes

Table 2 Performance of the models with comparison to baseline algorithms

Algorithm	Accuracy	F_1 Score	Recall	Precision	AUC	CT
Convex hull, naive + PCA	0.8739	0.2181	0.2702	0.2112	0.5730	17.0416
Convex hull, naive + t-SNE	0.9150	0.2337	0.2244	0.2240	0.5764	55.5415
Convex hull, elbow + PCA	0.8739	0.2181	0.2702	0.2112	0.6016	8.7743
Convex hull, elbow + t-SNE	0.9163	0.2335	0.2203	0.2261	0.5792	69.0557
Convex hull, optimal + PCA	0.8739	0.2181	0.2702	0.2112	0.5730	16.0395
Convex hull, optimal + t-SNE	**0.9154**	0.2343	0.2229	**0.2278**	0.5782	46.4837
Isolation Forest	0.9077	**0.2619**	0.6681	0.1812	**0.8866**	**1.1620**
Local Outlier Factor	0.9003	0.2110	0.6135	0.1438	0.8113	2.4859
One-class SVM	0.7665	0.1703	0.5824	0.1173	0.7191	7.1903
Gaussian mixture models	0.4131	0.0894	0.5203	0.0612	0.8863	2.2089
K-means	0.3727	0.0736	0.5086	0.0480	0.4655	2.4578
DBSCAN	0.0346	0.0528	**0.9968**	0.0283	0.5210	32.9040
Mean shift	0.0622	0.0497	0.9090	0.0267	0.4723	4.739

The best outcome for each metric is highlighted in bold

4.4 Sensitivity analysis

We computed the one-dimensional sensitivity analysis [69] of each of the parameters. Table 4 outlines the results of the analysis, presenting mean change obtained from a linear regression [70] model fitted on all the datasets ranging between 50 and 100% of the data with the smallest step size for each feature from the original value are presented in Table 1. We obtained a coefficient of determination of $R^2 = 0.906$, indicating the values of the linear regression well capture the sensitivity of the model's performance, in terms of F_1 score, to each of the model's parameters. Peculiarly, only the number of samples (rows), dimensionality reduction dimension, and stopping condition are statistically significant with $p < 0.05$.

In addition, in order to assess the robustness of the proposed CH anomaly detection algorithm against noisy data, we conducted a noise resilience analysis. Specifically, we

Table 3 Performance of the models with comparison to baseline algorithms for the four datasets detected as convex hull friendly

Algorithm	Accuracy	F_1 Score	Recall	Precision	AUC	CT
Convex hull, naive + PCA	**0.9485**	0.1452	0.1577	0.1460	0.6109	20.1798
Convex hull, naive + t-SNE	0.9462	0.1661	0.2022	0.1461	0.5940	17.9756
Convex hull, elbow + PCA	0.9507	0.1134	0.1194	0.1470	0.6144	3.4130
Convex hull, elbow + t-SNE	0.9462	0.1657	0.2015	0.1457	0.5989	19.6112
Convex hull, optimal + PCA	**0.9485**	0.1452	0.1577	0.1460	0.6109	19.6880
Convex hull, optimal + t-SNE	0.9464	**0.1698**	0.2067	**0.1491**	0.5062	17.7123
Isolation Forest	0.8856	0.0899	0.4992	0.0579	0.8871	1.0643
Local outlier factor	0.8886	0.1097	0.3523	0.0726	0.7841	0.5543
One-class SVM	0.7750	0.0569	0.4106	0.0349	0.7932	2.0343
Mean shift	0.0760	0.0484	**0.9995**	0.0261	0.4985	2.8032
Gaussian mixture models	0.4577	0.0449	0.5820	0.0249	**0.8992**	0.5632
K-means	0.4572	0.0459	0.6448	0.0253	0.4778	**0.1930**
DBSCAN	0.0344	0.0478	0.9927	0.0259	0.5070	0.2636

