Mean-shift outlier detection

Jiawei Yang

Master's Thesis

UNIVERSITY OF
EASTERN FINLAND

School of Computing

Computer Science

22.11. 2018

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing
Computer Science

Jiawei Yang: Mean-shift outlier detection
Master's Thesis, 45 p., no appendices
Supervisor of the Master's Thesis: Professor Pasi Fränti
22.11. 2018

Abstract: Most outlier detection techniques become ineffective when the number of outliers is large. We propose a mean-shift outlier detection method, which also works in case of very noisy datasets. The method processes every point by calculating its *k*-nearest neighbors (*k*-NN) and then replacing the point by the mean (or median) of its neighborhood. The process is repeated a few times. This mean-shift process can be applied in two ways: (1) calculating outlier score based on how much the point is moved to detect outliers, (2) pre-processing prior to data analysis such as clustering. We demonstrate that the method works both with numerical data using Euclidean distance and with text data using Edit distance. The method outperforms all comparative methods, and it does not need a priori knowledge of the noise level.

Keywords: mean-shift, medoid-shift, noise removal, outlier detection, data pre-processing, clustering.

CR Categories (ACM Computing Classification System, 2012 version):

- **Computing methodologies~Cluster analysis**
- Information systems~Clustering

Acknowledgment

I'm first grateful to my thesis advisor Professor Pasi Fränti. He always fast replies my emails whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my work and steered me in the right direction whenever he thought I needed it. He taught me how to write scientific papers patiently and how to do research wisely. Without his guidance and patiently supervising, I could not have done this thesis and completed the master studies. Big appreciation.

I would then like to thank Professor Susanto Rahardja, who provided chances to visit Northwestern Polytechnical University, where I did most of my thesis work including a few other papers. He activated my determinations to do research and extended my view to write high-quality scientific papers. Without his kind support and guidance, I cannot reach this achievement. Big thanks.

I would like to express my gratitude and great thank towards UEF teachers, who gave me the knowledge and help me with problems. I would also like to acknowledge Sami Sieranoja, who did big help in my thesis. I also want to show my gratefulness to Oili Kohonen, who has been helping me deal with kinds of problems whenever and whatever I sought for help. It was such a valuable and unforgettable experience for studying in UEF, Joensuu, meeting people from over the world, watching the northern light … so nice it was. Big hug UEF.

Finally, I must express my very profound gratitude to my family and my girl friend Yang, providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Big kiss.

Abbreviations

CI	Centroid index, a cluster-level measure of clustering quality
k-NN	K-nearest neighbors
LOF	Local outlier factor, outlier detection algorithm
ODIN	Outlier detection with indegree, outlier detection algorithm
MCD	Minimum covariance determinant, outlier detection algorithm
NC	The reverse unreachability, outlier detection algorithm
KNN	Distance to kth nearest neighbors, outlier detection algorithm
TONMF	Text outliers using nonnegative matrix factorization, outlier detection algorithm
MOD	Mean-shift outlier detection, outlier detection algorithm
DOD	Medoid-shift outlier detection, outlier detection algorithm
SSE	Sum of squared errors
NMI	Normalized mutual information
AUC	Area under the curve
ROC	Receiver operating characteristic
RS	Random swap clustering, clustering methods
K-means	K-means clustering, clustering methods
SD	Standard deviation
MAD	Median absolute deviation
IQR	Interquartile range

Notations

N	Number of objects in a data set
n_i	Number of objects in the partition p_i
$d\ (x, y)$	Euclidean distance between object x and y
K	Number of clusters in a dataset
D	Number of attributes in a data set
X	Set of N data objects $\{x_1, x_2, \ldots, x_N\}$
C	Set of K cluster centroids $\{c_1, c_2, \ldots, c_K\}$
P	Set of N cluster indices

Contents

1 Introduction 7

2 Outlier detection overview 10

2.1 Problem definition 10
2.2 Distance measure 10

3 Existing algorithms and their limitations 12

3.1 Outlier detection algorithms for numeric datasets 12
3.1.1 KNN 12
3.1.2 LOF 13
3.1.3 ODIN 14
3.1.4 MCD 15
3.1.5 NC 16
3.2 Outlier detection algorithms for string datasets 18

4 Mean-shift outlier filtering and detection 19

4.1 Mean-shift process 19
4.2 Mean or median 19
4.3 Mean-shift outlier filtering to improve clustering 20
4.4 Clustering algorithms 21
4.4.1 k-means 21
4.4.2 Random swap 23
4.5 Outlier detection 25
4.5.1 Mean-shift outlier detection 25
4.5.2 Edit distance 26

5 Experiments 27

5.1 Datasets 27
5.1.1 Two- dimensional dataset and noise model 27
5.1.2 Multi-dimensional dataset 28
5.1.3 String dataset and distance methods 28
5.2 Baseline outlier detection algorithm 28
5.3 Tested clustering algorithms 29
5.4 Experimental setup 29
5.5 Evaluating measurement 29
5.5.1 Clustering: Centroid index (CI) 30
5.5.2 Clustering: Normalized Mutual Information (NMI) 31
5.5.3 Outlier detection: ROC AUC 32

6 Experiment results and discussion 33

6.1 Clustering results 33
6.2 Outlier detection results 39
6.3 Effect of *k* in mean-shift process 40
6.4 Effect of mean-shift *iteration*s 41
6.5 Running time 42

7 Conclusions 43

References 44

1 Introduction

Outliers are objects that deviate from the typical data. They can represent significant information, which is wanted to be detected such as fraud detection, public health, and network intrusion [1], but they can also affect statistical conclusions based on significance tests [2]. Outliers can also be noise objects which might harm the data analysis process. In any case, we want to detect the outliers.

Outlier detection consists of two main steps: scoring and thresholding. The outlier score of a point is calculated based on a reference set. The outlier detection approaches fall roughly into global and local outlier models based on how the reference set is constructed [3]. The global methods use all other data object as reference set, and the local methods only a small subset of nearby points as reference set.

The obtained scores are then sorted, and data points with the highest scores are labeled as outliers, and the rest are labeled as inliers. The decision of how many outliers are detected can be made in two ways. The top-*N* approach uses a priori knowledge of the noise level (%) of the data and marks exactly the given number of highest scoring points as outliers. Another approach is to derive a threshold value from the statistical distribution of the scores with technique like *standard deviation* (SD), *median absolute deviation* (MAD), *interquartile range* (IQR). In [4], it proposed a two-stage threshold technique, which is better than the SD, MAD, IQR.

In clustering, detection of outliers can be considered as a pre-processing step. The idea is to clean the data from noisy points that might affect the quality of the clustering. Another approach is to perform the clustering first, and then label those points as outliers, who did not fit into any cluster, see Fig. 1. DBSCAN is an example of this kind of approach [5]. However, this is a *chicken-or-egg problem*: removing outliers can potentially improve the clustering, but on the other hand, performing clustering first would also make it easier to detect the outliers.

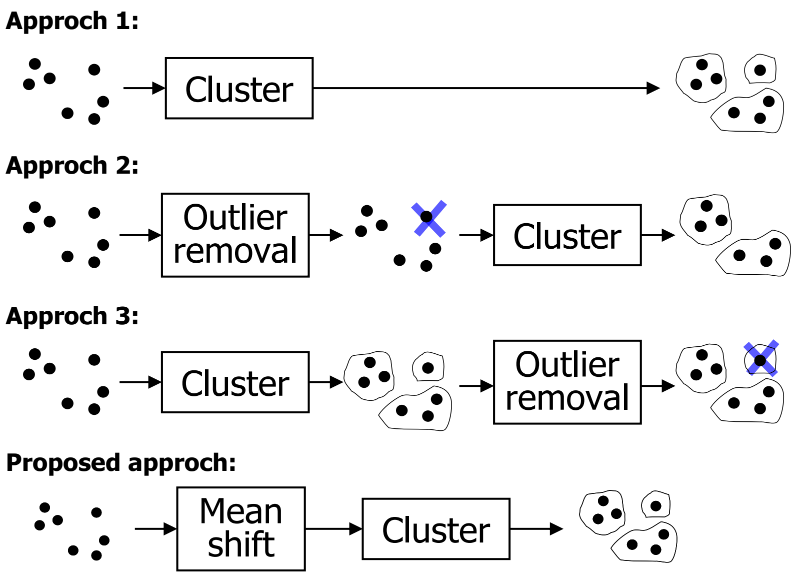


Fig. 1. Example of the iterative process of the medoid-shift [4].

The obtained scores are then sorted, and data points with the highest scores are labeled as outliers, and the rest are labeled as *inliers*. The decision of how many outliers are detected can be made in two ways. The *top-N* approach uses a priori knowledge of the *noise level* (%) of the data and marks exactly the given number of highest scoring points as outliers. Another approach is to derive a *threshold value* from the statistical distribution of the scores. However, this approach has been rarely used in clustering context.

In [4], we have proposed a noise removal pre-processing method before clustering. The key idea is to apply mean-shift or medoid-shift for several iterations (not to converge) as pre-processing. First, we find *k-nearest neighbors* (*k*-NN) for every data point. We replace the original point by the mean value (or medoid) of its neighbors. The process is then iterated a few times (experiment results show optimal is 3 iterations, see Section 6.4). This iterative process is demonstrated in Fig. 2. The method was then applied for outlier detection in [6]. However, after the mean-shift process as in [4], we then calculate how much distance it was moved, i.e., the distance between the original location and its shifted location. This distance is the outlier score. Example of the outlier scores is shown in Fig. 3.

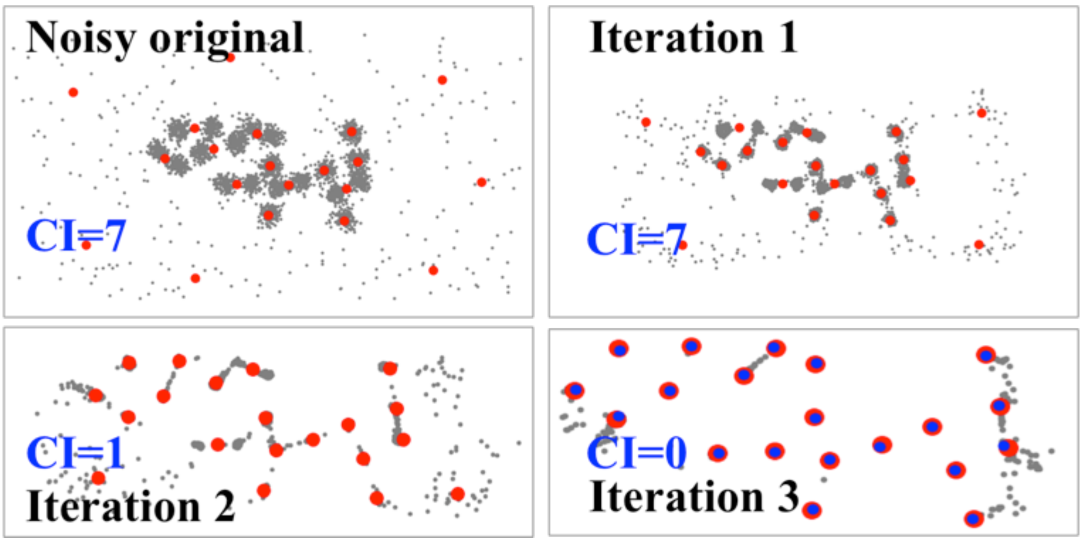


Fig. 2. Random swap clustering results in each mean-shift iteration on 8% noisy A1 dataset. The red points are predicted clusters and blue point are ground truth clusters.

