Pattern Recognition 115 (2021) 107874

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Mean-shift outlier detection and filtering

Jiawei Yang[a,b], Susanto Rahardja[a], Pasi Fränti[b,c,*]

[a] School of Marine Science and Technology, Northwestern Polytechnical University, China
[b] School of Computing, University of Eastern Finland, Finland
[c] School of Big Data and Internet, Shenzhen Technology University, China

ARTICLE INFO

Article history:
Received 13 May 2020
Revised 19 January 2021
Accepted 1 February 2021
Available online 8 February 2021

Keywords:
Outlier detection
Anomaly detection
Mean-shift
Medoid-shift
Clustering
Noise filtering
Outlier filtering

ABSTRACT

Traditional outlier detection methods create a model for data and then label as outliers for objects that deviate significantly from this model. However, when dat has many outliers, outliers also pollute the model. The model then becomes unreliable, thus rendering most outlier detectors to become ineffective. To solve this problem, we propose a mean-shift outlier detector. This detector employs a mean-shift technique to modify data and cancel the bias caused by the outliers. The mean-shift technique replaces every object by the mean of its *k*-nearest neighbors which essentially removes the effect of outliers before clustering without the need to know the outliers. In addition, it also detects outliers based on the distance shifted. Our experiments show that the proposed method works well regardless of the number of outliers in the data. This method outperforms all state-of-the-art methods tested, with both real-world numeric datasets as well as generated numeric and string datasets.

1. Introduction

Outliers are objects that deviate from typical data [1]. *Strong outliers* [1] are considered *anomalies*, which are expected to be detected and analyzed further. They can represent significant information and need to be detected critically in many applications such as earth science [1], fraud detection [1–3], intrusion detection [1,2], medical diagnosis [1,2], data cleaning [1,2], biological sequences [4,5], abnormal events from images [6–8] and videos [9,10], and traffic movement patterns [1,11]. They can also affect statistical analyses that are based on significance tests [12]. *Weak outliers* [1] are considered *noise*, which may harm data analysis such as clustering. In any case, regardless of strong or weak, outliers need to be detected.

In clustering, the detection and removal of outliers can be considered a preprocessing step. The idea is to clean the data from outliers that might affect the quality of cluster analysis. Another approach is to perform the clustering first and then label those objects as outliers that fail to fit into any cluster; see Fig. 1. An example of this kind of approach is DBSCAN [13]. However, this is a *chicken-and-egg problem*. Removing outliers can potentially improve clustering; on the other hand, performing clustering first would facilitate outlier detection.

Outlier detection consists of two main steps: scoring and thresholding. Generally, outlier detection approaches can be categorized into *global* and *local* outlier models based on how the reference set is constructed [14]. The *reference set* for an object is a set of the other objects used to model and calculate its outlier score. Global methods use all data objects as the reference set, whereas local methods use only a small subset of objects. The scores obtained are then sorted, and data objects with the highest scores are labelled outliers. The rest are labelled *normalities*. A decision on how many outliers are detected can be made in two ways. The *Top-N* approach uses a priori knowledge of the number of outliers and marks exactly the given number of the highest scoring objects as outliers. A more realistic approach is to derive a *threshold* from the statistical distribution of the scores, with techniques, such as *standard deviation* (SD) or *two-stage thresholding* (2T) [15].

Computing outlier scores relies on the choice of the reference objects in the dataset. *K-nearest-neighbors* (*k*–NN) based outlier detectors use neighbor objects as the reference and tune the size of the neighborhood by setting the value of *k*. However, when the number of outliers is large, most traditional detectors become ineffective. The reason is that many objects in the reference set are also outliers. In this case, relying merely on the original data to form the reference set is insufficient.

* Corresponding author.
E-mail address: pasi.franti@uef.fi (P. Fränti).

https://doi.org/10.1016/j.patcog.2021.107874

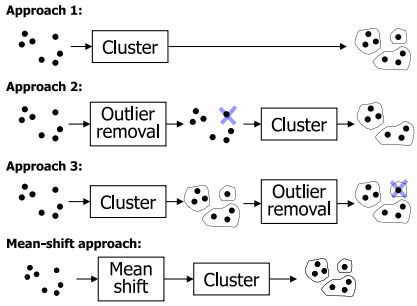

Fig. 1. Different approaches of dealing with outliers in clustering [16]: (1) ignore, (2) remove outliers in a preprocessing step, (3) detect outliers from the clustering result, (4) the mean-shift approach proposed.

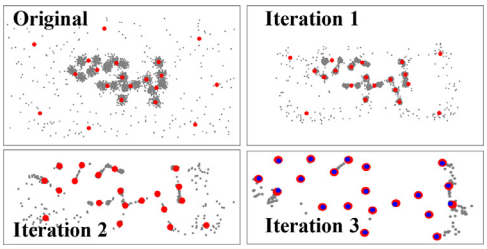

Fig. 2. Results for the random swap clustering algorithm after each mean-shift iteration on dataset **A1** with 8% of outliers. Red points are the predicted cluster centers, and blue points are the ground truth cluster centers. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

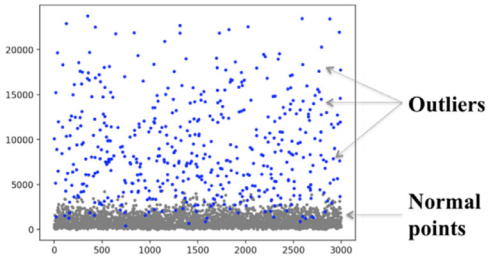

Fig. 3. Outlier scores (y axis) of the mean-shift (MOD) proposed for dataset **A1** with 16% of outliers: normal data points (gray) and outlier points (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In this paper, we propose a novel outlier detection method. The key idea is to apply mean shift (alternatively medoid-shift) as a preprocessing step. We find the k-NN for every object in the data and then replace the original object by the mean value (or medoid) of its k-NN. This process can be iterated a few times. An example is shown in Fig. 2, where we can observe improved clustering results on data preprocessed by mean shift.

We can utilize the result of mean shift in two ways. The first is to consider the shift merely as a preprocessing step that smooths the data based on neighborhood analysis. The second is to calculate the distance of the movement, i.e. the distance between the original object and its shifted version. This distance is defined as the outlier score. The larger the movement is, the more the object deviates from its neighborhood. Preliminary results of these two approaches have been presented in conferences: used as preprocessing before clustering [16], and as outlier detector [17]. For an example of the outlier scores provided by this approach, see Fig. 3.

Two variants are considered. In the *mean-shift outlier detector* (MOD) method, mean shift uses the mean value of the neighborhood. On the other hand, the *medoid-shift outlier detector* (DOD) method uses *medoid*, which is known to be more robust on noise or outliers. In data with many outliers, the reference set may contain more outliers than normalities. We therefore consider the extended neighborhood called the *extended reference set*, which includes objects both from the original dataset and their shifted versions. These revised variants are denoted as MOD+ and DOD+.

The contributions of this paper are as follows:

- The introduction of two variants, MOD and DOD; preliminary versions of these appear in [17]
- The idea of using an extended reference set leading to two improved variants, MOD+ and DOD+
- Extensive comparison with 11 baseline outlier detectors
- A wider set of experiments with 18 datasets including multidimensional, string, and real-world datasets.

The proposed outlier detection methods can be applied not only to numeric data with Euclidean distance but also to string data with edit distance. String data is challenging for k-NN-based analysis because of its high dimensionality (length of the string) and the discreteness of the edit distance measure. This measure is limited to integer values and takes only a few distinct values in a local neighborhood.

The rest of the paper is organized as follows. Existing work is briefly reviewed in Section 2. The proposed method is then introduced as a preprocessing step in Section 3 and applied for outlier detection in Section 4. Experiments are given in Section 5 and their results are shown in Section 6. Conclusions constitute Section 7.

2. Existing work

Outlier detectors can be categorized based on how the reference set is constructed [14]. A common approach is to calculate k-NN as the reference set. Here, we will discuss eleven well-known state-of-the-art outlier detectors, of which five are based on k-NN.