The best outcome for each metric is highlighted in bold

Table 4 Sensitivity analysis of the dataset and proposed model parameters

Parameter	Value	P value
# of samples	0.032	4.2×10^{-5}
# of features	-0.063	7.0×10^{-3}
Portion of anomalies	0.008	0.128
CH-friendly	0.049	0.038
Stopping condition—naive	-0.005	0.046
Stopping condition—elbow	0.001	0.033
Dimensionality reduction—PCA	-0.082	0.083
Dimensionality reduction dimension	-0.004	0.001

The table provides parameter values and their p values from a linear regression model predicting the model's average F_1 score across the datasets in Table 1

introduced Gaussian noise to the datasets at levels ranging from 0 to 5% of the data variance (with mean equal to zero). Table 5 presents the averaged results across multiple runs for the optimal configuration of the CH algorithm (t-SNE + optimal stopping condition). Each metric is reported as a function of the percentage of added noise. One can notice that the F_1 score shows a slight decline (from 0.1814 at 0% noise to 0.1604 at 5% noise), the precision remains relatively high, and recall exhibits only mild fluctuations. Accuracy remains consistently above 95%, and the AUC stays in the range of 0.57–0.59. Computation time increases steadily with the noise level.

4.5 Special cases

In Fig. 2a, the convex hull provides an accurate representation of the geometric structure of the data. Consequently, the convex hull method performs effectively in this case. In contrast, Fig. 2b, c illustrate cases involving a *torus* and a *circle* with noise, respectively. These cases are synthetically designed to highlight edge cases where the proposed convex hull model exhibits suboptimal performance. The primary reason for this limitation is that the majority of points in these sets lie on the convex hull's boundary. As a result,

Table 5 Noise resilience analysis of the convex hull (t-SNE + optimal stop condition) model

Metric/noise level	0%	1%	2%	3%	4%	5%
F_1 Score	0.1814	0.1607	0.1683	0.1616	0.1667	0.1604
Recall	0.2081	0.1695	0.1889	0.1847	0.1918	0.1814
Precision	0.4934	0.5321	0.5084	0.4929	0.4757	0.4733
Accuracy	0.9587	0.9628	0.9602	0.9602	0.9587	0.9596
AUC	0.5934	0.5770	0.5850	0.5830	0.5857	0.5810
CT (s)	49.88	52.43	53.31	54.07	55.02	55.33

Metrics are averaged across $n = 10$ repetitions and all datasets in Table 1 with noise levels ranging from 0 to 5%

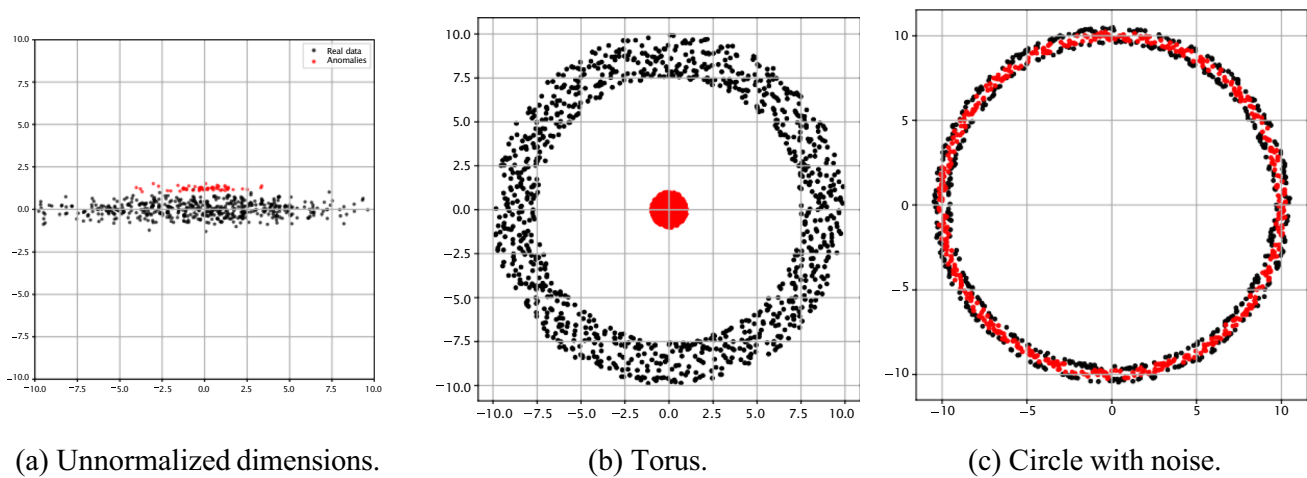(a) Unnormalized dimensions.   (b) Torus.   (c) Circle with noise.