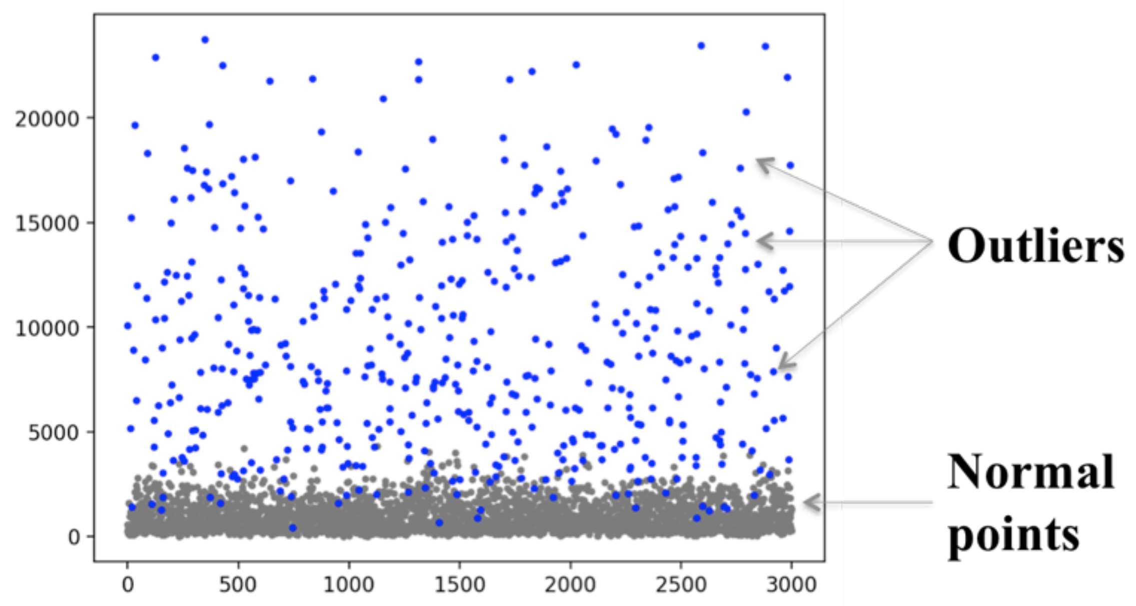


Fig. 3. Outlier scores (y-axis) for the A1 dataset with 16% noise level: normal data points (gray) and noise points (blue). The results are for the proposed MOD method (using mean).

We show that the proposed outlier detection method can be applied both to numeric data using Euclidean distance and to text data using Edit distance. Text data is challenging because of the high dimensional nature (large size of the string) but also because of the sparseness of the distance measure. Furthermore, many words in a document may be topically irrelevant to the context of the document and add to the noise in the distance computations.

2 Outlier detection overview

2.1 Problem definition

Given only a data set, the outlier detection problem can be seen as a binary classification problem: *inner* or *outlier*. A toy example is shown in Fig. 4.

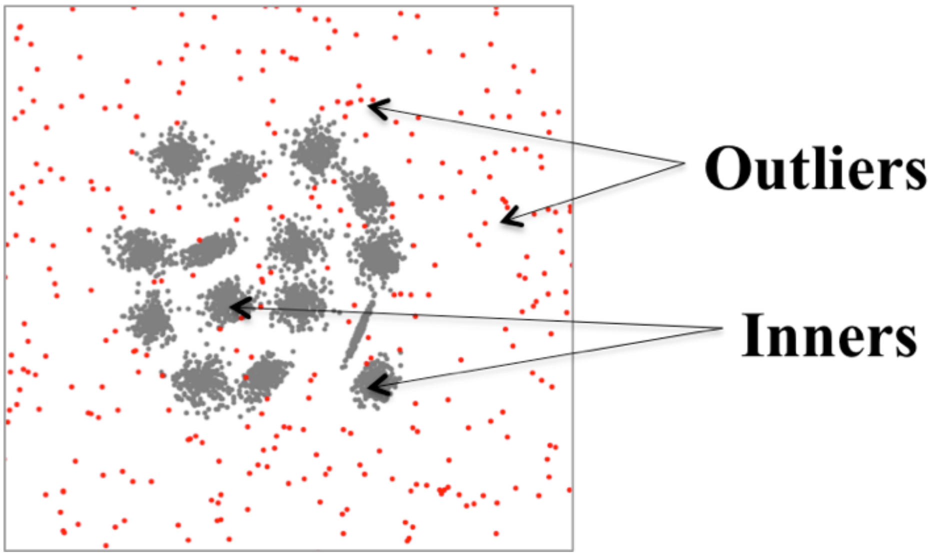


Fig. 4. An example of the outlier (red) and inner (gray)

Normally definition of an outlier relies on applications and assumptions of used models. Some definitions are general enough to handle various data type and methods, for example, an observation deviates so much from others to arise suspicions of different mechanisms in [7], an observation deviates markedly from norm [8], and an observation is inconsistent with the rest in [9].

2.2 Distance measure

Distance function gives the similarity between objects. For a numeric dataset, *Euclidean distance* is the most used. For a string dataset, the most used is *Levenshtein distance*, which belongs to the class of *Edit distance.*

For *D*-dimensional points in a Euclidean space, an obvious choice for a distance function is the Euclidean distance between given two points x_1 and x_2:

$$\mathrm{d}(x_1, x_2) = \sqrt{\sum_{i=1}^{D} \left(x_1^i - x_2^i\right)^2} = \|x_1 - x_2\| \qquad (1)$$

Euclidean distance is commonly used when the attributes of the data objects are coordinates, color components or other numeric types.

Edit distance is the number of the *edit operations* needed to transform one string into the other. A single edit operation may be the insertion, deletion, or substitution of a character. The precise definition of the edit distance varies. Typically, some operations are excluded, and the cost of the operations may vary.

Levenshtein distance is the most used Edit distance. It allows insertions, deletions or substitutions, each having unit cost. For example, the Levenshtein distance between “come” and “coffee” is 3:

1. co**m**e → co**e**e (substitute ‘m’ by ‘e’)
2. coee → co**f**ee (insert ‘f’)
3. cofee → co**f**fee (insert ‘f’)

Other popular distance includes the *longest common subsequence distance* (LCS), which only allows insertion and deletion operations, and *Hamming distance* where only substitutions are allowed. The LCS distance between "Beijing" and "Beef" is 7: substitutions are not allowed, so any characters not shared by the two strings must be deleted first. The Hamming distance for these strings is undefined since it requires the strings to have the same length.

In this thesis, the Euclidean distance is used in numeric datasets, and Levenshtein distance is used in string dataset.

3 Existing algorithms and their limitations

Outlier detection techniques fall into two categories according to the input type: numeric type or text string type. The most used technique is to calculate an outlier score for each object and then extract the top-*N* ranked as outliers, or threshold the scores to get outliers.

For numeric datasets, there are many methods to detect outliers. The widely used methods are based on *k*-nearest neighbors (*k*-NN). For string dataset, there are few methods, and most of them have been generalized from neighbors based method. They are usually based on Euclidean distance after converting strings to numbers, and few works with Edit distance.

3.1 Outlier detection algorithms for numeric datasets

3.1.1 KNN

KNN is a distance-based approach [10,11], based on the assumption that inner data objects have a dense neighborhood whereas outliers lie far apart from their neighbors. It calculates the *k*-nearest neighbors (*k*-NN) and uses the distance to the kth neighbor as an outlier score. Points with large distance are considered as outliers. However, distance-based outlier detection models have problems if the data has areas of varying densities. Fig. 5 shows the example, points in C_1 have bigger *k*-NN distance than the point O_2, but O_2 is outlier. In general, outlier score of KNN is given by equation (2).

$$\text{KNN outlier Score } (x_i) = \text{d } (x_i, \text{ k}^{\text{th}}\text{-nearest neighbor } (x_i)) \quad (2)$$

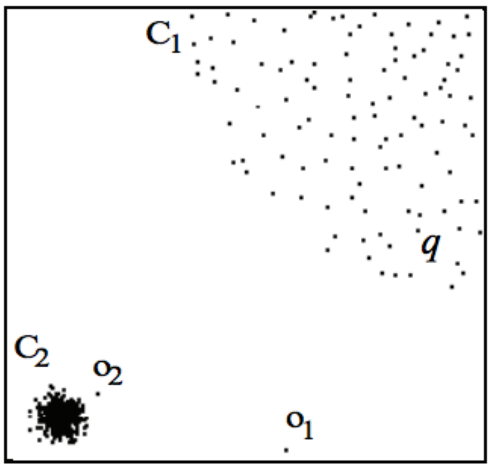


Fig. 5. Dataset varying in densities[1].

[1] https://archive.siam.org/meetings/sdm10/tutorial3.pdf

3.1.2 LOF

Local outlier factor (LOF) [12] is a density-based approach, which analyses the neighborhood of the points. It calculates the local density of the neighborhood. Points that have a lower density than their neighbors are more likely to be outliers. LOF is the best method among those compared in [13]. However, it will be problematic if clusters of different densities are not separated clearly, see Fig. 6. Point *p* will have a higher LOF than points *q* or *r* which is counter intuitive

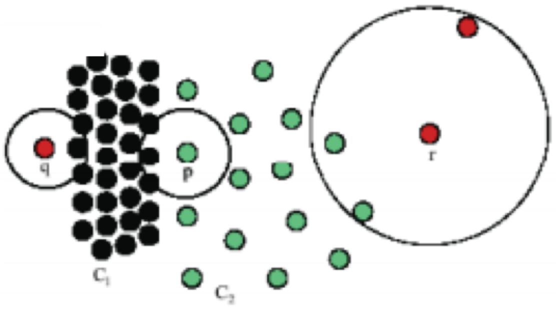


Fig. 6. Dataset varying in densities[2].

In practice, the local density is obtained from the *k*-nearest neighbors. The LOF score of observation is equal to the ratio of the average local density of his *k*-nearest neighbors, and its local density: a normal instance is expected to have a local density similar to that of its neighbors, while abnormal data are expected to have much smaller local density. This strategy is defined in equation (3) and illustrated in Fig. 7.

$$\text{LOF outlier score}(xi) = LOF_k \ (x_i) \ = \frac{-b\sum_{x_j \in N_{k-distance(x_i)}} \frac{lrd_k(x_j)}{lrd_k(x_i)}}{|N_{k-distance(x_i)}|} \tag{3}$$

$$lrd_k = 1/(\frac{\sum_{x_j c N_{k-distance(x_i)}} reach-dist_k(x_i,x_j)}{\left|N_{k-distance(x_i)}\right|}) \tag{4}$$

$$reach\text{-}dist_k(x_i, x_j) = \max \{k\text{-}distance(x_j), \mathrm{d}(x_i, x_j)\} \tag{5}$$

$$N_{k-distance(x_i)}(x_i) = \{x_l \ \in X\backslash\{x_i\} | \ \mathrm{d}(x_i, x_l) \ \leq k - distance(x_i) \} \tag{6}$$

where $k - \text{distance}(\mathrm{x_i})$ is defined as the Euclidean distance $\mathrm{d(x_i, x_j)}$ between $\mathrm{x_i}$ and an object $\mathrm{x_j} \in \mathrm{X}$ such that:

(i) For at least *k* objects $\mathrm{x_l} \in \mathrm{X} \setminus \{ \mathrm{x_i} \}$, it holds that $\mathrm{d(x_i, x_l))} \leq \mathrm{d(x_i, x_j)}$;

(ii) For at most *k*-1 objects $\mathrm{x_l} \in \mathrm{X} \setminus \{ \mathrm{x_i}\}$, it holds that $\mathrm{d(x_i, x_l)} < \mathrm{d(x_i, x_j)}$.