Distance-based outlier detectors [18–20] assume that outliers are far away from their neighbors. The detector in [18] calculates as an outlier score the distance between an object and its k^{th} neighbor. This detector is called KNN. A variant in [19] calculates the average distance to all its k-NNs. Instead of using k-NN, the detector in [20] defines a distance threshold and then counts objects within the given distance to the object. This count then serves as the outlier score.

Outlier detection using indegree of nodes (ODIN) in [19] also uses the k-NN graph. Instead of the distances, it counts how many times the object is considered as a neighbor by other objects and then uses this count as outlier score. The relationship of neighborhood corresponds to the indegree of a vertex in the corresponding k-NN graph.

As defined in [21], *Reverse unreachability* (NC as defined in [21]) is a representation-based detector. An object is represented by a linear combination of its k-NNs. This representation provides a weight matrix of how much each neighbor contributes. In the case of outliers, negative weights might be needed to represent the object. The number of negative weights is the outlier score.

Density-based detectors assume that the density of outliers is considerably lower than that of their neighbors. *Local outlier factor* (LOF) [22] calculates the relative density of an object to its k-NN and uses this as an outlier score. According to [23], LOF was the best detector among 12 k-NN-based detectors.

Some detectors are based on statistical analysis. They have the following assumption: norm objects follow the same distribution, whereas outliers do not. *Minimum covariance determinant* (MCD) [24] looks for 50% objects with the smallest scatter. The outlier score is the distance between an object and the center of these 50% objects.

While other detectors rely on distance or density, *Isolation-based anomaly detection* (IFOREST as defined in [25]) [25] con-

structs a tree of the dataset. It randomly selects a value between the minimum and maximum of a randomly selected feature and using this value, recursively splits the data. To reduce the effect of randomness, the process is repeated several times. The average path length of an object defines its outlier score.

Support vector machines (SVMs) can recognize patterns in data and can be used in the classification task. *One class support vector machine* (OCSVM) [26] trains the support vector model by treating all objects as one class. The outlier score is the distance between an object and the model. SVMs are also used to extract features to detect outliers in [27].

Principal component analysis (PCA) is an established technique in data mining. Using PCA, the variance and structure of the data can be extracted. It works by extracting the principal features of the dataset. PCA has been applied as preprocessing before clustering, for high-dimensional data [28]. It has also been used for outlier detection tasks. The *principal component analysis-based outlier detection* (PCAD) *method* [29] computes the projection of an object on the eigenvectors. It uses the normalized reconstruction error between the projected object and its original as the outlier score.

Angle-based outlier detection (ABOD) [30] analyzes the angles between an object and the remaining objects. The outlier score is the variance of the angles. It can overcome the so-called "curse of dimensionality" in high-dimensional data which is better than distance-based measures [30].

Compared to PCAD, which merely considers the top eigenvalue/eigenvector, *quantum entropy scoring* (QES) [48] considers all the eigenvalues $>>1$ in high-dimensional data. QES penalizes any object that is causing a large eigenvalue in any direction; it thus tries to find a distribution containing information about as many outlier directions as possible.

Multiple-objective generative adversarial active learning (MO-GAAL) [49] is an unsupervised outlier detection method mainly for high-dimensional data. It is based on a neural network, which is trained on a binary classification task to classify its generative data and real data. Thereafter, this trained neural network will assign possibilities to objects in real data as outlier scores and thus predict the real data.

These existing methods fall into two groups: *local detectors* (k-NN-based), and other detectors. The first group includes LOF, ODIN, NC, KNN, and ABOD. The second group contains MCD, IFOREST, OCSVM, PCAD, QES, and MO-GAAL. However, in all the said methods, the outlier objects in the reference set bias the outlier scores, especially for datasets containing large numbers of outliers.

3. Outlier filtering

In this work, we propose a simple but effective method for outlier filtering based on mean shift. These methods are mean-shift outlier detector (MOD), and its counterpart is medoid-shift outlier detector (DOD).

3.1. Mean-shift process

The mean-shift process works locally by analyzing the neighborhoods of objects, using k-NN. This method replaces every object by the mean of its k-NNs, forcing objects to move towards denser areas. The distance of the movement can serve as evidence of being an outlier. An object with a bigger movement is more likely to be an outlier. The mean-shift process is shown in Algorithm 1.

This idea is closely related to mean-shift filtering used in image processing [31] and to the mean-shift clustering algorithm [32]. Mean-shift filtering takes pixel value and its coordinates as the feature vector, and it transforms each feature vector towards the mean of its neighbors. The method has been proven to be useful in detecting fingerprint and contamination defects in multicrystalline solar wafers [33]. This idea also resembles the low-pass and median filtering used for image denoising.

Algorithm 1 Mean-shift process (MS (X, k, I)).

Input: Dataset X, neighborhood size k, iterations I
Output: Modified X^*
REPEAT I TIMES:
  FOR $X_i \in$X:
    k-NN(X_i) $\leftarrow$ Find its k-nearest neighbors $\in X$;
    $M_i \leftarrow$ Calculate the mean of the neighbors k-NN(X_i);
    $X^*_i \leftarrow M_i$ (replace x_i by the mean M_i), $X^*_i \in X^*$;

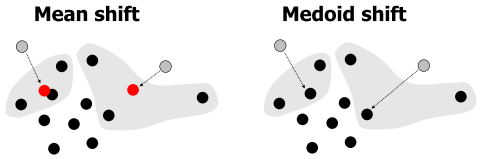

Fig. 4. Effect of the mean (left) and medoid (right) with two sample points in a neighborhood with outliers [16].

3.2. Mean or median?

Both mean and median have been used in the mean-shift clustering concept [32,34]. Mean can be used when dealing with numeric data. However, its main disadvantage is that it can cause a blurring effect; outlier objects bias the calculation of the mean of clean objects. The median is less sensitive to this bias because a single outlier object in the neighborhood may not affect the calculation of the median at all. Therefore, it has lesser effect on clean objects. Another benefit of the median is that when repeated until converging, it can reach the root signal [35]. However, it is unclear how the median can be computed in multidimensional data. To improve this point, we use the *medoid*, which is the object in a set that has the minimum total distance to all the other objects in the set. A disadvantage of the medoid is that its calculation can be time-consuming. An example of using the mean and medoid is shown in Fig. 4.

3.3. Mean-shift outlier filtering

The mean-shift process has been applied for outlier filtering in [16]. The idea is to apply Algorithm 1 and modify the data so that the effect of outliers is minimized. Being an implementation as a separate preprocessing step, it is therefore independent of the choice of clustering method. This process can be iterated several times for a stronger outlier removal effect. The number of iterations is the parameter. Based on our experiments in Section 6.3, for typical clustering data, three-iterations is the best choice.

This iterative variant is similar to the ORC algorithm [36] which iteratively removes the most remote objects in each cluster. The difference is that, in ORC, outliers are chosen based on the distance to their cluster centroid in the intermediate clustering solution. Therefore, if the cluster is not yet correctly determined, objects can be falsely removed.

In our method, instead of relying on the clustering process, we remove objects based on their local neighborhood. Fig. 5 demonstrates this process. A small number of outlier points can form a cluster (Fig. 5), and the distances between the outlier points to the centroid can be very small. ORC will therefore falsely remove normality points that have bigger distances to their centroids. Fig. 5 shows an example in which despite 90 iterations, the outliers still remain. ORC on the other hand has by then removed a lot of normality points.

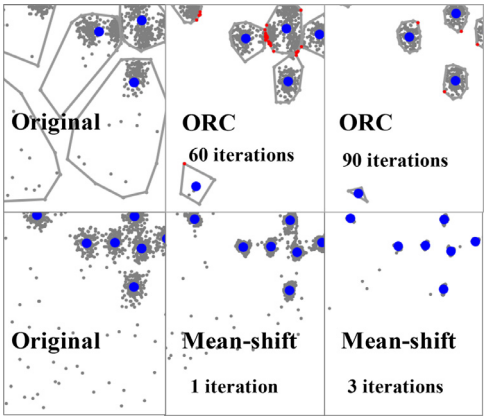

Fig. 5. Example of the iterative process of ORC (up) and the mean shift proposed (down) on a part of dataset **A1** with 8% outliers.

Algorithm 2 Mean-shift outlier detector (MOD (*X*, ***k***)).