Fig. 2 Special cases where the proposed convex hull algorithm produces poor results. The black dots indicate the in-distribution, while the red ones indicate anomalies

Table 6 F_1 score of the convex hull algorithm with the optimal stopping strategy compared to the Isolation Forest algorithm for the three edge cases shown in Fig. 2b, 2c, and 2a

Algorithm	Case	F_1 Score
Convex hull, optimal	Torus	0.0000
Convex hull, optimal	Circle with noise	0.2393
Convex hull, optimal	Unnormalized dimensions	0.0823
Isolation Forest	Torus	1.0000
Isolation Forest	Circle with noise	0.4170
Isolation Forest	Unnormalized dimensions	0.1874

processing these boundary points demands substantial computational resources. However, it is important to note that the cases are relatively rare in most real-world datasets.

Table 6 presents the F_1 score of the convex hull algorithm with the optimal stopping strategy compared to the Isolation Forest algorithm for the three edge cases shown in Fig. 2a–c. For the torus case, the convex hull receives F_1 score of 0, while the Isolation Forest achieves F_1 score of 1. The latter two cases show less dramatic differences, with around 0.18 and 0.1 differences in the F_1 score between the Isolation Forest and the convex hull algorithms.

5 Conclusions

In this study, we introduce a volume-based convex hull method for anomaly detection. The approach effectively identifies anomalies by leveraging the geometric properties of data distribution, prioritizing compact and dense subsets while excluding anomalies. This dual-objective strategy ensures that anomalies, which disproportionately increase the convex hull's volume, are systematically identified. However, balancing the competing objectives of minimizing volume and maximizing data inclusion is a non-trivial challenge that requires careful calibration of parameters and stopping conditions.

The computational complexity of the proposed method is significant, with a worst-case time complexity of $O(N^4)$, where N denotes the number of data points in the dataset. This is between one and two order(s) of magnitude compared to the compared algorithms (see Table 7 in the "Appendix" for full time complexity information). This high computational cost primarily stems from the iterative nature of the algorithm and the repeated calculation of convex hulls. However, the integration of dimensionality reduction techniques, such as Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE), as demonstrated in the experimental section, plays a critical role in alleviating these computational burdens. These techniques enable the method to achieve performance that is often comparable to, and occasionally surpasses, that of current state-of-the-art anomaly detection algorithms. Nonetheless, it is important to acknowledge the trade-offs associated with dimensionality reduction, particularly the potential distortion of the original data structure, which may affect detection accuracy.

Furthermore, the choice of stopping criteria is pivotal to the method's overall effectiveness. The algorithm's performance varies under different stopping strategies, including naive, elbow-based, and theoretically optimal approaches. While the naive strategy is straightforward to implement, its dependence on fixed thresholds may lead to either premature termination or excessive computation. The elbow method offers a more balanced alternative by dynamically responding to changes in the convex hull's volume. Although the

optimal stopping rule is impractical for real-world deployment, it provides a valuable benchmark for assessing the algorithm's theoretical performance ceiling.

The proposed algorithm considers the geometric properties of the set, particularly the importance of distances and angles. It remains invariant under geometric transformations such as translation, rotation, and reflection. In addition, it is invariant under isotropic scaling. However, transformations that do not preserve angles, such as shearing, can alter the algorithm's outcome. It is important to note that in applications where angle preservation is critical, this behavior is acceptable.

Interestingly, the proposed algorithm produces poor results for *inside* out-of-distribution cases [71], as indicated by the *torus*-shaped data with several anomaly data points in the center of the torus, unlike other algorithms such as the Isolation Forest which takes into account only the topology, not the distance and the angles. As the latter portions, the feature space locally rather than globally like the proposed convex hull algorithm.