[2] https://archive.siam.org/meetings/sdm10/tutorial3.pdf

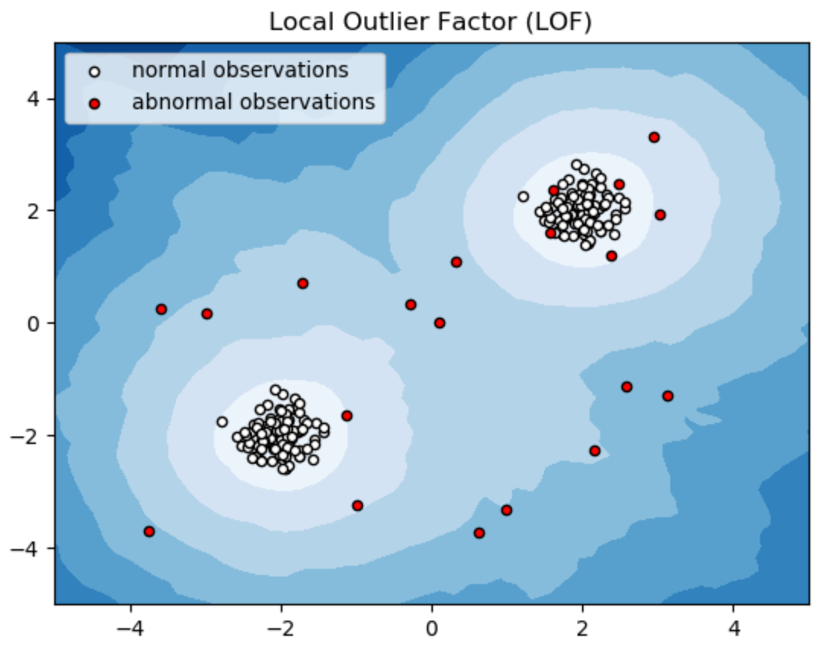


Fig. 7. LOF toy example[3].

3.1.3 ODIN

ODIN [11] is a graph-based approach [11, 14] analyzing the indegree of the nodes in *k*-NN graph. For a given point, it calculates the amount how many other points consider it as their *k*-nearest neighbor. The smaller the value, the more likely the point is an outlier. However, it has problems when the distance inside each cluster is varying. Also, since the outlier scores are integers, thresholding can be a problem if objects have same outlier scores. The outlier score is defined by equation (7) and illustrated in Fig. 8.

$$\text{ODIN Outlier score } (xi) = \text{indegree of vertex } \mathrm{x_i} \text{ for a given KNN graph G} \qquad (7)$$

[3] http://scikit-learn.org/stable/modules/outlier_detection.html

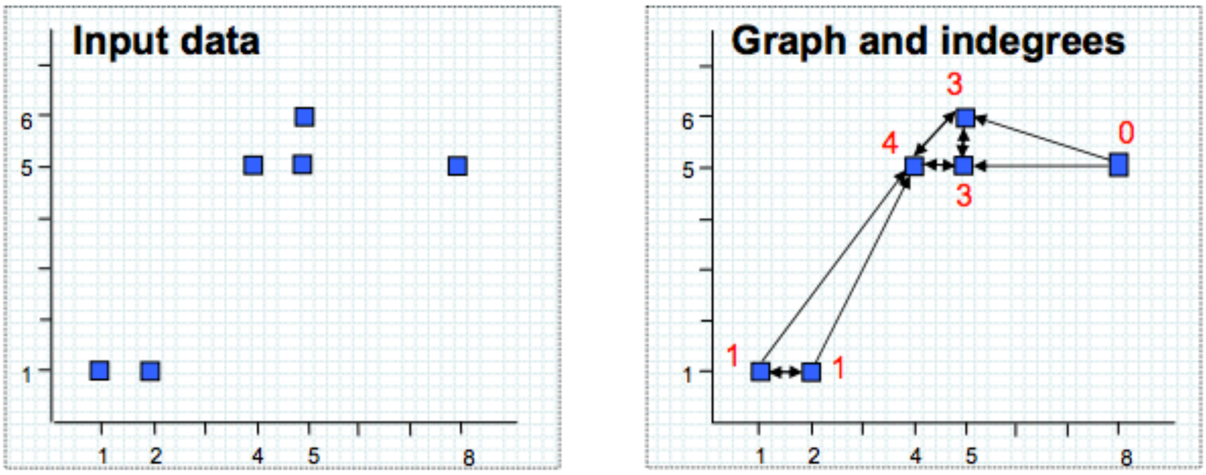


Fig. 8. ODIN toy example (k=2)[4].

3.1.4 MCD

Minimum covariance determinant (MCD) [15, 16] is based on a statistical test with the assumption that the inner data objects follow a (known) distribution and occur in a high probability region of this model. Outliers are expected to deviate strongly from this distribution. However, this method has high computing time to obtain the distribution parameters. The results are also different even for the same dataset when re-run the second time as it can obtain different parameter values each time.

For instance, assuming that the inlier data are Gaussian distributed, it estimates the inlier location and covariance robustly compared to Mahalanobis distances (i.e., without being influenced by outliers, see Fig. 9 (MLE dist)). The robust distances obtained from this estimate are used to derive a measure of outlyingness. This strategy is illustrated in Fig. 9 (robust dist).

$$\text{MCD Outlier score}(X) = \text{RD}(X) = \sqrt{(X - \hat{\mu}_{MCD})^{t}\hat{\Sigma}_{MCD}^{-1} - (X - \hat{\mu}_{MCD})} \quad (8)$$

Where, $\hat{\mu}_{MCD}$ is the MCD estimate of location, and $\hat{\boldsymbol{\Sigma}}_{MCD}$ the MCD covariance estimate.

[4] http://cs.uef.fi/pages/franti/cluster/Clustering-Part7-Outliers.pdf

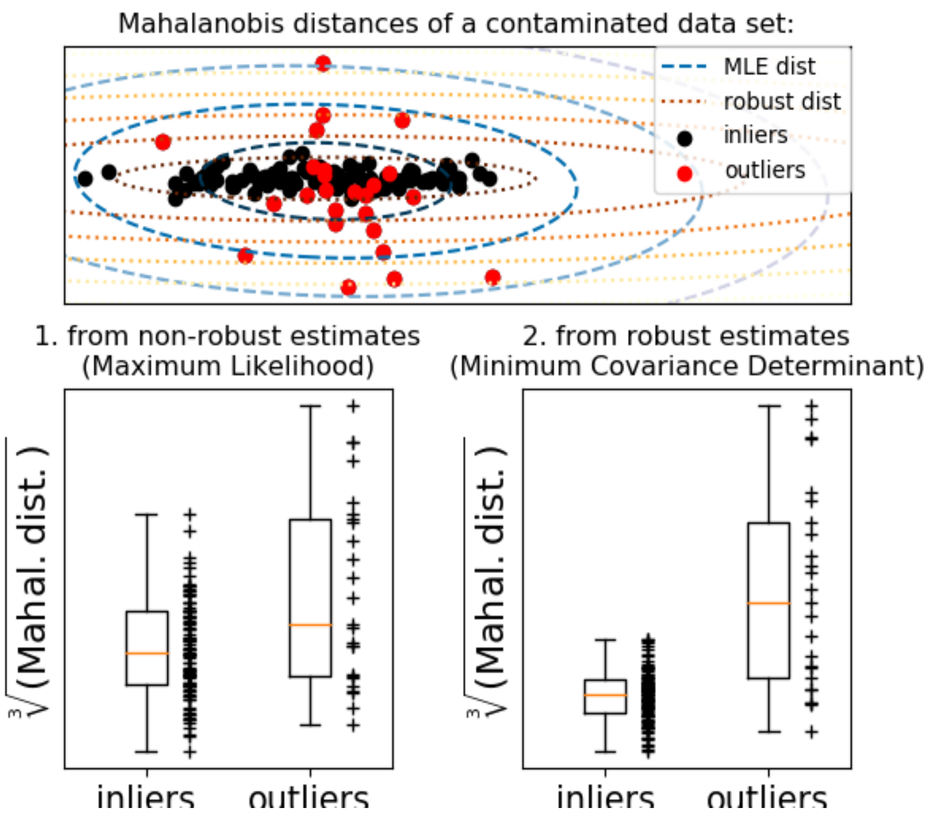


Fig. 9. MCD toy example[5].

3.1.5 NC

NC [17] is a math representation approach based on math optimization, in which points X are represented by a matrix W in equation (10), where x_j belongs to *k* nearest neighborhood of x_i. Fig. 10 shows the relationship between matrix W and location of a sample point P and its *k* nearest neighborhood. For each point, the amount of negative values in column of matrix W is used as an outlier score. The bigger the value, the more likely to be an outlier. However, it will fail if the hull constructed by *k* nearest neighborhood is non-convex hull, and also since outlier scores are integers, thresholding is a big problem when objects have same outlier scores. This outlier score is given by equation (9), and an example is shown in Fig. 11. the

$$\text{NC Outlier score}(x_i) = \text{ The amount of negative values in } i^{\text{th}} \text{ column of W in (10)} \qquad (9)$$

$$min \sum_{i=1}^{n} |x_i - \sum_j W_{ij}\, x_j|^2 , s.t., \sum_j W_{ij} = 1 \qquad (10)$$