Input: Dataset *X*, neighborhood size *k*
Output: Outlier score *S*
$Y \leftarrow$ MS(*X*, *k*, 3) (Algorithm 1);
FOR $X_i \in X$:
$S_i \leftarrow |X_i - Y_i|$, $Y_j \in Y$, $S_i \in S$;

4. Outlier detection

The idea of using mean shift is next applied to outlier detection. The mean-shift process here (Algorithm 1) is the same as for outlier filtering, but instead of only modifying objects, we calculate their outlier scores.

4.1. Mean-shift outlier detection

Mean-shift clustering [32] iterates the mean-shift process until convergence to clustering data. In [37], mean-shift clustering is applied first, and clusters having objects less than the threshold value are treated as outlier clusters. We apply the same process, but we do not cluster the data. Instead, because we aim at finding outliers, we use the processing result merely for analysis purposes. Specifically, we compare the location of the object before and after shifting. This difference is defined as the outlier score.

The pseudo-code of the method is summarized in Algorithm 2. We can see that the reference set contains both the original data in *X* and the modified data in *Y*. An example of the outlier scores is given in Fig. 6. Compared to [37], the proposed method differs

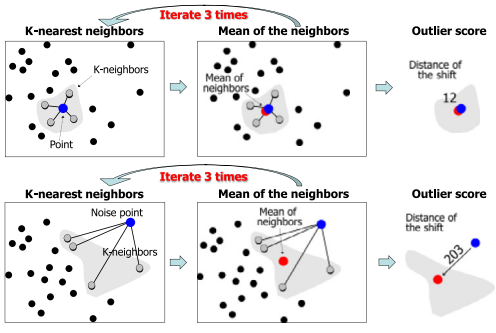

Fig. 6. Example of MOD with parameters $k = 4$ and *iterations* =1. The outlier score is calculated as the distance of the original object (blue point) from its mean-shifted version (red point). The point below has a clearly bigger outlier score (203) than the point above does (12). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

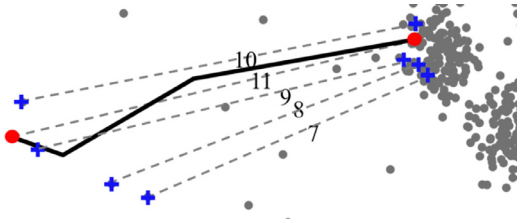

Fig. 7. *K*-NN before and after the mean-shift process on a part of dataset A1 with 8% of outliers. The black line shows the process of shifting of the red point. The dashed line shows the distance of the neighbors before and after shifting (blue crosses). The outlier score of the red point is the sum of the shifting distances of its neighbors: 10+11+9 + 8 + 7 = 45. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

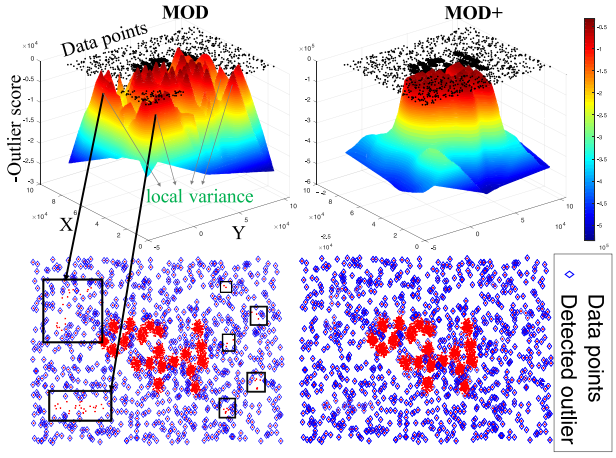

Fig. 8. Outlier score of objects (top row). Detected outliers are marked by blue diamonds (below row). MOD (left column) has local variations and fails to detect several outliers in the rectangle while MOD+ (right column) can detect most outliers correctly. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in three ways. First, the motivation in [37] is merely clustering while the techniques proposed here aim at detecting outliers in general. Second, the method in [37] applies the mean-shift process until convergence while the techniques proposed apply it only for a few iterations. Finally, instead of only modifying the data, the techniques proposed also provide outlier scores for objects.

4.2. The extended reference set

When data abounds with outliers, using *k*-NN as the reference set can bias results. Having many outliers in the same neighborhood can cause outliers to form fake clusters. As a result, some outliers can have only a small movement when their neighbors are equally distributed everywhere around them. However, this is unlikely to happen to all points.

To overcome this problem, we extend the reference set by including not only the nearest neighbors themselves but also their shifted versions, see Fig. 7. If the neighbors are also outliers, many of them are likely to be shifted by a greater distance towards the fake cluster. This effect is enough to remove the potential bias in areas with many outliers.

Fig. 8 and Fig. 9 demonstrate the effect of applying the extended reference set. While MOD fails to detect the objects in the black rectangle in these figures, the extended reference set succeeds in doing so. Another example is point *a* in Fig. 9. It is shifted only by a short distance to point *A* because the nearest neighbors are evenly distributed around point *a*.

To improve these situations, we extend the reference set by including not only the original *k*-NN neighbors but also the modified

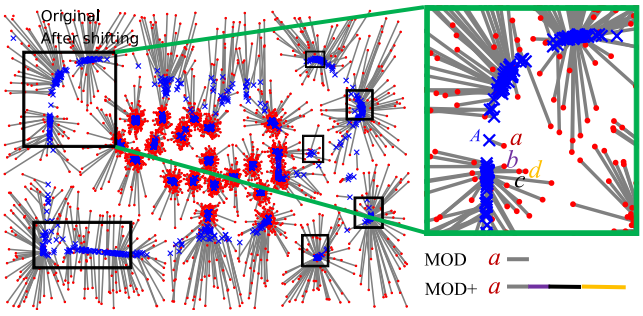

Fig. 9. Original points (red) and their shifted versions (blue) are connected by gray lines. Point *a* is shifted only a short distance to point *A*, resulting in a low outlier score. MOD+ computes the outlier score as the sum of the movement of point *a* with that of its three nearest neighbors (point *b, c*, and *d*). The outlier score, therefore, becomes bigger. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Algorithm 3 Mean-shift outlier detector + (MOD+ (*X*, ***k***)).

Input: Dataset *X*, neighborhood size *k*
Output: Outlier score *S*
$Y \leftarrow$ MS(X, *k*, 3) (Algorithm 1);
FOR $X_i \in X$:
  k-NN(X_i) $\leftarrow$ Find k-nearest neighbors of X_i;
  $S_i \leftarrow | X_i - Y_i | + \Sigma_j | X_j - Y_j |$, $X_j \in k\text{-}NN(X_i)$, $Y_j \in Y$, $S_i \in S$;

k-NN neighbors *after* the mean shift. The *extended reference set*, therefore, contains more information from the data. This feature makes it more robust in the case of data with many outliers. The variants of MOD and DOD use the extended reference set (MOD+ and DOD+); these variants are summarized in Algorithm 3.

The outlier score of an object is calculated here as the sum of the shifting distances of both the object and its *k*-NNs. MOD+ assumes that the possibility of an object being an outlier can be corrected or enhanced by considering its neighbors. An example of the difference between MOD and MOD+ is shown in Fig. 9. We can see MOD fails to detect many outliers while MOD+ succeeds. Point *a* in Fig. 9 shows how MOD+ can increase the possibility of a point being an outlier by considering its neighbors' movement. After we sum up its movement with that of its neighbors, point *a* has a much bigger outlier score than the previous, thus reflecting its possibility of being an outlier better.

4.3. String data

K-NN-based methods rely on distance calculations; therefore, they can straightforwardly be applied to string data as well. To calculate the distance between strings, we used the well-known *edit distance*. However, complications arise when we need to calculate the *mean*, which is not obvious in non-numeric spaces. For this reason, we use the *medoid* within the proposed method. The medoid is the object with the minimum total distance to all other objects in the *k*-nearest neighborhood. The following methods are applicable for strings: KNN, ODIN, DOD, and DOD+.

Fig. 10 shows an example of the process and the corresponding result. For example, the word *ysys* is shifted to the word *finan*. Their edit distance is 5; hence, the outlier score of *ysys* is 5. Similarly, *denmark* is shifted to the word *denmark* itself; hence, the edit distance is 0, and the corresponding outlier score is 0.