This study is not without its limitations. First, the selection of stopping conditions remains a critical yet inherently subjective component of the proposed method. Although adaptive techniques such as the elbow method introduce a degree of flexibility, there is currently no universally accepted criterion for determining optimal stopping points across diverse datasets. This represents a compelling direction for future research. Second, the study does not extensively examine the effects of noise or overlapping anomalies, both of which may further challenge the robustness of the approach. Lastly, the experimental evaluation is primarily confined to small- and medium-scale real-world datasets. As a result, the scalability and generalizability of the method to large, complex datasets remain largely unexplored. This limitation is largely attributable to the scarcity of high-quality, labeled anomalies in such datasets. Addressing this gap should be a priority for future investigations become available.

Taken jointly, the findings of this study highlight the potential of the convex-hull-based approach as a robust and interpretable method for anomaly detection across diverse datasets. By leveraging geometric principles, the method effectively identifies anomalies while offering insights into the underlying structure of the data. Despite its computational intensity and sensitivity to certain data distributions, the proposed framework demonstrates a promising balance between accuracy and interpretability, particularly when combined with dimensionality reduction techniques.

Appendix

Baseline algorithm's hyperparameters

The baseline algorithms were configured with the following hyperparameters. For the Isolation Forest, we set `contamination=0.1` and `random_state=42`. The One-class SVM was used with `nu=0.1`, `kernel="rbf"`, and `gamma=0.1`. The Gaussian mixture models employed `n_components=2` and `covariance_type="full"`. K-means was run with `n_clusters=2` and `random_state=42`. The Local Outlier Factor used `n_neighbors=20` and `contamination=0.1`. DBSCAN was applied with `eps=0.5` and `min_samples=5`, while mean shift relied on its default parameters.

The PCA and TSNE algorithms were constructed via `initialize_reducer()`, which returns either `PCA(n_components=2)` or `TSNE(n_components=2)` with all other TSNE parameters left at scikit-learn defaults (e.g., `perplexity=30`, `learning_rate="auto"`, and no fixed `random_state`).

Baseline algorithm's time and memory requirements

Table 7 summarizes the computational requirements of the baseline anomaly detection algorithms in terms of average- and worst-case time complexity, as well as memory consumption. Here, n denotes the number of samples in the dataset and f denotes the number of features.

Table 7 Time and memory complexity of the used anomaly detection algorithms, where n is the dataset's sample size, f is the dataset's feature size

Algorithm	Time complexity (worst/average)	Memory	Source
Isolation Forest	$O(n \log (n))/O(n \log (n))$	$O(\log(f))$	[72]
Local Outlier Factor	$O(n^2)/O(n \log (n))$	$O(nf)$	[73]
One-class SVM	$O(n^3)/O(n^3)$	$O(nf)$	[74]
Mean shift	$O(n^3)/O(n^2 \log (n))$	$O(nf)$	[75]
Gaussian mixture models	$O(nf^2 + f^3)/O(nf^2 + f^3)$	$O(nf + f^2)$	[76]
K-means	$O(n^2 f)/O(nf)$	$O(nf)$	[77]
DBSCAN	$O(n^2)/O(\log (n))$	$O(nf)$	[78]