[5] http://scikit-learn.org/stable/modules/outlier_detection.html

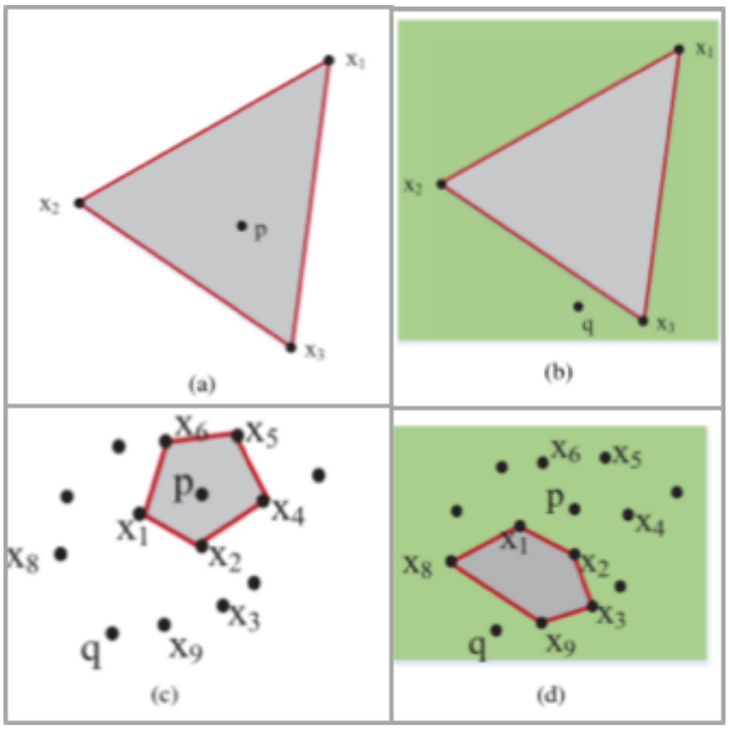
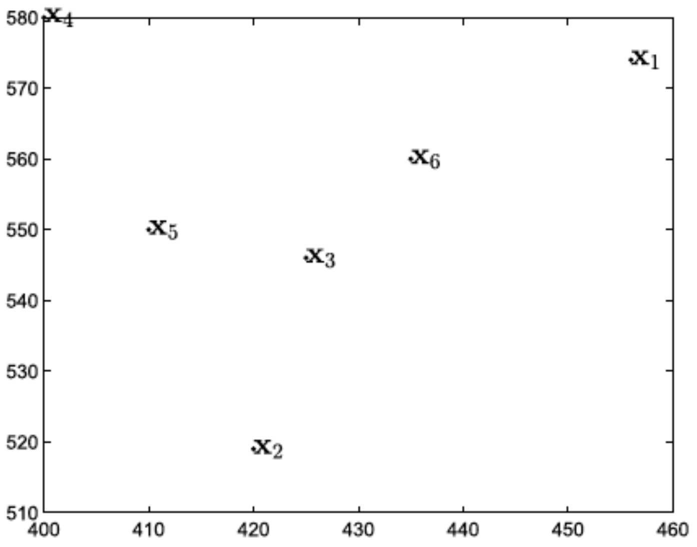


Fig. 10. Given a point *p* and its 3 nearest neighbors, we can get matrix W via equation (10). Clearly, in (a) and (b), it has the following: 1) *p* is within of the triangle if 0< w_i <1. 2) *p* lies on the boundary of the triangle if any w_i = 0. 3) *q* lies outside the triangle if any w_i <0 or w_i >1. Similarly, the conclusion holds for point *p* and *q* in (c) and (d), represented by their *k*-NN. [17].



$$W = \begin{pmatrix} 0 & -0.4024 & 0.1527 & -0.2802 & -0.1176 & 0.4732 \\ -0.5400 & 0 & 0.3718 & -1.0635 & 0.3327 & 0.1147 \\ 0.5400 & 0.9773 & 0 & 0.2748 & 0.2382 & 0.2035 \\ -0.2324 & -0.6584 & 0.0642 & 0 & 0.4315 & 0.1001 \\ -0.3725 & 0.7925 & 0.2156 & 1.6576 & 0 & 0.1084 \\ 1.6048 & 0.2909 & 0.1957 & 0.4112 & 0.1152 & 0 \end{pmatrix}$$

Fig. 11. NC toy example [17]: sample points (up) and representation matrix w. 1) Use equation (1) for points in Fig 11 (up) to get W shown in Fig 11 (down). 2) Outlier score for each point is the negative value amount of each column: outlier score for {x1, x2, x3, x4, x5, x6} is {3, 2, 0, 2, 1, 0} respectively. 3) the bigger the outlier score, the more likely to be the outlier. Hence {x1, x2, x4} is more like to be the outlier.

3.2 Outlier detection algorithms for string datasets

Methods like KNN and ODIN can also detect string outliers with Edit distance directly. It is a very challenging task to detect string outliers but it is an indeed high demand in bioinformatics sequence datasets.

4 Mean-shift outlier filtering and detection

In clustering applications, outliers can significantly affect the results; hence it is better to remove the outliers before clustering. Traditional methods treat this task in two steps: detect and remove. However, what amount to remove is a common problem, as it is not possible to know the correct amount of outliers in advance. We propose mean-shift outlier filtering method, which can remove outliers without requiring knowing the amount of outliers at all. After using the mean-shift outlier filtering method, any clustering can be applied to do the clustering task. We call this method mean-shift outlier filtering algorithm, for improving clustering.

Next, we also propose a technique to detect the outliers, which point is an outlier and which is an inner. We call this method mean-shift outlier detection (MOD) and its counter-part using medoid as medoid-shift outlier detection (DOD), for detecting outlier purpose.

4.1 Mean-shift process

The mean-shift process works locally by analyzing the neighborhoods of the points using *k*-nearest neighbors. The method replaces every point by the mean of its *k* neighbors, which forces points to move towards denser areas. The distance of the movement can be evidence of being an outlier; points with greater movement are more likely to be outliers. The mean-shift process is summarized in Algorithm 1.

The idea is closely related to mean-shift filtering used in mean-shift clustering algorithm [18], and image processing [19]. The first one takes pixel value and its coordinates as the feature and transforms each feature towards the mean of its neighbors. It has been used for detecting fingerprint and contamination defects in multicrystalline solar wafers [20]. The idea also resembles the low-pass and median filtering used for image denoising.

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

```
Input:    Dataset $\mathbf{X} \subset R^{d\times n}$, $\boldsymbol{k} \subset R$, $\boldsymbol{I} \in R$
Output:   $\mathbf{X} \subset R^{d\times n}$
Repeat I times:
  For every point $x_i \in \mathbf{X}$:
    Find its k-nearest neighbors kNN($x_i$)
    Calculate the mean $M_i$ of the neighbors kNN($x_i$)
    Replace $x_i$ by the mean $M_i$, $x_i = M_i$
```

4.2 Mean or median

Both mean and median have been used in the mean-shift clustering concept [21, 22]. The benefit of using mean is that it is trivial to calculate for numeric data. Its main disadvantage is that it can cause a blurring effect where very noisy points bias the calculation of the mean of clean points as well. Median is less sensitive to this because the single noisy point in the neighborhood may not affect the calculation of the median at all. Therefore, it has less effect on the clean points. Another benefit of the median is that, when repeated until converging, it can reach root signal [22].

However, with multivariate data, it is not obvious how the median is defined. We use the *medoid*, which is the point in the set that has the minimum total distance to all other points in the set. A disadvantage of medoid is that its calculation can be more time-consuming. A toy example of using mean and median is shown in Fig. 12.

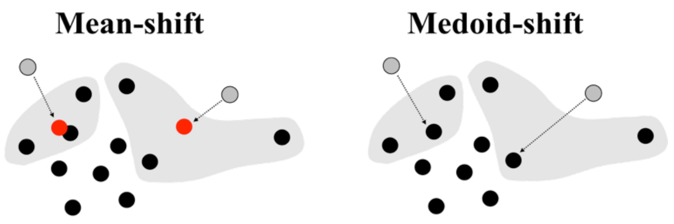


Fig. 12. Effect of the mean (left) and median (right) of two sample points in a noisy neighborhood. [4]

4.3 Mean-shift outlier filtering to improve clustering

As outliers can affect the clustering results, we need to remove it before clustering, Fig. 13 shows the effect of outliers. When adding some outliers, it will bias the clustering result and centroids will be falsely distributed in the noise area.

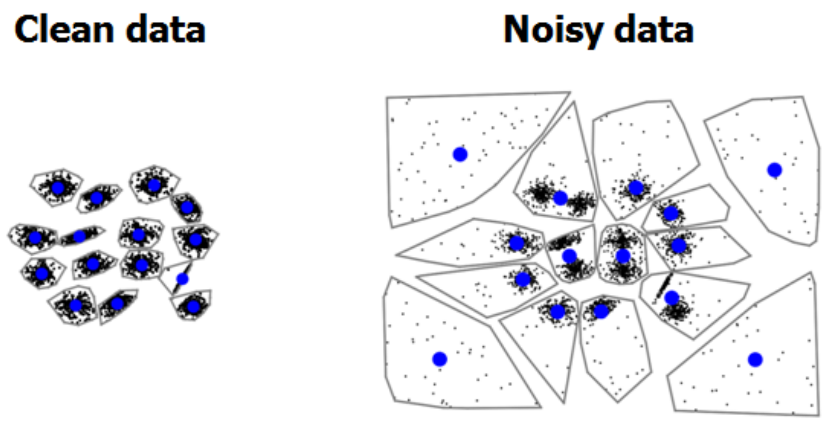


Fig. 13. Example of noisy data and how it affects clustering [4].

This iterative variant is similar to the ORC algorithm [23] where most remote points are iteratively removed in each cluster. The difference is that the outliers are chosen based on the distance to their cluster centroid in the intermediate clustering solution. It is therefore possible that points can be falsely removed if the cluster is not correctly determined. Fig. 14 demonstrates the process. We can see that there is still noise existing while only 3 clusters are left after 90 iterations, but originally it has 5 clusters because ORC falsely removes the inners when the noise cluster is very small, and the distance of noise to its centroid is smaller than the inner cluster with bigger radius, see Fig. 14 (up).

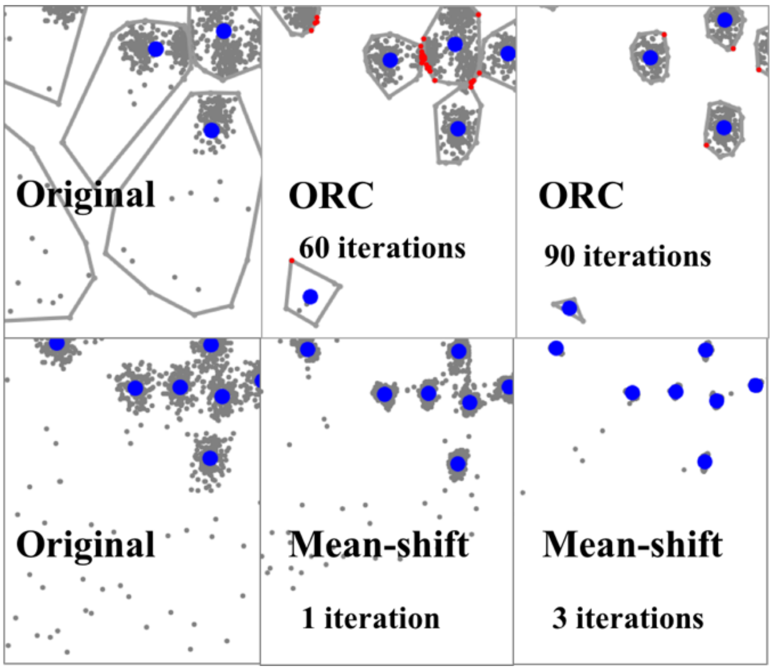


Fig. 14. Example of the iterative process of the ORC [23] (up) and the proposed mean-shift (down) on the part of the A1 dataset with 8% noise level.

The mean-shift process was applied for noise filtering in [4]. The idea is to apply Algorithm 1 as such and modify the data so that the effect of the noisy points is minimized. It is implemented as a separate pre-processing step so that it is independent on the clustering method applied. The process can be iterated several times to have stronger noise filtering effect. The number of iterations is a parameter. Based on our experiments, three iterations are the best choice for typical clustering data. A visualization example of mean-shift noise filtering for clustering in each iteration is shown in Fig 14 (down).