4.4. Discussion

K-NN-based methods [19–22, 30] differ in how they use *k*-NN information. The KNN detector [18] can be considered as a special case of our method but with one iteration of mean shift. Our method differs from it in three aspects. First, we iterate mean shift multiple times; second, we also consider the medoid instead of just the mean. Finally, our method can be applied also as a preprocessing technique; in this way, it can avoid the need for threshold selection [15]. From this perspective, the proposed method is more general than other methods based on *k*-NN.

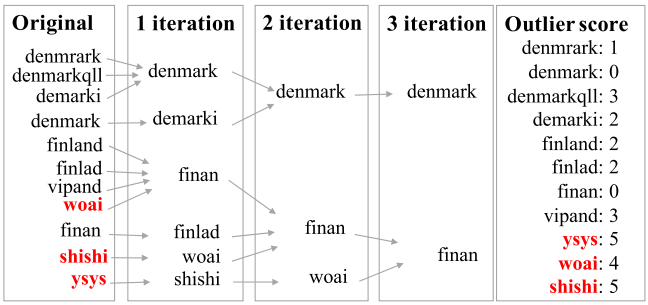

Fig. 10. An example of medoid-shift outlier scores with $k = 3$ when using edit distance for strings from the **Countries** dataset. The arrows show the medoid-shift process. The outlier scores are calculated as the edit distance between the strings before and after three iterations of the medoid shift. Black strings are the original (normalities) and the red strings outliers.

We also note that the concept of *mean shift* [50] has also been introduced in the classical least squares in regression when handling outliers [51–54]. However, this *mean shift* [50] is a concept totally different from the *mean-shift process* we have adopted from [31,32,34]. In [50], *mean shift* is used as a name of a parameter specific to each observation in the regression function. If the value of the parameter is non-zero, the observation associated with the parameter is an outlier. The technique proposed in [50] is specific to regression tasks with sequential data and it directly outputs labels other than outlier scores for outliers and normalities. It does not generalize high-dimensional data and cannot be applied here.

5. Experimental set-up

5.1. Methods

First, we tested the proposed mean-shift outlier filtering algorithm as a preprocessing method with two clustering algorithms: *k-means* [38] and *random swap* [39]. Both algorithms minimize sum-of-squared errors. The first one is commonly used but does not always find the correct clustering, even with clean data. The second one does find the correct results with all datasets tested with clean data. In the case of data with outliers, all errors in the clustering are caused by outliers. We compared the proposed method to the outlier detectors from Section 2, summarized in Table 1. ABOD, QUE, and MO-GAAL are compared with high-dimensional data, for which they were mainly developed.

Table 1
BASELINE OUTLIER DETECTORS.

Method	Type basis	Data	Publication and year
KNN [18]	Distance	N/S	ACM SIGMOD, 2000
LOF [22]	Density	N	ACM SIGMOD, 2000
ODIN [19]	Graph	N/S	ICPR, 2004
NC [21]	Representation	N	IEEE-TNNLS, 2018
MCD [24]	Statistical	N	J. A. Stat. Assoc, 1984
IFOREST [25]	Tree	N	TKDD, 2012
OCSVM [26]	SVM	N	Neural computation, 2001
PCAD [29]	PCA	N	ACM, 2000
ABOD [30]	Angle	N	KDD, 2008
QES [48]	Entropy	N	NeurIPS, 2019
MO-GAAL[49]	Neural network	N	IEEE-TKDE, 2020
DOD/MOD [16,17]	Shifting	N/S, N	ICAISC, FSDM, 2018
DOD+/MOD+	Shifting	N/S, N	Proposed

N = numeric data, *S* = string data.

Table 2
DATASETS GENERATED.

Dataset name	Data size	Cluster	Dim	Type
S1-S4	5000	15	2	Numeric
A1-A3	3000–7500	20, 35, 50	2	Numeric
B1-B2	100,000	100	2	Numeric
Unbalance	6500	8	2	Numeric
XOR	2000	4	2	Numeric
Countries/N	2400	48	–	String
REAL-WORLD DATASETS FOR OUTLIER DETECTION				
Dataset name	Data size	Outliers	Dim	Outlier object
KDD-Cup99	60,632	246	38	Network attack
Stamps	340	31	9	Forged stamps
PageBlocks	5473	560	10	Pictures or graphics
Pima	768	268	8	Patients
Arrhythmia	450	206	259	Affected patients
Parkinson	195	147	22	Patients

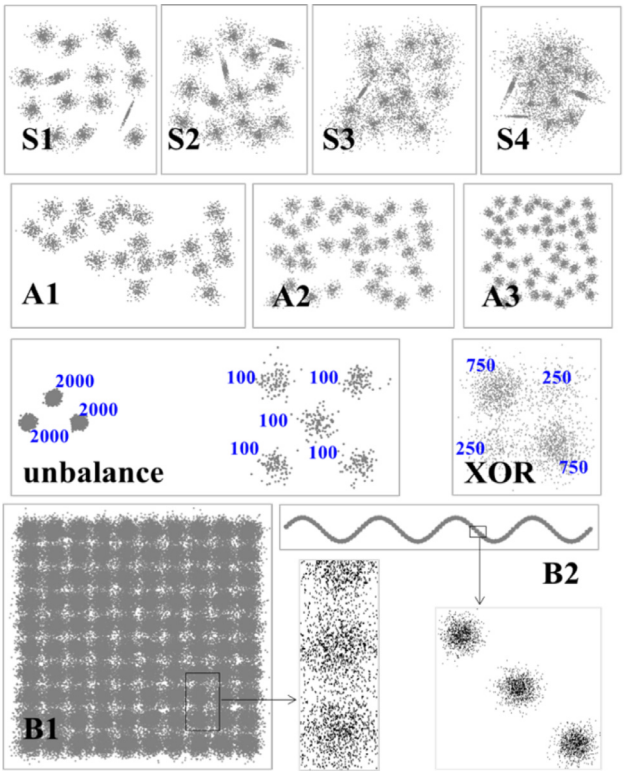

Fig. 11. Two-dimensional datasets used in the experiments.

5.2. Datasets

We used 11 clustering datasets, as shown in Table 2. For a visualization of the 2-D datasets, see Fig. 11. The **S** sets have varying levels of cluster overlap; the **A** sets have varying numbers of clusters. The **B** sets have varying shape; the **unbalance** and **XOR** datasets [40] have clusters with different densities. Most of these datasets can be found in the basic clustering benchmark in [41]; the **XOR** dataset originates from [40]. All these datasets are available on the web.[1]

The ***Countries***[2] dataset contains modified copies of the names of the 48 European countries. The modifications are random insert and delete operations. The number of operations is 30%, so the resulting strings can still be identified with some effort. Outliers are then added by generating fake strings so that their length and character distribution resembles those of the real country names, but their content has no intelligible meaning. Experiments with 10%, 20%, 30%, and 40% of outliers are then tested.

In addition to these artificial datasets, we used six public real-world semantically meaningful datasets from [23], summarized in Table 2. We removed duplicates and scaled every attribute by subtracting the mean and dividing by standard deviation. The dimensions vary from 8 to 259.

[1] http://cs.uef.fi/sipu/datasets/
[2] http://cs.uef.fi/sipu/string/countries/

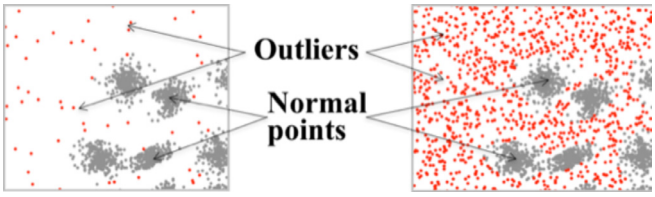

Fig. 12. Part of dataset **S1** with 8% (left) and 128% (right) of outliers. Gray points are normalities, and red points are outliers. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.3. Measurement

We evaluated the clustering results by using *the centroid index* (CI) [42]. It is a cluster-level measurement, which counts the number of wrongly located clusters. The value CI=0 indicates that, at the cluster level, the clustering is correct with respect to the ground truth. We evaluated outlier detection methods by using the *receiver operating characteristic* (ROC), a 2-dimensional plot of the true-positive rate against the false-positive rate, over varying thresholds. The curve can also be summarized by a single value known as the *area under the curve* (AUC), which ranges between 0 and 1. A perfect ranking of database objects would result in an AUC value of 1, and the worst possible ranking would produce a value of 0.