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Khan, A.Q., El Jaouhari, S., Tamani, N., Mroueh, L.: Knowledge-based anomaly detection: survey, challenges, and future directions. Eng. Appl. Artif. Intell. **136**, 108996 (2024)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
3. Zamanzadeh Darban, Z., Webb, G.I., Pan, S., Aggarwal, C., Salehi, M.: Deep learning for time series anomaly detection: a survey. ACM Comput. Surv. **57**(1), 1–42 (2024)
4. Fährmann, D., Martín, L., Sánchez, L., Damer, N.: Anomaly detection in smart environments: a comprehensive survey. IEEE Access **12**, 64006–64049 (2024)
5. Pang, G., Shen, C., Cao, L., Van Den Hengel, A.: Deep learning for anomaly detection: a review. ACM Comput. Surv. (CSUR) **54**(2), 1–38 (2021)
6. Xu, X., Liu, H., Yao, M.: Recent progress of anomaly detection. Complexity **2019**(1), 2686378 (2019)
7. Ahmed, M., Mahmood, A.N., Islam, R.: A survey of anomaly detection techniques in financial domain. Futur. Gener. Comput. Syst. **55**, 278–288 (2016)
8. Ten, C.W., Hong, J., Liu, C.C.: Anomaly detection for cybersecurity of the substations. IEEE Trans. Smart Grid **2**(4), 865–873 (2011)
9. Prasad, N.R., Almanza-Garcia, S., Lu, T.T.: Anomaly detection. Comput. Mater. Continua **14**(1), 1–22 (2010)
10. Habeeb, R.A.A., Nasaruddin, F., Gani, A., Hashem, I.A.T., Ahmed, E., Imran, M.: Real-time big data processing for anomaly detection: a survey. Int. J. Inf. Manag. **45**, 289–307 (2019)
11. Jia, W., Shukla, R.M., Sengupta, S.: Anomaly detection using supervised learning and multiple statistical methods. In: 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1291–1297. IEEE (2019)
12. Görnitz, N., Kloft, M., Rieck, K., Brefeld, U.: Toward supervised anomaly detection. J. Artif. Intell. Res. **46**, 235–262 (2013)
13. Lazebnik, T.: Pulling the carpet below the learner's feet: genetic algorithm to learn ensemble machine learning model during concept drift. Eng. Appl. Artif. Intell. **152**, 110772 (2025)
14. Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M.: Machine learning for anomaly detection: a systematic review. IEEE Access **9**, 78658–78700 (2021)
15. Fernando, T., Gammulle, H., Denman, S., Sridharan, S., Fookes, C.: Deep learning for medical anomaly detection: a survey. ACM Comput. Surv. (CSUR) **54**(7), 1–37 (2021)
16. Himeur, Y., Ghanem, K., Alsalemi, A., Bensaali, F., Amira, A.: Artificial intelligence based anomaly detection of energy consumption in buildings: a review, current trends and new perspectives. Appl. Energy **287**, 116601 (2021)
17. Hang, Yu., Zhang, Q., Liu, T., Jie, Lu., Wen, Y., Zhang, G.: Meta-add: a meta-learning based pre-trained model for concept drift active detection. Inf. Sci. **608**, 996–1009 (2022)
18. Li, P., Hang, Yu., Luo, X., Jia, Wu.: LGM-GNN: a local and global aware memory-based graph neural network for fraud detection. IEEE Trans. Big Data **9**(4), 1116–1127 (2023)
19. Li, L., Hansman, R.J., Palacios, R., Welsch, R.: Anomaly detection via a Gaussian mixture model for flight operation and safety monitoring. Transp. Res. Part C Emerg. Technol. **64**, 45–57 (2016)
20. Barnard, J.P., Aldrich, C.: Detecting outliers in multivariate process data by using convex hulls. In: Computer Aided Chemical Engineering, vol. 8, pp. 103–107. Elsevier (2000)
21. Cohen, K., Zhao, Q.: Active hypothesis testing for anomaly detection. IEEE Trans. Inf. Theory **61**(3), 1432–1450 (2015)
22. Costa, G.B.P., Ponti, M., Frery, A.C.: Partially supervised anomaly detection using convex hulls on a 2d parameter space. In: Partially Supervised Learning: Second IAPR International Workshop, PSL 2013, Nanjing, China, 13–14 May 2013, Revised Selected Papers 2, pp. 1–8. Springer (2013)
23. Alpaydin, E.: Introduction to Machine Learning. MIT Press (2020)
24. Wang, D., Shang, Y.: A new active labeling method for deep learning. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 112–119. IEEE (2014)
25. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**(1), 139–154 (2001)
26. Oza, P., Patel, V.M.: One-class convolutional neural network. IEEE Signal Process. Lett. **26**(2), 277–281 (2018)
27. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 665–674 (2017)
28. Cheng, Z., Zou, C., Dong, J.: Outlier detection using isolation forest and local outlier factor. In: Proceedings of the Conference on Research in Adaptive and Convergent Systems, pp. 161–168 (2019)
29. Boukerche, A., Zheng, L., Alfandi, O.: Outlier detection: methods, models, and classification. ACM Comput. Surv. (CSUR) **53**(3), 1–37 (2020)
30. Novello, P., Prudent, Y., Dalmau, J., Friedrich, C., Pequignot, Y: Improving out-of-distribution detection by combining existing post-hoc methods. arXiv preprint arXiv:2407.07135 (2024)
31. Blaise, A., Bouet, M., Conan, V., Secci, S.: Group anomaly detection in mobile app usages: a spatiotemporal convex hull methodology. Comput. Netw. **216**, 109277 (2022)
32. Peterson, L.E.: K-nearest neighbor. Scholarpedia **4**(2), 1883 (2009)
33. Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Trans. Database Syst. (TODS) **42**(3), 1–21 (2017)
34. Mandhare, H.C., Idate, S.R.: A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques. In: 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 931–935. IEEE (2017)
35. Avis, D., Bremner, D.: How good are convex hull algorithms? In: Proceedings of the Eleventh Annual Symposium on Computational Geometry, pp. 20–28 (1995)
36. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. (TOMS) **22**(4), 469–483 (1996)
37. Barnhill, R.E., Riesenfeld, R.F.: Computer Aided Geometric Design. Academic Press (2014)