4.4 Clustering algorithms

Two clustering algorithms: *K*-means [24] and *Random swap* (RS) [25] are tested in this thesis. Both minimize sum-of-squared errors. The first one is commonly used but does not always find the correct clustering solution even with clean data. The second does find the correct result with all of the tested datasets in this thesis. *K*-means tends to converge to a local optimum instead of the global one, but Random swap is much more likely to reach a global one or very close to it.

4.4.1 k-means

The *k*-means algorithm is an iterative, centroid-based clustering algorithm. It is very popular clustering algorithm because of its simplicity.

Given a data set $X = \{x_1, x_2, \dots, x_N\}$ and the number of clusters K as parameters, and it returns the centroid locations $C = \{c_1, c_2, \dots, c_K\}$ and the partitioning $P = \{p_1, p_2, \dots, p_N\}$ of the data points as the results (Algorithm 2).

```
Algorithm 2: k-means clustering (K-means (X, P, C))
Input:    Dataset $X \subset R^{d\times n}$, label $P \subset R^{1\times n}$ , centroid $C \subset R^{d\times n}$
Output:   label $P \subset R^{1\times n}$ , centroid $C \subset R^{d\times n}$
REPEAT
  $C_{prev} \leftarrow C$
  FOR i := 1 TO $N$ DO
    $P_i \leftarrow$ FindNearestCentroid ($X_i$, $C$)
  FOR j := 1 TO $k$ DO
    $C_i \leftarrow$ CalculateCentroid ($X$, $P$, $j$)
UNTIL  $C = C_{prev}$
```

The algorithm starts with an initial solution, and iterates until it cannot find any more improvement. It consists of two main steps:

1. Partitioning step
2. Centroid step.

There are several ways to initialize *k*-means, and the most common ways are to pick random data objects as the initial centroids. In the partitioning step, each data object will be labeled according to its nearest centroid. In the centroid step, new centroids will be calculated as the mean over same labeled data objects.

The advantage of *k*-means is its simplicity with providing relatively good results with many data sets. The disadvantage of *k*-means is its high dependency on the initial partitioning, leading to converging to a local optimum instead of the global one, see Fig. 15.

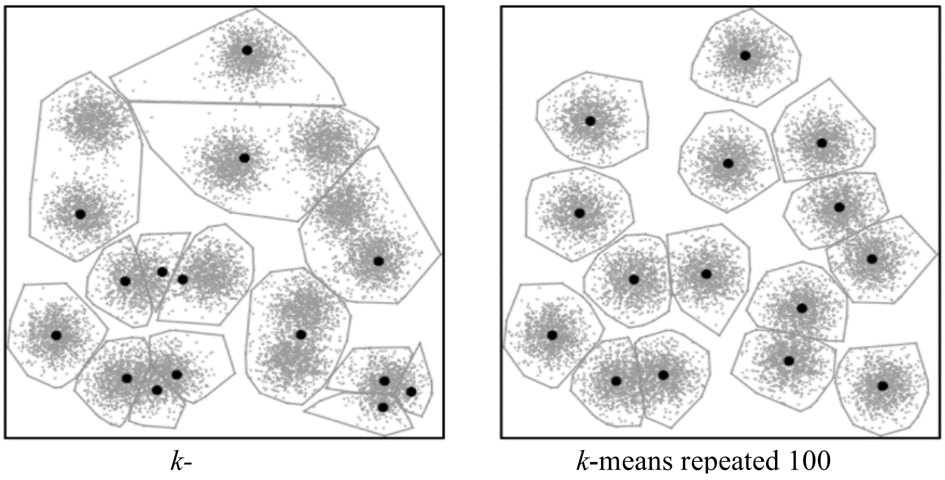


Fig. 15. A single run of *k*-means and repeated *k*-means[6].

[6] http://cs.uef.fi/sipu/pub/MSc_JarkkoPiiroinen.pdf

4.4.2 Random swap

K-means often ends up with a local optimum solution, where two or more than one centroids are located inside the same cluster, or there is no centroid in a real cluster. To move the centroid from *centroid-rich* to a *centroid-poor* area, random swap is a method by randomly guessing the correct centroid solution and accept the improved solutions.

Like k-means, random swap is an iterative, centroid-based clustering algorithm, which is also a simple but very powerful global optimization algorithm.

Given a data set $X = \{x_1, x_2, \dots, x_N\}$ and the number of clusters K as parameters, it returns the centroid locations $C = \{c_1, c_2, \dots, c_K\}$ and the partitioning $P = \{p_1, p_2, \dots, p_N\}$ of the data points as the results (Algorithm 3).

```
Algorithm 3: RandomSwap clustering (RS ($X, P, C$))
Input:    Dataset $X \subset R^{d\times n}$, $I \in R$,  Label $P \subset R^{1\times n}$, Centroid $C \subset R^{d\times n}$
Output:   Label $\mathbf{P} \subset R^{1\times n}$, Centroid $\mathbf{C} \subset R^{d\times n}$
REPEAT $I$ TIMES
    $(C_{nev}, j) \leftarrow$ SwapCentroid $(X, C)$
    $P_{new} \leftarrow$ LocalRepartition $(X, C_{nev}, p, j)$
    $(C_{nev}, P_{new}) \leftarrow$ FindNearestCentroid $(X_i, C)$
    IF $f(C_{nev}, P_{new}) > f(C, P)$ THEN
        $(C, P) \leftarrow (C_{nev}, P_{new})$
RETURN $(C, P)$
```

Like k-means, the algorithm starts with an initial solution (such as using random data points as the initial centroids) and iterates T times. It consists of three main steps:

1. Random swap
2. Local repartition
3. Iteration by k-means.

In the random swap step, a current centroid is randomly selected and replaced by another randomly selected data object with the chance to move centroid from *centroid-rich* to a *centroid-poor* area.

In the *local repartition* step, since centroid is changed after *the* random swap step, all data objects needed to re-label to their new nearest centroids. Note that in [25], it was noted that this step is optional if k-means is applied after the swap.

Finally, *iterations by k*-means are performed. This will fine-tune the result by finding a new local optimum, giving the new configuration of centroids/partitioning. In [26], it was noted that only two iterations of k-means were enough to achieve high-quality clustering.

At this point, the result after the swap has not always improved and it could be worse. Therefore, the objective function is employed to decide whether to accept the new candidate centroids/partitioning.

For example, if sum square error (SSE) is used as the objective function, then the SSE of the new candidate result must be lower than the SSE before the trial swap. If it is not, the result will be discarded.

The advantage of random swap is that implementing it is easy, and it is also efficient: it was shown that the algorithm gives results that are competitive with more complex techniques such as *genetic algorithms* and *tabu search* in [27]. Unlike *k*-means, the algorithm does not depend on the initialization method: centroid swapping enables the algorithm to fix situations where *k*-means would get stuck, such as the highlighted regions in Fig. 16 (current solution). As a result, random swap can detect the clusters when *k*-means struggled with, as seen in Fig. 17.

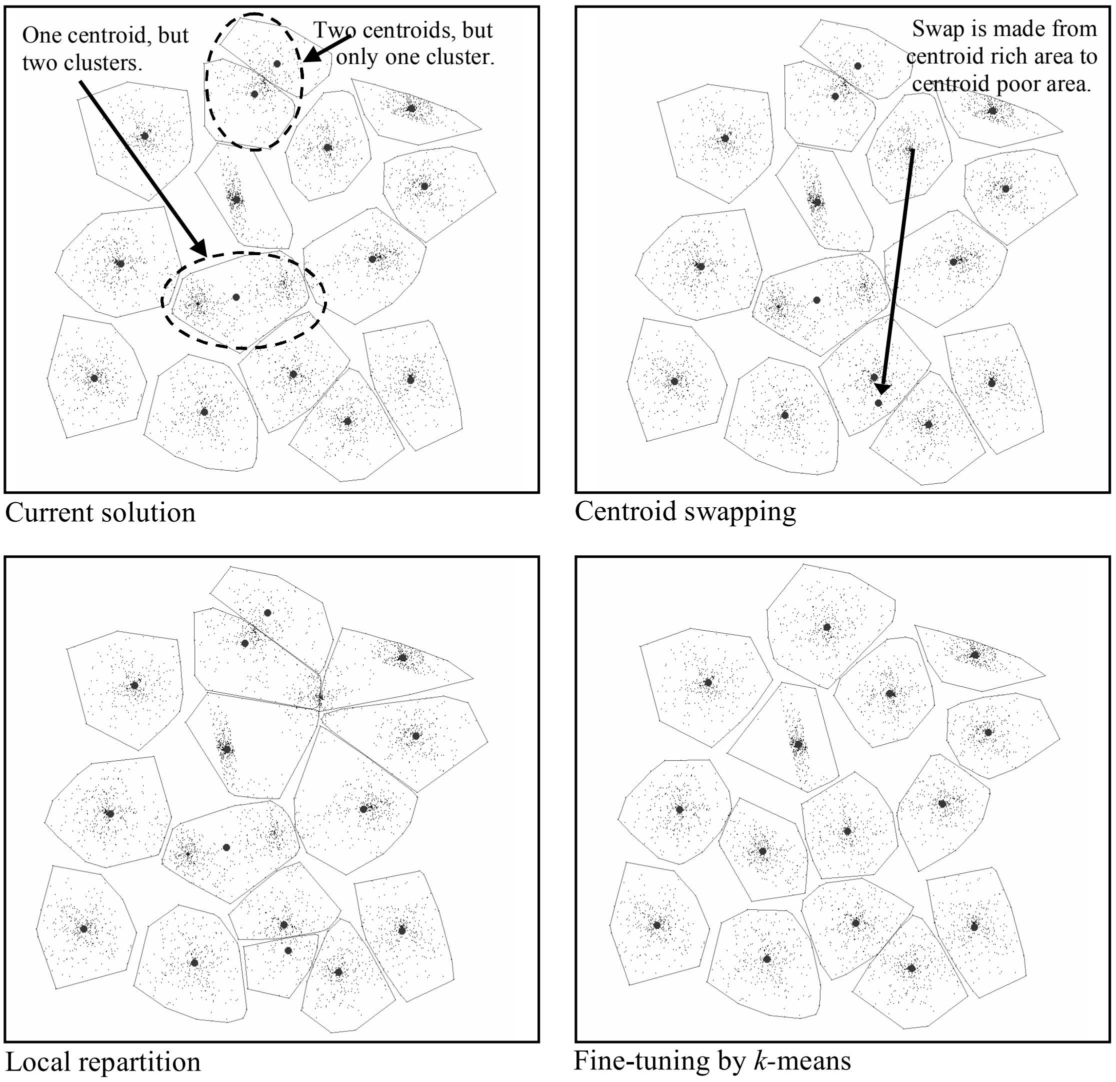

One centroid, but two clusters.
Two centroids, but only one cluster.
Current solution
Swap is made from centroid rich area to centroid poor area.
Centroid swapping
Local repartition
Fine-tuning by *k*-means

Fig. 16. Demonstration of centroid swapping [26].