5.4. Outlier model

Our method does not rely on any assumption of the distribution of outliers. For simplicity, we use uniformly distributed random outliers added to the **A, B, S, Unbalance,** and **XOR** datasets. Random values are generated in each dimension between [x_{mean}-$2\cdot range$, x_{mean}+$2\cdot range$], where x_{mean} is the mean of all data points, and *range* is the maximum distance of any point from the mean: $range = \max(|x_{\max}- x_{\text{mean}}|, |x_{\text{mean}}- x_{\min}|)$. An example of a dataset with outliers is shown in Fig. 12.

6. Results and discussion

The experimental results are summarized in Tables 3–9. Clustering results are also shown in Table 3. Tables 4,5,6, and 7 summarize outlier detection results. The relationships between the percentage of outliers and the number of shifting iterations are summarized in Table 7 and 8 and Fig. 13. The running times of the algorithms are summarized in Table 11.

6.1. Clustering results

Clustering results are summarized in Table 3. Neighborhood size is fixed at $k = 30$, and Top-N is fixed at half of the number of outliers. For example, with 8% of outliers, we select the top 0.04^*N points with the highest outlier score as outliers.

For the results in Table 3, we have several observations. First, all datasets can be perfectly clustered when using the better random swap algorithm if there is no outlier (CI=0 for 0% of outlier). When

Table 3
SUMMARY OF THE CLUSTERING RESULTS WITH DIFFERENT PERCENTAGE OF OUTLIERS. THE NUMBERS ARE AVERAGE **CI VALUES** FOR THE 2-D CLUSTERING DATASETS IN TABLE 2.

RANDOM SWAP The percentage of outliers												
Type	Method	0%	0.025%	0.5%	1%	2%	4%	8%	16%	32%	64%	128%	Avg.
None	None	0.00	1.36	2.55	3.82	5.18	7.64	10.09	12.45	14.27	17.73	20.45	9.55
Outlier filtering	DOD	0.18	0.36	2.18	1.64	2.55	4.27	6.55	9.73	13.09	16.18	19.73	7.63
	MOD	0.18	0.18	1.00	1.64	2.45	4.00	6.00	9.82	12.91	16.36	19.64	7.40
Outlier removal	LOF	–	0.73	1.36	2.64	4.00	6.73	9.64	13.27	16.27	19.73	23.18	9.76
	ODIN	–	1.18	2.36	3.27	5.09	7.45	9.55	12.55	15.82	18.27	20.73	9.63
	NC	–	1.73	2.18	3.27	4.64	7.18	9.55	12.73	15.64	17.82	20.64	9.54
	KNN	–	0.27	1.00	1.73	2.91	3.36	5.73	8.45	11.82	15.18	18.27	6.87
	MCD	–	0.91	1.55	2.27	3.64	4.91	6.82	8.64	10.55	13.64	16.18	6.91
	IFOREST	–	1.91	2.45	3.18	4.55	6.09	7.73	9.55	12.00	13.73	15.64	7.68
	OCSVM	–	1.55	2.18	2.73	4.09	5.73	7.73	8.45	10.91	12.27	12.36	6.80
	PCAD	–	1.73	2.55	3.45	4.73	6.64	7.27	9.09	11.18	12.91	14.64	7.42
K-MEANS The percentage of outliers													
Type	Method	0%	0.025%	0.5%	1%	2%	4%	8%	16%	32%	64%	128%	Avg.
None	None	3.27	3.64	4.73	5.00	4.91	4.73	7.45	9.27	11.36	14.73	18.55	8.44
Outlier filtering	DOD	5.55	4.27	3.73	3.18	4.64	5.00	6.00	7.27	8.91	13.09	15.73	7.18
	MOD	5.45	3.64	3.91	4.09	4.45	5.64	6.45	7.27	10.18	12.82	16.18	7.46
Outlier removal	LOF	–	3.73	3.36	3.73	4.91	5.36	9.36	9.36	12.73	17.45	20.55	9.06
	ODIN	–	3.91	3.91	4.82	4.36	6.09	9.36	8.73	11.91	14.91	18.73	8.67
	NC	–	3.36	4.09	4.45	4.55	6.27	9.36	9.00	12.27	15.09	19.91	8.84
	KNN	–	3.45	4.00	4.18	4.55	5.45	6.55	7.45	9.00	11.73	16.09	7.25
	MCD	–	3.00	4.09	4.36	4.91	5.00	5.36	6.73	8.55	11.09	14.73	6.78
	IFOREST	–	2.00	2.45	3.09	4.45	6.27	7.36	9.45	11.36	13.36	15.45	7.53
	OCSVM	–	1.55	2.18	2.91	4.45	5.82	7.64	8.82	10.55	11.91	12.36	6.82
	PCAD	–	1.73	2.45	2.82	4.55	6.27	7.55	9.73	11.27	12.82	14.82	7.40

Table 4
BEST **AUC** FOR REAL-WORLD DATASETS WITH k RANGING FROM 2 TO 100.

Dataset (outliers)	KDD. (0.4%)	Stamps (9.1%)	Page. (10.2%)	Pima (34.9%)	Arrh. (45.8%)	Parkinson (75.4%)	AVG
DOD+	0.98	0.94	0.91	0.78	0.73	0.71	0.84
MOD+	0.99	0.94	0.91	0.76	0.74	0.68	0.84
DOD	0.78	0.92	0.89	0.71	0.74	0.71	0.79
MOD	0.99	0.90	0.91	0.68	0.74	0.67	0.82
LOF	0.84	0.89	0.91	0.69	0.73	0.61	0.78
ODIN	0.81	0.83	0.79	0.63	0.72	0.53	0.72
NC	0.69	0.68	0.70	0.57	0.67	0.61	0.65
KNN	0.99	0.91	0.92	0.73	0.74	0.64	0.82
ABOD	0.78	0.73	0.75	0.68	0.71	0.61	0.71
MCD	0.97	0.85	0.92	0.68	0.72	0.65	0.80
IFORES.	0.99	0.86	0.90	0.67	0.76	0.49	0.78
OCSVM	0.99	0.87	0.91	0.62	0.74	0.36	0.75
PCAD	0.99	0.90	0.90	0.63	0.73	0.38	0.76
QES	0.96	0.89	0.77	0.60	0.66	0.36	0.71
MO-GAAL	0.85	0.38	0.25	0.69	0.47	0.75	0.57

Table 5
BEST **AUC** FOR THE **COUNTRIES** DATASET WHEN K RANGES FROM 2 TO 100.

Method	The percentage of outliers							
	10%		20%		30%		40%	
	AUC	k	AUC	k	AUC	k	AUC	k
DOD+	0.84	2	0.83	4	0.83	5	**0.85**	5
DOD	0.86	2	0.84	4	**0.84**	5	0.83	5
KNN	0.82	2	0.80	2	0.79	2	0.80	2
ODIN	**0.88**	28	**0.85**	30	0.83	37	0.81	32

changing the clustering algorithm to k-means, we can also see the effect of the inferior clustering algorithm. Even with clean data, the k-means algorithm yields CI=3.27 errors, on average.

However, with outliers present in the data, the performance difference of the clustering algorithms disappears, and both random swap and k-means produce very similar results. The more outliers in the data, the more the clustering result deteriorates. With 8% of outliers, we already have errors with 10 clusters (CI=10.09), on average; when the percentage of outliers reaches 128%, errors double (CI=20.45).

The exceptions are NC, ABOD, IFOREST, OCSVM, and PCAD, all of which perform slightly worse with 0.025% of outliers. Overall, the medoid-shift variant (DOD) proposed performs best with a low percentage of outliers, up to 2%. KNN works best with an intermediate percentage of outliers (4%–8%), and MCD and OCSVM work best with a higher percentage of outliers (16%–128%).