38. Wang, T., Cai, M., Ouyang, X., Cao, Z., Cai, T., Tan, X., Lu, X.: Anomaly detection based on convex analysis: a survey. Front. Phys. **10**, 873848 (2022)

39. Li, P., Niggemann, O.: Improving clustering based anomaly detection with concave hull: an application in fault diagnosis of wind turbines. In: 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), pp. 463–466. IEEE (2016)

40. Olteanu, M., Rossi, F., Yger, F.: Meta-survey on outlier and anomaly detection. Neurocomputing **555**, 126634 (2023)

41. Novoa-Paradela, D., Fontenla-Romero, O., Guijarro-Berdiñas, B.: Online learning for anomaly detection via subdivisible convex hulls. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)

42. Jove, E., Casteleiro-Roca, J.-L., Quintián, H., Mendez-Perez, J.-A., Calvo-Rolle, J.L.: A new method for anomaly detection based on non-convex boundaries with random two-dimensional projections. Inf. Fus. **65**, 50–57 (2021)

43. Li, P., Niggemann, O., Hammer, B.: A geometric approach to clustering based anomaly detection for industrial applications. In: IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, pp. 5345–5352. IEEE (2018)

44. Liu, Z., Liu, J.G., Pan, C., Wang, G.: A novel geometric approach to binary classification based on scaled convex hulls. IEEE Trans. Neural Netw. **20**, 1215–1220 (2009)

45. Casale, P., Pujol, O., Radeva, P.: Approximate polytope ensemble for one-class classification. Pattern Recogn. **47**, 854–864 (2014)

46. He, Z., Shao, H., Cheng, J., Yang, Y., Xiang, J.: Kernel flexible and displaceable convex hull based tensor machine for gearbox fault intelligent diagnosis with multi-source signals. Measurement **163**, 107934 (2020)

47. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Comput. Surv. (CSUR) **54**(3), 1–33 (2021)

48. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online outlier detection in sensor data using non-parametric models. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 187–198 (2006)

49. Jayaram, M.A., Fleyeh, H.: Convex hulls in image processing: a scoping review. Am. J. Intell. Syst. **6**(2), 48–58 (2016)

50. Lee, D.T., Preparata, F.P.: Computational geometry—a survey. IEEE Trans. Comput. **100**(12), 1072–1101 (1984)

51. Lachand-Robert, T., Oudet, E.: Minimizing within convex bodies using a convex hull method. SIAM J. Optim. **16**(2), 368–379 (2005)

52. Cristescu, G., Lupsa, L.: Non-Connected Convexities and Applications, vol. 68. Springer (2013)

53. Chan, T.M.: Optimal output-sensitive convex hull algorithms in two and three dimensions. Discret. Comput. Geom. **16**(4), 361–368 (1996)