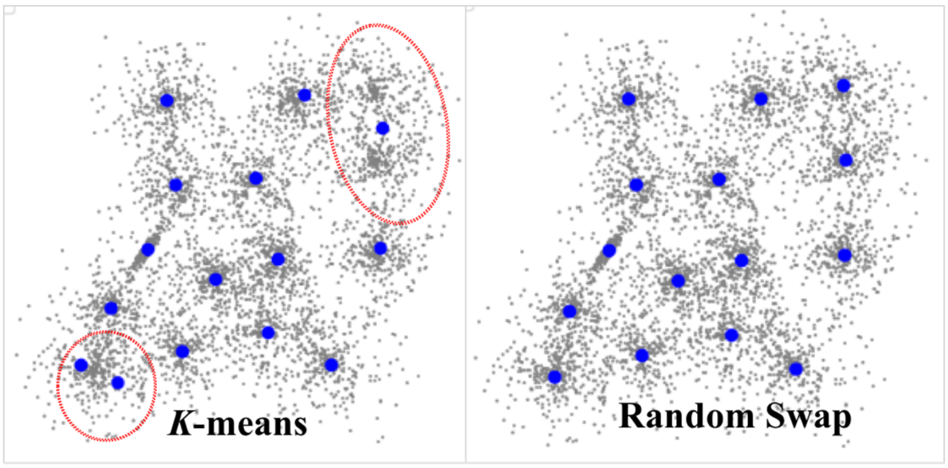


Fig. 17. Random swap can solve S3 dataset perfectly but *K*-mean can not.

4.5 Outlier detection

The idea is next applied to outlier detection. The mean-shift process is basically the same as in the noise filtering, but after the mean-shift process, we calculate the distance before and after shifting as outlier scores.

4.5.1 Mean-shift outlier detection

Mean-shift clustering [21] iterates the mean-shift process until convergence. We apply the same process, but since we are not clustering the data but aim at finding outliers, we use the processing result merely for analysis purpose. In specific, we compare the location of the points before and after the shifting. This difference is the outlier score. The pseudo code of the method is summarized in Algorithm 4. A toy example of the outlier scores is given in Fig. 18.

```
Algorithm 4: Mean-shift outlier detection (MOD (X, k))
Input:    Dataset X ⊂ $R^{d\times n}$, k
Output:   Outlier score D ⊂ $R^{1\times n}$
Step 1:   Use Algorithm 1 to get Y = MS (X, k, 3)
Step 2:   Calculate distance $D_i = |x_i - y_i|$, $x_i \in X$, $y_i \in Y$, $D_i \in D$ and return D
```

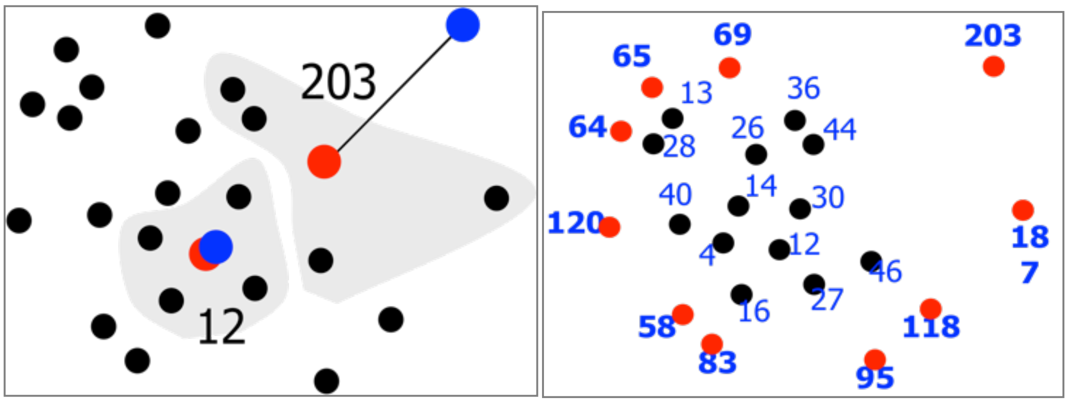


Fig. 18. Example of mean-shift outlier scores based on Euclidean distance for sample points using *k*=4 (left). The red point is the mean of the neighbors of the blue point, and the number shown is the outlier score. All outlier scores are shown (right); the red points are detected outliers with top-10 ranking outlier scores.

4.5.2 Edit distance

Medoid-shift can also be directly applied to strings or single word list using Edit distance. Fig. 19 shows the process and the results. For example, the word "albapax" is shifted to the word "lbanin" and the Edit distance between them is 4, hence outlier score of "albapax" is 4; similar to "lbanin", it is shifted to "lbanin" itself, hence the Edit distance is 0, namely outlier score is 0.

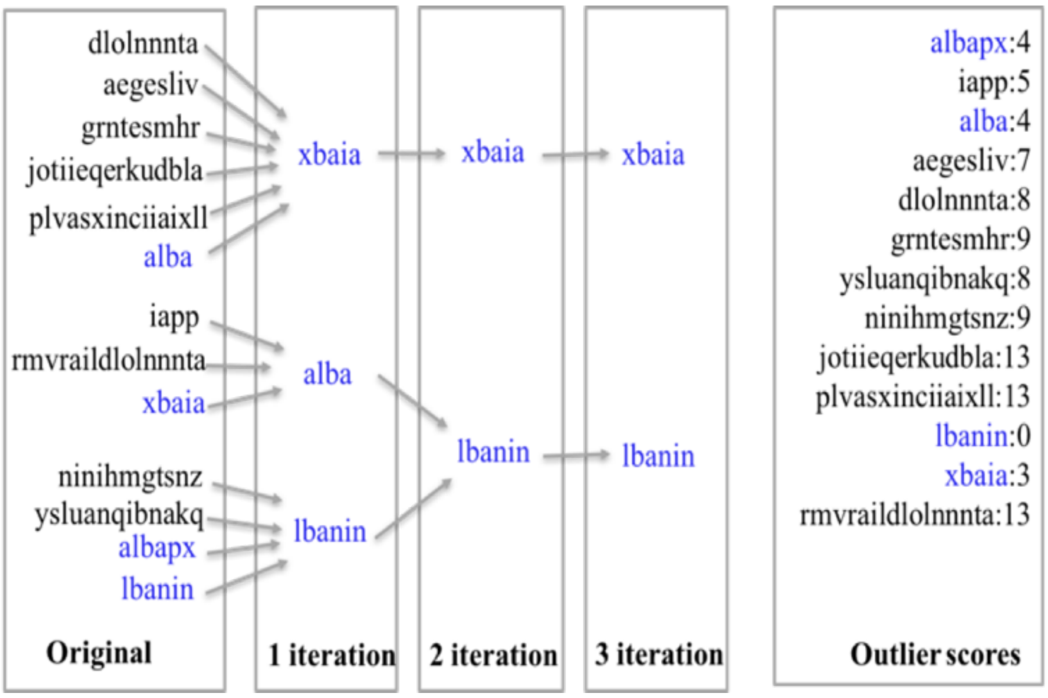


Fig. 19. Example of medoid-shift outlier scores based on edit distance for part of Country string data using *k*=4. The string before and after the arrow is string before and after medoid-shift. The left figure is the medoid-shifting process; the right figure is the outlier score of each string. Black strings are ground truth outliers, and blue strings are normal strings (inners).

5 Experiments

5.1 Datasets

In this thesis, datasets are carefully selected including numeric and string varying in the amount of outlier, attributes, and domains. We use the 15-benchmark datasets in Table I.

TABLE I

DATASETS USED IN THE EXPERIMENTS

Dataset	Ref	Size	Clusters	Dim	Type
S1-S4	[27]	5000	15	2	Numeric
A1-A3	[27]	3000, 5250, 7500	20, 35, 50	2	Numeric
B1-B2	[27]	100,000	100	2	Numeric
Unbalance	[27]	6500	8	2	Numeric
XOR	[16]	2000	4	2	Numeric
KDDCup99	[13]	48113	-	40	Numeric
SpamBase	[13]	4207	-	57	Numeric
Parkinson	[13]	195	-	22	Numeric
Country[7]	-	2666, 3000, 3428, 4000	48	-	String

5.1.1 Two- dimensional dataset and noise model

Eleven 2-d datasets are selected and visualized in Fig. 20. The S sets have varying level of cluster overlap, and A sets varying number of clusters, B sets varying shape, unbalance and XOR datasets having clusters with different densities

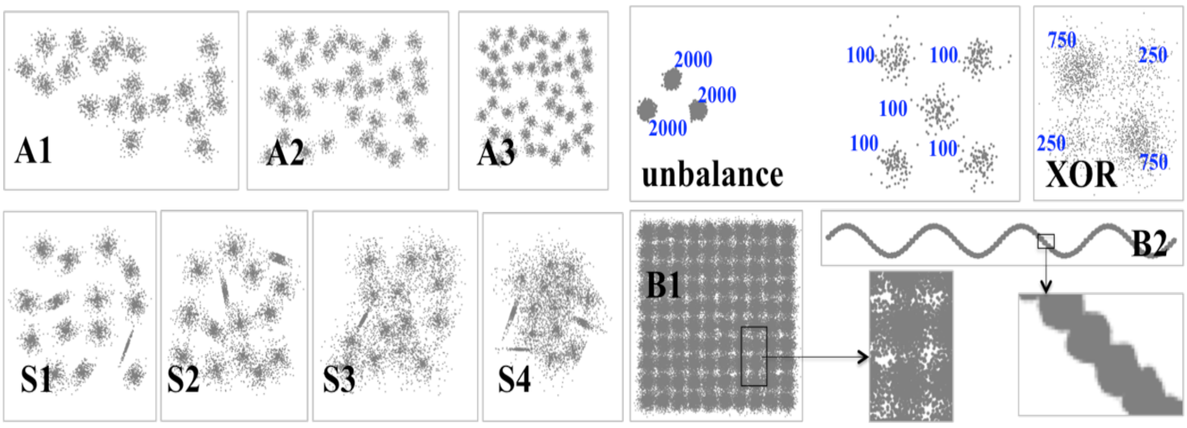


Fig. 20. Eleven 2-D datasets used in the experiments.

Uniformly distributed random noise is added to the A, B, S, unbalance and XOR data sets. Random values are generated in each dimension between [$x_{\text{mean}}-2\cdot range$, $x_{\text{mean}}+2\cdot range$], where x_{mean} is the mean of all data points, and *range* is the maximum distance of any point from the mean: $range = \max(|x_{\max}-x_{\text{mean}}|, |x_{\text{mean}}-x_{\min}|)$. Noisy datasets are shown in Fig. 21.

7 http://cs.uef.fi/sipu/string/countries/

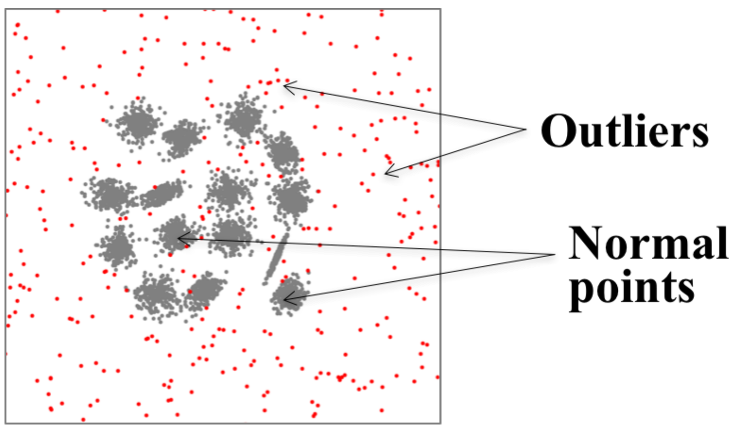


Fig. 21. The noisy S1 dataset with 8% noise. Gray points are normal data, and red points are noise.