Our second observation is that both the proposed mean-shift filtering and all the outlier removal methods improve the clustering but only up to a point where the percentage of outliers is 8%.

Table 6
OUTLIER SCORES FOR TOY EXAMPLE IN FIG. 10 ($k = 3$).

Strings	DOD+	DOD	KNN	ODIN
denmrark	1	1	3	3
denmark	1	0	2	4
denmarkqll	2	3	**4**	**1**
demarki	1	2	3	4
finland	1	2	2	4
finlad	1	2	2	3
finan	1	0	2	5
vipand	2	3	3	**1**
ysys	**5**	**5**	**5**	3
woai	**3**	**4**	**4**	3
shishi	**5**	**5**	**5**	**2**
SUCCESS	**YES**	**YES**	**NO**	**NO**

Thereafter, most methods start to lose their effectiveness in improving clustering.

The choice between the mean and medoid seems insignificant. First, when using *k*-means, medoid shift (DOD) is sometimes better than mean shift (MOD). However, when using random swap, the situation is the opposite. Second, *k*-means becomes slightly better than random swap, when the number of outliers increases beyond 8%. This aspect is a side effect of the *k*-means algorithm. Adding outliers generates fake low-density clusters, in which a good algorithm can identify more effectively. However, *k*-means has problems when clusters have less overlap or varying density [41]. It is therefore less efficient in finding fake clusters, which appears as a positive effect but this is merely a side effect of its inferior optimization capability.

We conclude that to have perfect clustering performance, one must have a good algorithm and clean data. When the percentage of outliers becomes high, no outlier detection can fix all the problems. The choice of the clustering algorithm also becomes less significant.

One requirement for a good outlier removal method is that it should not destroy clean data. In our case, slight errors were detected in the case of 0% of outliers. CI=0 increased to CI=0.18, on average, when mean shift was applied to clean data. However, most of these errors are originated from datasets **B1** and **B2**.

The main problem of traditional outlier removal methods is that they are based on thresholding and require knowing the percentage of outliers to set up the Top-*N* parameter. If the correct number of outliers is used, their performance is close to that of our method. However, if some default value, such as 1% or 8% is applied, they (start to) fail much more severely. This feature highlights the importance of outlier filtering (our approach) compared to traditional outlier removal.

6.2. Results for outlier detection

Outlier detection results are summarized in Table 4 for the high-dimensional real-world datasets and in Table 5 for the **Countries** string datasets. We tested all *k*-NN-based outlier detectors, using all values of *k* between 2 and 100. The value of *k* that provides the best result is chosen for each outlier detector. For the other detectors, we use their default parameter settings found in the literature.

For results of the high-dimensional real-world datasets in Table 4, the proposed methods are far better than the other methods. For instance, both MOD and KNN reach an AUC value of 0.99 with the **KDD-Cup99** data, which has only 0.4% outliers. However, for the **Parkinson** dataset (75.4% outliers), medoid-shift and MO-GAAL perform better than others (AUC>0.70). MOD+ and DOD+ are slightly better than MOD and DOD (0.84 vs. 0.82), on average.

For the **Countries** dataset, the results when using edit distance are summarized in Table 5. We can see that when the percentage of outliers is low (10% and 20%), ODIN outperforms KNN, DOD, and DOD+. However, when the percentage of outliers is high (30% and 40%), the proposed DOD and DOD+ outperform KNN and ODIN. An example with 30% of outliers is provided in Table 6. We can observe that both KNN and ODIN fail to detect outliers, but the proposed DOD and DOD+ succeed.

From Fig. 10, we can note that after three iterations, many normality objects such as *Denmark,* have already been shifted to their original variant. This produces a small outlier score. True outlier words such as *ysys*, will keep on being shifted further away. This shift results in a bigger outlier score than that of normality objects. The other methods have been defined only using Euclidean distance and not edit distance. Therefore, we have not compared the proposed method to these other methods on string data.

Table 8
THE EFFECT OF ITERATIONS ON DIFFERENT PERCENTAGES OF OUTLIERS IN THE **COUNTRIES** DATASETS (**AUC**).

	The percentage of outliers			
Iteration	10%	20%	30%	40%
1	**0.86**	0.76	0.76	0.75
2	**0.86**	0.84	**0.84**	0.83
3	**0.86**	**0.85**	**0.84**	**0.84**

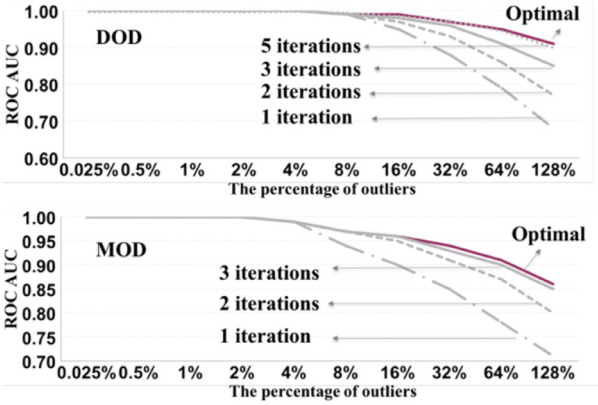

Fig. 13. The effect of the percentage of outliers on the number of shifting iterations needed. Results are shown for medoid shift (above) and mean shift (below).

Table 7
OPTIMAL ITERATIONS FOR AVERAGE RESULTS OF DATA SET **S1** WITH PERCENTAGE OF OUTLIERS (REPEATED 99 TIMES). (**ITERATIONS**).

	The percentage of outliers										
	0.025%	0.5%	1%	2%	4%	8%	16%	32%	64%	128%	Avg.
DOD	1	1	1	1	1	1	5	4	5	7	2.7
MOD	1	1	1	1	1	2	3	4	4	4	2.2

Table 9
AUC FOR THE REAL-WORLD DATASETS WHEN $k = 6^{*}\mathrm{LOG}(n)$.

Dataset (outliers)	KDD. (0.4%)	Stamps (9.1%)	Page. (10.2%)	Pima (34.9%)	Arrh. (45.8%)	Parkinson (75.4%)	AVG
DOD+	0.79	0.89	0.89	0.71	0.73	0.63	0.78
MOD+	0.94	0.91	0.89	0.73	0.72	0.66	0.81
DOD	0.69	0.81	0.80	0.68	0.73	0.68	0.73
MOD	0.92	0.78	0.84	0.65	0.74	0.60	0.75
LOF	0.60	0.53	0.73	0.60	0.73	0.56	0.62
ODIN	0.61	0.58	0.62	0.56	0.70	0.45	0.59
NC	0.58	0.50	0.52	0.52	0.60	0.46	0.53
KNN	0.96	0.89	0.92	0.72	0.74	0.54	0.79
ABOD	0.82	0.81	0.80	0.70	0.72	0.64	0.75

To sum up, the proposed methods (MOD and DOD) outperform others in most cases, regardless of the type and dimensionality of the data. They perform well, especially for real-world datasets that have large number of outliers.

6.3. Effect of shifting iterations

We also study the relationship between the percentage of outliers and the number of shifting iterations. We repeat the experiment 99 times with dataset **S1,** using random outliers on each percentage. The average AUC results are shown in Fig. 13 and Table 7. We can see that, with the increasing of the percentage of outliers, more shifting iterations are needed to reach optimal performance and experiment shows that for MOD and DOD, the number of iterations is three iterations and five respectively.

The smallest numbers of iterations required to reach the best performance are summarized in Table 7. To reach the percentage of outliers of 16% for MOD and 8% for DOD, three iterations are sufficient. With a low percentage of outliers, one iteration would suffice. Additional iterations would not yield any additional benefit but they do no harm either except for a slightly longer processing time. Based on these results, we fix the number of iterations at three as a default value.

For the **Countries** datasets, the effect of the number of iterations is shown in Table 8. We can see that three iterations provide the best results; due to the more discrete nature of the data, additional iterations are less critical. The difference in performance when using two and three iterations is only marginal. Nevertheless, one iteration is still too few, except with the lowest percentage of outliers (10%).