54. Xu, J., Zheng, Z., Feng, Y., Qing, X.: A concave hull algorithm for scattered data and its applications. In: 2010 3rd International Congress on Image and Signal Processing, vol. 5, pp. 2430–2433. IEEE (2010)

55. Nielsen, F., Nock, R.: Sided and symmetrized Bregman centroids. IEEE Trans. Inf. Theory **55**(6), 2882–2904 (2009)

56. Zhang, X.-Q., Gu, C.-H.: CH-SVM based network anomaly detection. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3261–3266. IEEE (2007)

57. Ghahramani, Z.: Unsupervised learning. In: Summer School on Machine Learning, pp. 72–112. Springer (2003)

58. Shin, H.J., Eom, D.-H., Kim, S.-S.: One-class support vector machines—an application in machine fault detection and classification. Comput. Ind. Eng. **48**(2), 395–408 (2005)

59. Alghushairy, O., Alsini, R., Soule, T., Ma, X.: A review of local outlier factor algorithms for outlier detection in big data streams. Big Data Cognit. Comput. **5**(1), 1 (2020)

60. Münz, G., Li, S., Carle, G.: Traffic anomaly detection using k-means clustering. In: Gi/itg Workshop mmbnet, vol. 7 (2007)

61. Yang, J., Rahardja, S., Fränti, P.: Mean-shift outlier detection and filtering. Pattern Recognit. **115**, 107874 (2021)

62. Hyde, D.: The sorites paradox. In: Vagueness: A Guide, pp. 1–17. Springer (2011)

63. Thorndike, R.L.: Who belongs in the family? Psychometrika **18**(4), 267–276 (1953)

64. H. Akaike. Akaike's information criterion. *International encyclopedia of statistical science*, pages 25–25, 2011.

65. Roweis, S.: EM algorithms for PCA and SPCA. Adv. Neural Inf. Process. Syst. **10** (1997)

66. Wattenberg, M., Viégas, F., Johnson, I.: How to use t-SNE effectively. Distill **1**(10), e2 (2016)

67. Bholowalia, P., Kumar, A.: Ebk-means: a clustering technique based on elbow method and k-means in WSN. Int. J. Comput. Appl. **105**(9), 17–24 (2014)

68. Ultsch, A., Lötsch, J.: Euclidean distance-optimized data transformation for cluster analysis in biomedical data (edotrans). BMC Bioinform. **23**(1), 233 (2022)

69. Peter, J.E.V., Dwight, R.P.: Numerical sensitivity analysis for aerodynamic optimization: a survey of approaches. Comput. Fluids **39**(3), 373–391 (2010)

70. Seber, G.A.F., Lee, A.J.: Linear Regression Analysis. Wiley (2012)

71. Lazebnik, T.: Introducing 'inside' out of distribution. *arXiv* (2024)

72. Tokovarov, M., Karczmarek, P.: A probabilistic generalization of isolation forest. Inf. Sci. **584**, 433–449 (2022)

73. Alghushairy, O., Alsini, R., Soule, T., Ma, X.: A review of local outlier factor algorithms for outlier detection in big data streams. Big Data Cognit. Comput. **5**(1), 1 (2021)

74. Kang, S., Kim, D., Cho, S.: Approximate training of one-class support vector machines using expected margin. Comput. Ind. Eng. **130**, 772–778 (2019)

75. Cui, Y., Cao, K., Zheng, G., Zhang, F.: An adaptive mean shift algorithm based on LSH. Proc. Eng. **23**, 265–269 (2011)

76. Gogebakan, M.: A novel approach for gaussian mixture model clustering based on soft computing method. IEEE Access **9**, 159987–160003 (2021)

77. Pakhira, M.K.: A linear time-complexity k-means algorithm using cluster shifting. In: 2014 International Conference on Computational Intelligence and Communication Networks, pp. 1047–1051 (2014)

78. Cheng, D., Zhang, C., Li, Y., Xia, S., Wang, G., Huang, J., Zhang, S., Xie, J.: Gb-dbscan: a fast granular-ball based dbscan clustering algorithm. Inf. Sci. **674**, 120731 (2024)

Springer