5.1.2 Multi-dimensional dataset

KDD Cup99, Parkinson, and SpamBase are real-word semantically meaningful datasets. KDD Cup99 classified network attack as an outlier (0.42% outliers), Parkinson classified the patient as the outlier (39.91% outliers), and SpamBase classified spam email as the outlier (75.38% outliers).

As data pre-processing, duplicates in high dimensional datasets are removed, and each attribute is scaled to the range between 0 and 1 (subtracting mean and divided by standard deviation). Also, categorical attributes are removed from the KDDCup99 dataset.

5.1.3 String dataset and distance methods

Countries datasets contain modified copies of the names of the 48 European countries. The modifications are random insert and delete operations, and the amount of operation is 10%, 20%, 30%, 40% so that the resulting strings can still be identified with some effort. Noise has also been added so that fake strings have been generated so that their length and character distribution resembles those of the country names, but their content is complete nonsense. The task is then to identify which strings are from the original dataset and which are the fake ones (noise).

Edit distance between the strings will be used for Country dataset. This approach allows using existing methods that are based on distances such as ODIN and KNN.

5.2 Baseline outlier detection algorithm

Baseline algorithms are carefully selected from existing algorithms, according to categories, popularity, and performance. We will compare all the algorithms introduced in Section 2, and summarized in Table II.

TABLE II

COMPARED OUTLIER DETECTION ALGORITHMS

Method	**Ref**	**Type**	**Data**	**Publication and year**
KNN	[10]	Distance-based	N/S	ACM SIGMOD, 2000
LOF	[12]	Density-based	N	ACM SIGMOD, 2000
ODIN	[11]	Graph-based	N/S	ICPR, 2004
MCD	[15]	Statistical testing	N	J. A. Stat. Assoc, 1984
NC	[16]	Representation	N	IEEE-TNNLS, 2018
DOD	[4, 6]	Shifting-based	N/S	-
MOD	[4, 6]	Shifting-based	N/S	-

* N = numeric data, S= string data.

5.3 Tested clustering algorithms

We will test with k-means and random swap clustering algorithms, introduced in Section 4.4. Both minimize sum-of-squared errors. The first one is commonly used but does not always find the correct clustering solution even with the clean data. The second does find the correct result with all of these datasets; in this case, all errors are due to the noise.

5.4 Experimental setup

For the clustering task, all 2-d datasets are tested. Neighborhood size *k* is set to be 30 for all algorithms. Top-*N* is set to be half of the amount of outliers for all algorithms that need top-*N* setting.

For outlier detection task, all datasets are tested. Neighborhood size *k* is set to be 30 for all algorithms for all 2-d datasets. For all multi-dimensional datasets, as they are real-world datasets, we vary *k* from 2 to 100 to find the optimal value for it for each method. For Country datasets, we vary *k* from 2 to 50 to find the optimal value for it.

Clustering algorithms (*k*-means and random swap) are implemented in C, and all outlier detection algorithms are implemented in Python. All codes are run on MacOS High Sierra 10.13 with Intel Core i7 and 16-GB memory.

5.5 Evaluating measurement

To compare the clustering results from different algorithms, the *centroid index* (CI) [28] and *normalized mutual information* (NMI) [6] are employed in this thesis. CI is a cluster level measure, which counts how many cluster centroids are wrongly located. The value CI=0 indicates that, at the cluster level, the clustering is correct with respect to the ground truth. NMI is a point-level measure that calculates the amount of information the clustering shares with the ground truth. Value 1 indicates that the result is identical to the ground truth.

To compare the outlier detection results of different algorithms, we use the *area under the receiver operating characteristics curve* (ROC AUC) [6] to evaluate the performance. It ranges between 0 and 1, and value 0 indicates the worst matching to ground truth while value 1 indicates to a perfect matching.

5.5.1 Clustering: Centroid index (CI)

Centroid index is a cluster-level similarity measure proposed in [28] that estimates the similarity of two clustering solutions based on their centroids. The centroid index for two solutions $C = \{c_1, c_2, \ldots, c_{K_1}\}$ and $C' = \{c'_1, c'_2, \ldots, c'_{K_2}\}$ is calculated as follows. First, the *nearest neighbor mappings* $C \rightarrow C'$ are constructed:

$$q_i \leftarrow \underset{1 \le j \le K_2}{\arg\min} \left\| c_i, c'_j \right\|^2 \quad \forall\, i \in [1, K_1] \quad (16)$$

A target centroid c'_j is an *orphan* if there are no centroids c_i that consider it the nearest:

$$\text{orphan}(c'_j) = \begin{cases} 1, & q_i \ne j \; \forall\, i \\ 0, & \text{otherwise} \end{cases} \qquad (17)$$

The dissimilarity between C and C' is then defined as the number of orphans in the target clustering solution:

$$\text{CI}_1(C, C') = \sum_{j=1}^{K_2} \text{orphan}(c'_j) \qquad (18)$$

This mapping is not symmetric, meaning that in general $\text{CI}_1(C, C') \ne \text{CI}_1(C', C)$. A symmetric variant is defined in [28] as follows:

$$\text{CI}(C, C') = \max\{\text{CI}_1(C, C'), \text{CI}_1(C', C)\} \; (19)$$

The value $\text{CI} = 0$ means that each centroid c_i is mapped to exactly one c'_j and vice versa. In other words, the two solutions have the same global structure. A value of $\text{CI} > 0$ gives the number of clusters that have been allocated differently between the solutions.

Fig. 22 shows an illustration of how the centroid index is calculated. The ground truth centroids are shown in blue, and the clustering solution is shown in red. Each solution centroid is mapped to the nearest ground truth centroid. The numbers beside the ground truth centroids indicate the number of solution centroids that are mapped to each.

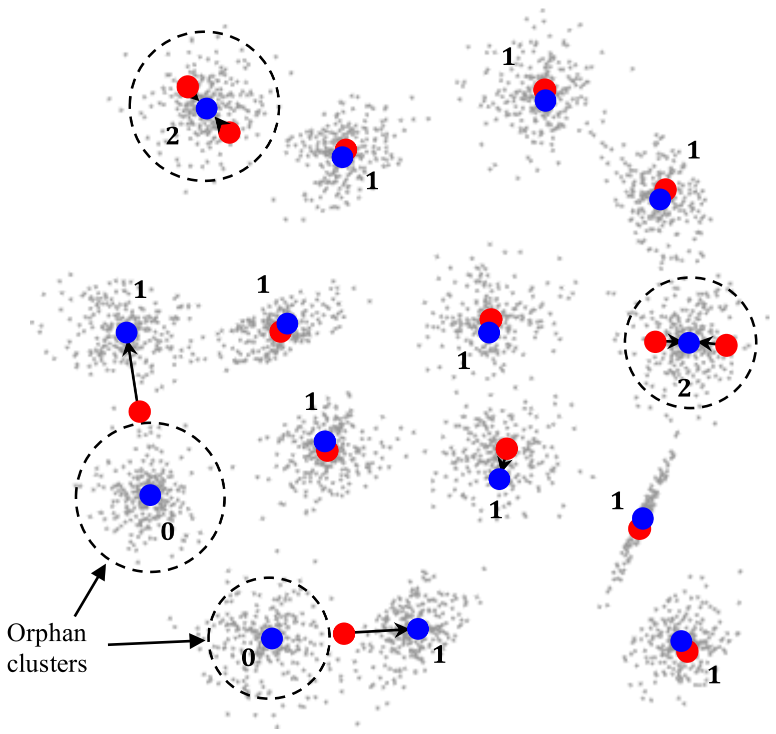


Fig. 22. A clustering solution with CI equaling to 2.

If a ground truth centroid has no solution centroids mapping to it, it is considered an orphan. The differences in the global clustering structure are highlighted with dotted lines. In the above case, there are two orphan clusters, resulting I CI equaling to 2.

5.5.2 Clustering: Normalized Mutual Information (NMI)

The aim of clustering is to measure if objects are correctly labeled. Given the knowledge of the ground truth class assignments true labels and our clustering algorithm assignments of the same samples predicted labels, the *mutual information* is a function that measures the agreement of the two assignments, ignoring permutations. Assume two label assignments (of the same N objects), U and V. *Normalized mutual information* (NMI) is defined as:

$$\mathrm{NMI(U, V)} = \frac{MI(U,V)}{\sqrt{H(U)V(U)}} \qquad (20)$$

The mutual information (MI) between U and V is calculated by:

$$\mathrm{MI(U,V)} = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \quad (21)$$

where $P(i,j) = |U_i \cap V_j|/N$ is the probability that an object picked at random falls into both classes U_i and V_j.

$$\mathrm{H(U)} = -\sum_{i=1}^{|U|} P(i)\log\big(P(i)\big) \qquad (22)$$

where $P(i) = |U_i|/N$ is the probability that an object picked at random from U falls into class U_i. Likewise for V with $P'(j) = |V_j|/N$

$$\mathrm{H(V)} = -\sum_{i=1}^{|V|} P'(j)\log\big(P'(j)\big) \qquad (23)$$

5.5.3 Outlier detection: ROC AUC

Outlier detection methods are measured by ROC curve drawn on the outlier score. The idea of the ROC curve is to plot true positive rate against false positive rate, over ranked outlier score at various threshold values. The area under the ROC curve provides accuracy evaluation, which ranges from 0 to 1. ROC value of a perfect outlier prediction equals to 1.

For a data set D, an outlier set G, and a predicted outlier set *S*(*t*) decided by a threshold *t* on the outlier scores. In this case, the true positive rate is graphed against the false positive rate. The true positive rate TPR(*t*) and the false positive rate FPR(*t*) is same in [29], and these definitions are as follows:

$$\mathrm{TPR}(t)=\mathrm{Recall}(t) = \frac{|S(t) \cap G|}{|G|} \qquad (24)$$

$$\mathrm{FPR}(t) = \frac{|S(t) - G|}{|D - G|} \qquad (25)$$

For every threshold *t*, we can get a pair of TPR and FPR. ROC curve is drawn with TPR against FPR, see Fig. 23. The ROC curve endpoints are always at (0,0) and (1,1). The area under the ROC curve is called ROC value, which ranges from 0 to 1 and perfect prediction will be 1.

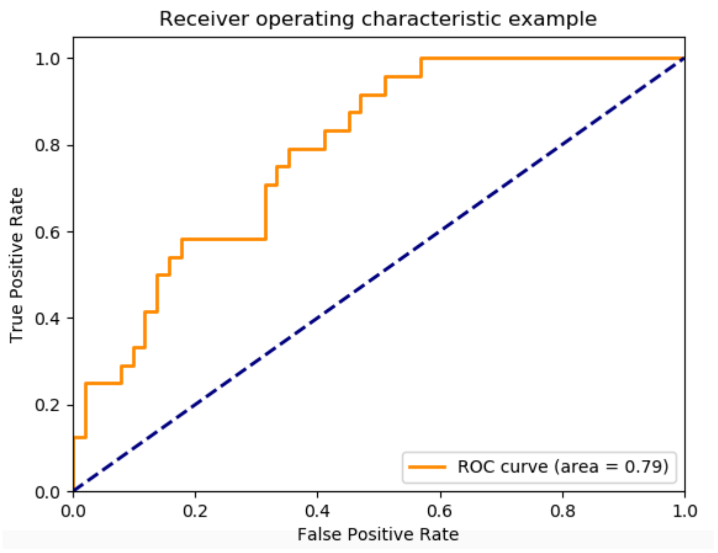


Fig. 23. ROC AUC curve example[8].