6.4. Neighborhood size k

In all k-NN-based methods, selecting k is a challenge. We, therefore, study the sensitivity of these methods to the parameter k. First, we plot AUC over k on the **Pima** dataset in Fig. 14. We can observe that a relatively large-value k performs best, but the methods proposed are not very sensitive to the exact choice of k. Mean shift (MOD/MOD+) is somewhat less sensitive than medoid shift (DOD/DOD+); their performance is almost equal.

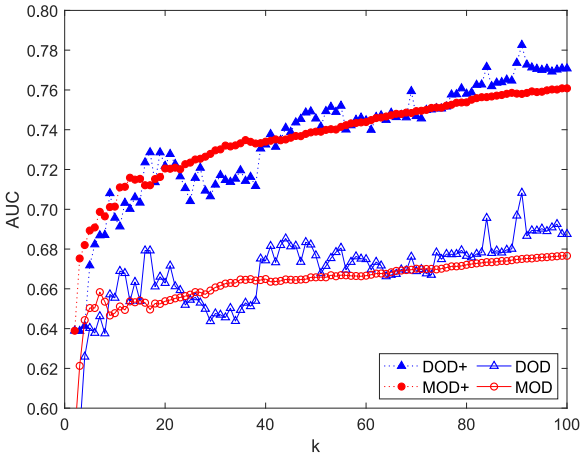

Fig. 14. AUC over k in the **Pima** dataset.

One approach for choosing the value of k is to select a number that is proportional to the data size n. We consider two heuristic rules discussed in [21,54,55] based on logarithmic relationship: $k = 6^{*}\log(n)$ and $k = 2^{*}[5^{*}log(n)/2]+1$, where the brackets refer to rounding operation. The results of applying these two heuristic rules are shown in Tables 9 and 10. We can see that MOD+ remains the best method, with only slight degradation from the optimum (0.84 to 0.81). The corresponding AUC results for the optimal choices of k from Table 4 are: ODIN=0.72, KNN=0.82, MOD=0.82, DOD=0.79, MOD+/DOD+=0.84.

6.5. Computational complexity

All these methods are based on k-NN, which requires $O(N^2)$ calculations. To address this slowness of the brute-force approach, for 2-dimensional datasets, we use the KD-tree technique [43] with all algorithms; for multidimensional datasets, we use the Ball-tree [44] technique with all algorithms. KD-tree works fast with low dimensions but can become inefficient with high dimensions (D>20), and Ball-tree works inefficiently with high dimensions. Alterna-

Table 10
AUC FOR THE REAL-WORLD DATASETS WHEN $k = 2^{*}[5^{*}\mathrm{LOG}(n)/2)+1$.

Dataset (outliers)	KDD. (0.4%)	Stamps (9.1%)	Page. (10.2%)	Pima (34.9%)	Arrh. (45.8%)	Parkinson (75.4%)	AVG
DOD+	0.71	0.88	0.89	0.72	0.73	0.64	0.76
MOD+	0.93	0.90	0.88	0.70	0.73	0.70	0.81
DOD	0.67	0.79	0.79	0.65	0.74	0.70	0.72
MOD	0.90	0.78	0.84	0.65	0.73	0.58	0.75
LOF	0.59	0.53	0.72	0.60	0.73	0.58	0.63
ODIN	0.60	0.58	0.61	0.55	0.69	0.46	0.58
NC	0.59	0.48	0.52	0.46	0.60	0.43	0.51
KNN	0.95	0.88	0.91	0.72	0.74	0.58	0.80
ABOD	0.82	0.80	0.80	0.69	0.72	0.64	0.74

Table 11
RUNNING TIME (**S**) ($k = 30$).

	Data size	100	1000	10,000	100,000
k-NN based	DOD	0.10	0.27	0.99	5.41
	MOD	0.09	0.27	0.93	4.86
	LOF	0.03	0.01	0.11	1.52
	ODIN	0.03	0.01	0.10	1.62
	NC	0.13	1.07	9.19	82.59
	KNN	0.00	0.01	0.10	1.48
	ABOD	0.06	0.62	6.00	58.44
Other	MCD	0.28	0.92	5.82	61.63
	IFOREST	0.20	0.28	0.83	7.25
	OCSVM	0.04	0.03	4.12	493.88
	PCAD	0.14	0.26	0.90	4.94
	QUE	0.38	1.22	745.85	>1000
	MO-GAAL	1.95	2.11	7.82	546.01

tively, faster approximates, such as *NNDES* [45], *Random pair divisive* (RP-div) [46], or Z-curve [47] can become more efficient.

To evaluate algorithm complexity and execution speed, the above-mentioned algorithms are implemented in Python 3.7, using a PC with an Intel Core i7 CPU, 16 GB RAM, and a clock frequency of 2.3 GHz.[3] Table 11 shows the average AUC improvement and the average extra computing time evaluated on the 2-D datasets generated. The proposed method is about 3.5 times slower than ODIN, LOF and KNN, mainly because of the need to calculate *k*-NN 3 times, once per iteration. However, it is faster than other competing detectors. DOD is also slower than MOD because of the need to search for the medoid.

7. Conclusions

Mean shift and medoid shift are proposed for filtering the data before analysis such as clustering and to detect outliers. For the clustering task, our results demonstrate that they improve both *k*-means and random swap when used as preprocessing. The proposed approach outperforms five existing outlier removal methods in this task: LOF, ODIN, NC, IFOREST, and ABOD. The most important property of the proposed approach is that it does not require the number of outliers in advance.

For the outlier detection task, the mean-shift outlier detector (MOD) is slightly more effective than the medoid-shift outlier detector (DOD). Our experiments show the proposed approaches outperform eleven state-of-the-art outlier detectors: LOF, NC, KNN, ODIN, MCD, IFOREST, OCSVM, PCAD, and ABOD. The most important property of the proposed approach is that it is competitive especially when the number of outliers is large. The method is also not limited to numeric data and is can be applied to string data using edit distance.

We also demonstrate that when data contains large numbers of outliers, the model becomes polluted by outliers thus biasing outlier detection. The cumulative effect of outliers can be potentially solved by the fundamentally new concept introduced, the extended reference set, which contains both objects from original data and mean-shift-modified objects.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[3] Our algorithm can be found via http://cs.uef.fi/sipu/soft/MeanShift-OD.py.

References

[1] C.C. Aggawal, Outlier Analysis Second Edition, Springer International Publishing, 2016 v.
[2] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: experiments and analyses, Pattern Recognit 74 (406–421) (2018).
[3] D. Chakraborty, V. Narayanan, A. Ghosh, Integration of deep feature extraction and ensemble learning for outlier detection, Pattern Recognit 89 (161–171) (2019).
[4] D. Carrera, B. Rossi, P. Fragneto, G. Boracchi, Online anomaly detection for long-term ecg monitoring using wearable devices, Pattern Recognit 88 (482–492) (2019).
[5] M. Gupta, J. Gao, C.C. Aggawal, J. Han, Outlier Detection for Temporal Data, Morgan & Claypool Publishers, 2014.
[6] G. Wang, yufei. Chen, X. Zheng, Gaussian field consensus: a robust nonparametric matching method for outlier rejection, Pattern Recognit 74 (305–316) (2018).
[7] Y. Cong, J. Yuan, J. Liu, Abnormal event detection in crowded scenes using sparse representation, Pattern Recognit 46 (7) (2013) 1851–1864.
[8] B. Tu, X. Yang, N. Li, C. Zhou, D. He, Hyperspectral anomaly detection via density peak clustering, Pattern Recognit. Lett. 129 (144–149) (2020).
[9] M. Ribeiro, A.E. Lazzaretti, H.S. Lopes, A study of deep convolutional auto-encoders for anomaly detection in videos, Pattern Recognit. Lett. 105 (13–22) (2018).
[10] S. Li, C. Liu, Y. Yang, Anomaly detection based on maximum a posteriori, Pattern Recognit. Lett. 107 (91–97) (2018).
[11] J.W. Yang, R. Mariescu-Istodor, P. Fränti, Three Rapid Methods for Averaging GPS Segments, Applied Sciences 9 (22) (2019) 4899.
[12] T.V. Pollet, L. van der Meij, To remove or not to remove: the impact of outlier handling on significance testing in testosterone data, Adapt Human Behav Physiol 3 (1) (2017) 43–60.
[13] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, *Int. Conf. on Knowledge Discovery and Data Mining* (KDD) (1996) 226–231.
[14] H.P. Kriegel, P. Kröger, A. Zimek, Outlier detection techniques, 13th Pacific-Asia Conf. Knowledge Discovery Data Mining (2009) 1–73.
[15] J.W. Yang, S. Rahardja, P. Fränti, Outlier detection: how to threshold outlier scores, International Conference on Artificial Intelligence, Information Processing and Cloud Computing, 2019 AIIPCC2019accepted.
[16] P. Fränti, J.W. Yang, Medoid-shift noise removal to improve clustering, Int. Conf. Art. Int. Soft Computing (2018) 604–614.
[17] J.W. Yang, S. Rahardja, P. Fränti, Mean-shift outlier detection, *Int. Conf. Fuzzy Systems and Data Mining* (FSDM) (2018) 208–215.
[18] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, ACM SIGMOD Record 29 (2) (2000) 427–438.
[19] V. Hautamäki, I. Kärkkäinen, P. Fränti, Outlier detection using knearest neighbor graph, *Int. Conf. on Pattern Recognition* (ICPR) (2004) 430–433.
[20] E.M. Knorr, R.T. Ng, Algorithms for mining distance-based outliers in large datasets, in: Int. Conf. Very Large Data Bases, 1998, pp. 392–403.
[21] X. Li, J. Lv, Z. Yi, An efficient representation-based method for boundary point and outlier detection, IEEE Trans. on Neural Networks and Learning Systems 29 (1) (2018) 51–62.
[22] M.M. Breunig, H. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, ACM SIGMOD Int. Conf. on Management of Data 29 (2) (2000) 93–104.
[23] G.O. Campos, A. Zimek, J. Sander, R.J.G.B. Campello, B. Micenkova, E. Schubert, I. Assent, M.E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min Knowl Discov 30 (4) (2016) 891–927.
[24] P.J. Rousseeuw, Least median of squares regression, J. Am Stat Ass (1984) 79–871.
[25] F. Liu, T. Ting, K. Ming, Z.H. Zhou, Isolation-based anomaly detection, *ACM Transactions on Knowledge Discovery from Data* (TKDD) 6 (1) (2012) 3:1–3:39.
[26] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput 13 (7) (2001) 1443–1471.
[27] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, Pattern Recognit 58 (2016) 121–134.
[28] C. Ding, X. He, K-means clustering via principal component analysis, In Proceedings of the twenty-first international conference on Machine learning (2004) 29 ICML '04). ACM-.
[29] M.-.L. Shyu, S.-.C Chen, K. Sarinnapakorn, L.W. Chang, A Novel Anomaly Detection Scheme Based on Principal Component Classifier, ICDM Foundation and New Direction of Data Mining workshop (2003) 172–179.
[30] H. Kriegel, M. Schubert, A. Zimek, Angle-based Outlier Detection in High-dimensional Data, in: the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, 2008, pp. 444–452.
[31] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.
[32] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Analysis and Machine Intelligence 17 (8) (1995) 790–799.
[33] D.-.M. Tsai, J.-.Y. Luo, Mean shift-based defect detection in multicrystalline solar wafer surfaces, IEEE Trans. on Industrial Informatics 7 (1) (2011) 125–135.

[34] Y.A. Sheikh, E.A. Khan, T. Kanade, Mode-seeking by Medoidshifts, IEEE Int. Conf. on Computer Vision, ICCV, 2007.
[35] H.V. Nguyen, V. Gopalkrishnan, Feature extraction for outlier detection in high--dimensional spaces, J Mach Learn Res Proc Track 10 (2010) 66–75.
[36] V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen, P. Fränti, Improving *k*-means by outlier removal, in: *Scand. Conf. on Image Analysis* (SCIA), Lecture Notes of Computer Science, 2005, pp. 978–987.
[37] M. Okade, P.K. Biswas, Mean shift clustering based outlier removal for global motion estimation, 2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, NCVPRIPG, 2013.
[38] E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classification, Biometrics 21 (1965) 768–780.
[39] P. Fränti, Efficiency of random swap clustering, J Big Data 5 (13) (2018) 1–29.
[40] Y. Li, L.P. Maguire, Selecting critical patterns based on local geometrical and statistical information, IEEE Trans. Pattern Analysis Machine Intelligence 33 (6) (2011) 1189–1201.
[41] P. Fränti, S. Sieranoja, K-means properties on six clustering benchmark datasets, Applied Intelligence 48 (12) (2018) 4743–4759.
[42] P. Fränti, M. Rezaei, Q. Zhao, Centroid index: cluster level similarity measure, Pattern Recognit 47 (9) (2014) 3034–3045.
[43] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun ACM 18 (9) (1975) 509–517.
[44] S.M. Omohundro, Five balltree construction algorithms, International Computer Science Institute Technical Report (1989).
[45] W. Dong, C. Moses, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, ACM Int. Conf. on World Wide Web (2011) 577–586.
[46] S. Sieranoja, P. Fränti, Fast random pair divisive construction of kNN graph using generic distance measures, in: Int. Conf. on Big Data and Computing, ICBDC, 2018, pp. 95–98.
[47] S. Sieranoja, P. Fränti, Constructing a high-dimensional kNN-graph using a Z-order curve, ACM Journal of Experimental Algorithmics 23 (1) (2018) 1–21 1.9.
[48] Y. Dong, S.B. Hopkins, J. Li, Quantum entropy scoring for fast robust mean estimation and improved outlier detection, Conference version in NeurIPS (2019).
[49] Y. Liu, et al., Generative Adversarial Active Learning for Unsupervised Outlier Detection, IEEE Trans Knowl Data Eng 32 (8) (2020) 1517–1528.
[50] Y. She, A.B. Owen, Outlier detection using nonconvex penalized regression, J Am Stat Assoc 106 (494) (2012) 626–639.
[51] Y.Q. Ma, S.C. Liu, Q. Z. L., An advanced multiple outlier detection algorithm for 3d similarity datum transformation, Measurement 163 (2020) 107945 2020.
[52] B. Wang, J. Yu, C. Liu, M. Li, B. Zhu, Data snooping algorithm for universal 3D similarity transformation based on generalized EIV model, Measurement 119 (2018) 56–62.
[53] P. Xu, Sign-constrained robust least squares, subjective breakdown point and the effect of weights of observations on robustness, J Geod 79 (1–3) (2005) 146–159.
[54] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun.* ACM 24 (6) (1981) 381–395.
[55] Y. Li, L.P. Maguire, Selecting critical patterns based on local geometrical and statistical information, IEEE Trans. Pattern Anal. Mach. Intell. 33 (6) (2011) 1189–1201.

Jiawei Yang received the B.CN. degree in Electronic Engineering from Beihang University, China in 2013, the M.*Sc.* and Ph.D. degrees in Computer Science from University of Eastern Finland, Finland, in 2019 and 2020, respectively.

Susanto Rahardja (F'11) received the B.Eng. degree from National University of Singapore in 1991, the M.Eng. and Ph.D. degrees all in Electronic Engineering from Nanyang Technological University, Singapore, in 1993 and 1997, respectively. He is currently a Chair Professor at the Northwestern Polytechnical University (NPU) under the Thousand Talent Plan of People's Republic of China. His-research interests are in multimedia, signal processing, wireless communications, discrete transforms and signal processing algorithms, implementation and optimization. Dr. Rahardja was the recipients of numerous awards, including the IEE Hartree Premium Award, the Tan Kah Kee Young Inventors' Open Category Gold award, the Singapore National Technology Award, A*STAR Most Inspiring Mentor Award, Finalist of the 2010 World Technology & Summit Award, the Nokia Foundation Visiting Professor Award and the ACM Recognition of Service Award.

Pasi Fränti received the MSc and PhD degrees in science from the University of Turku, 1991 and 1994. He has worked as a professor in the University of Eastern Finland since 2000. During the preparation of this manuscript, he was a visiting professor in the Shenzhen technology university, China. He has published 94 journals and 173 peer review conference papers. His-research interests include clustering algorithms, location based services, pattern recognition, machine learning and data mining.