[8] http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

6 Experiment results and discussion

The experiment results are summarized from Table IV to Table XI. In Table III, Table IV, and Table V, it shows the clustering results. In Table VI, Table VII, and Table VIII, it shows the outlier detection results.

The effect of *k* nearest neighbor size in mean-shift process are summarized in Table IX and Table X; relationship between noise level and shifting iterations are summarized in Table XI, Table XII, and Fig. 26 and running time of algorithms are summarized in Table XIII.

6.1 Clustering results

The clustering results are summarized in Table III, Table IV, and Table V, both at the point level (NMI) and at the cluster level (CI). Neighborhood size *k* is fixed to 30 and top-*N* is half of the amount of noise.

Several observations can be made from the results in Table III. First, all datasets can be perfectly clustered using random swap algorithm if there is no noise (CI=0 for 0% noise). The more noise there is in the data, the more the clustering result deteriorates. With 8% noise, there are already errors with 10 clusters (CI=10.09), on average, and it doubles (CI=20.45) when the noise level reaches 128%.

Second, both the proposed mean-shift filtering and all the outlier removal methods improve the clustering up to noise level 8%. The only exception is NC which perform slightly worse with 0.025% noise level. Among the methods, our mean-shift variant (MOD) clearly outperforms the others except KNN. However, with higher noise levels (16-128%) MCD performs better although the proposed method is still the third best and clearly outperforming the others.

When changing the algorithm to standard k-means, we can also see the effect of the clustering algorithm. Even with clean data, there are already errors: CI=3.27, on average. Otherwise, the behavior is similar: adding more noise worsens the clustering. Except KNN, the proposed method also works best until 8% of noise level, after which MCD becomes more effective.

The choice of the algorithm also has two side effects. First, the medoid shift (DOD) is sometimes better than the mean-shift (MOD) when using *k*-means. Second, the *k*-means can become slightly better when the noise increases beyond 8%. This is a side effect of the *k*-means algorithm. Adding noise generates fake low-density clusters, which a good algorithm can find more effectively. *K*-means, on the other hand, is known to have problems when the clusters have less overlap [29]. This makes it less optimized for the noisy data.

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

TABLE III

SUMMARY OF AVERAGE CLUSTERING RESULTS (**CI-VALUES**) OF ALL 2-D DATASETS.

RANDOM SWAP

		Noise level											
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
Noise 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
Noise 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
	MCD	-	0.91	1.55	2.27	3.64	4.91	6.82	8.64	10.55	13.64	16.18	6.91
	NC	-	1.73	2.18	3.27	4.64	7.18	9.55	12.73	15.64	17.82	20.64	9.54
	KNN	-	**0.09**	0.73	1.36	2.45	2.82	5.18	6.64	10.91	14.00	17.82	6.20

***K*-MEANS**

		Noise level											
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
Noise filtering	DOD	5.55	4.27	3.73	3.18	4.64	5.00	5.91	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
Noise 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
	MCD	-	3.00	4.09	4.36	4.91	5.00	5.36	**6.73**	**8.55**	**11.09**	**14.73**	6.78
	NC	-	3.36	4.09	4.45	4.55	6.27	9.36	9.00	12.27	15.09	19.91	8.84
	KNN	-	0.18	0.64	1.27	2.45	2.82	5.18	7.55	11.55	14.64	17.36	6.36

Table IV shows the number of datasets that were perfectly clustered. The results show the same trend as Table III; the more noise, the less successful the clustering. Random swap solves all 11 datasets if no noise. Even a small amount of noise causes errors in some datasets, and eventually, only one dataset is correctly clustered anymore after 16% noise level. The proposed mean-shift filtering is slightly better: with 2% noise, it still solves 5 datasets whereas the others only 2 or 3.

TABLE IV

SUMMARY OF CLUSTERING RESULTS AS NUMBER OF DATA SETS FOR WHICH **CI** IS 0 IS REACHED.

RANDOM SWAP

		Noise level											
Type	**Method**	**0%**	**0.025%**	**0.5%**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
None	None	11	7	4	3	2	1	1	1	1	1	1	2.20
Noise filtering	DOD	9	8	8	6	6	1	1	1	1	1	1	3.40
	MOD	**10**	**10**	**9**	9	5	3	2	1	1	1	1	4.20
Noise removal	LOF	-	9	7	5	2	1	1	1	1	0	1	2.80
	ODIN	-	7	5	4	3	1	1	1	1	1	1	2.50
	MCD	-	8	7	5	3	1	1	1	0	0	1	2.70
	NC	-	6	5	4	2	1	1	1	1	**2**	1	2.40
	KNN	-	**10**	**9**	9	6	5	2	1	1	1	0	4.40

***K*-MEANS**

		Noise level											
Type	**Method**	**0%**	**0.025%**	**0.5%**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
None	None	2	4	2	2	1	1	1	0	1	1	1	1.40
Noise filtering	DOD	1	1	1	2	1	1	3	1	1	1	**2**	1.40
	MOD	2	3	1	1	2	1	2	0	1	1	1	1.30
Noise removal	LOF	-	1	2	2	1	1	1	1	0	1	1	1.10
	ODIN	-	2	1	1	1	1	1	1	1	1	1	1.10
	MCD	-	2	2	2	1	1	1	1	1	**2**	**2**	**1.50**
	NC	-	1	2	1	2	2	1	1	1	1	1	1.30
	KNN	-	**10**	**9**	9	6	5	2	1	1	1	0	4.40

Table V shows that in the point level results show that the mean-shift improves both *k*-means and random swap clustering results from about NMI = 0.85 to 0.88. The mean-shift is slightly better than the medoid-shift. It seems to indicate that the CI measure can reflect the clustering results better than NMI as the gaps of NMI are small but the difference from varying clustering results is significantly big.

TABLE V

SUMMARY OF CLUSTERING RESULTS (**NMI**) OF ALL 2-D DATASETS.

RANDOM SWAP

		Noise level											
Type	**Method**	**0%**	**0.025%**	**0.5%**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
None	None	0.93	0.92	0.92	0.91	0.90	0.86	0.84	0.82	0.80	0.74	0.72	0.84
Noise filtering	DOD	**0.93**	**0.93**	0.86	**0.93**	**0.92**	0.91	0.89	0.85	0.81	0.80	0.74	0.86
	MOD	**0.93**	**0.93**	**0.93**	**0.93**	**0.92**	0.91	0.89	0.85	0.82	0.80	0.77	0.87
Noise removal	LOF	-	**0.93**	**0.93**	0.92	0.91	0.89	0.85	0.82	0.80	0.73	0.71	0.85
	ODIN	-	**0.93**	0.92	0.92	0.90	0.88	0.84	0.82	0.80	0.77	0.72	0.85
	MCD	-	**0.93**	**0.93**	0.92	0.91	0.90	0.87	0.85	0.81	0.79	**0.80**	0.87
	NC	-	0.92	0.92	0.92	0.90	0.88	0.85	0.83	0.81	0.74	0.72	0.85
	KNN	-	**0.93**	**0.93**	**0.93**	**0.92**	**0.92**	0.90	0.86	0.84	0.80	0.78	**0.88**

***K*-MEANS**

		Noise level											
Type	**Method**	**0%**	**0.025%**	**0.5%**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
None	None	0.88	0.89	0.88	0.88	0.88	0.89	0.86	0.84	0.84	0.80	0.77	0.85
Noise filtering	DOD	0.87	0.88	0.88	0.89	0.88	0.88	0.89	0.87	0.85	0.81	0.75	0.86
	MOD	0.86	0.88	0.88	0.88	0.89	0.88	0.89	0.85	0.84	0.82	0.76	0.86
Noise removal	LOF	-	0.88	0.89	0.88	0.88	0.89	0.85	0.85	0.80	0.77	0.71	0.84
	ODIN	-	0.88	0.89	0.88	0.88	0.89	0.85	0.85	0.80	0.77	0.71	0.84
	MCD	-	0.89	0.88	0.88	0.88	0.88	0.89	**0.87**	**0.85**	**0.84**	**0.81**	0.87
	NC	-	0.89	0.88	0.88	0.88	0.88	0.85	0.87	0.85	**0.84**	**0.81**	0.87
	KNN	-	0.89	0.88	0.88	0.88	0.88	0.90	0.87	0.83	0.78	0.75	0.85

One good property of noise removal is that it should destroy clean data. In our case, slight errors were detected: CI=0 is increasing to CI=0.18, on average, if mean-shift is applied for clean data. However, most of this was caused by b1 and b2 dataset.

The main problem of the traditional outlier removal methods is that they are based on thresholding, and require knowing the noise level (Top-*N* parameter). If the correct noise level is used (Top-*N* parameter), their performance is not far from our method. However, if we apply some default value like1% or 8%, then they start to fail much more severely. This shows the importance of noise filtering (our approach) comparing to the traditional noise (outlier) removal. The clustering results on A1 dataset are illustrated in Fig. 24 and 25, we can see that proposed methods outperforms than others with both *k*-means and Random swap.

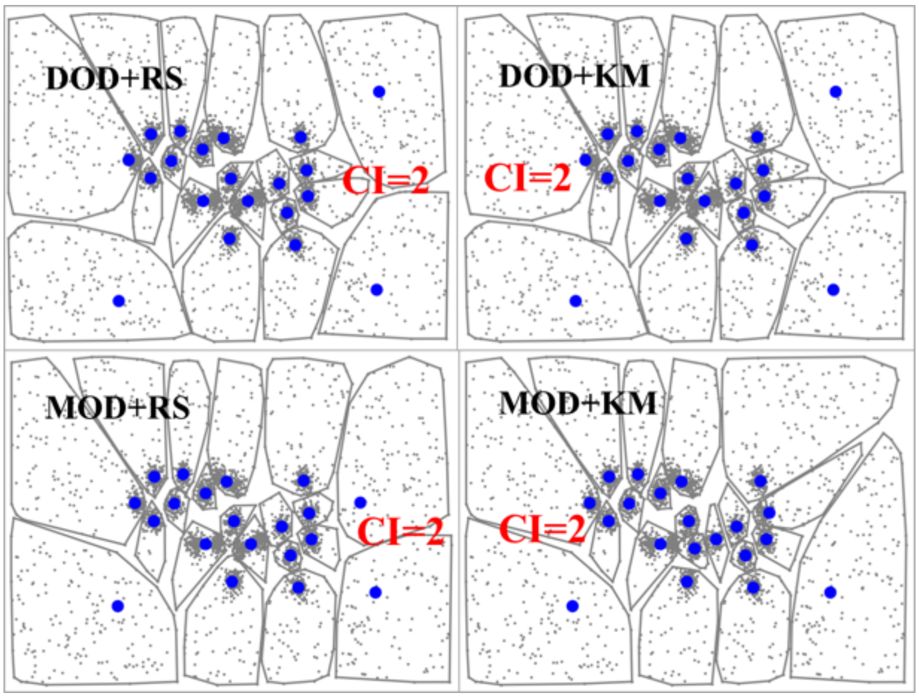


Fig. 24. Noise filtering + clustering results on 8% noisy A1.

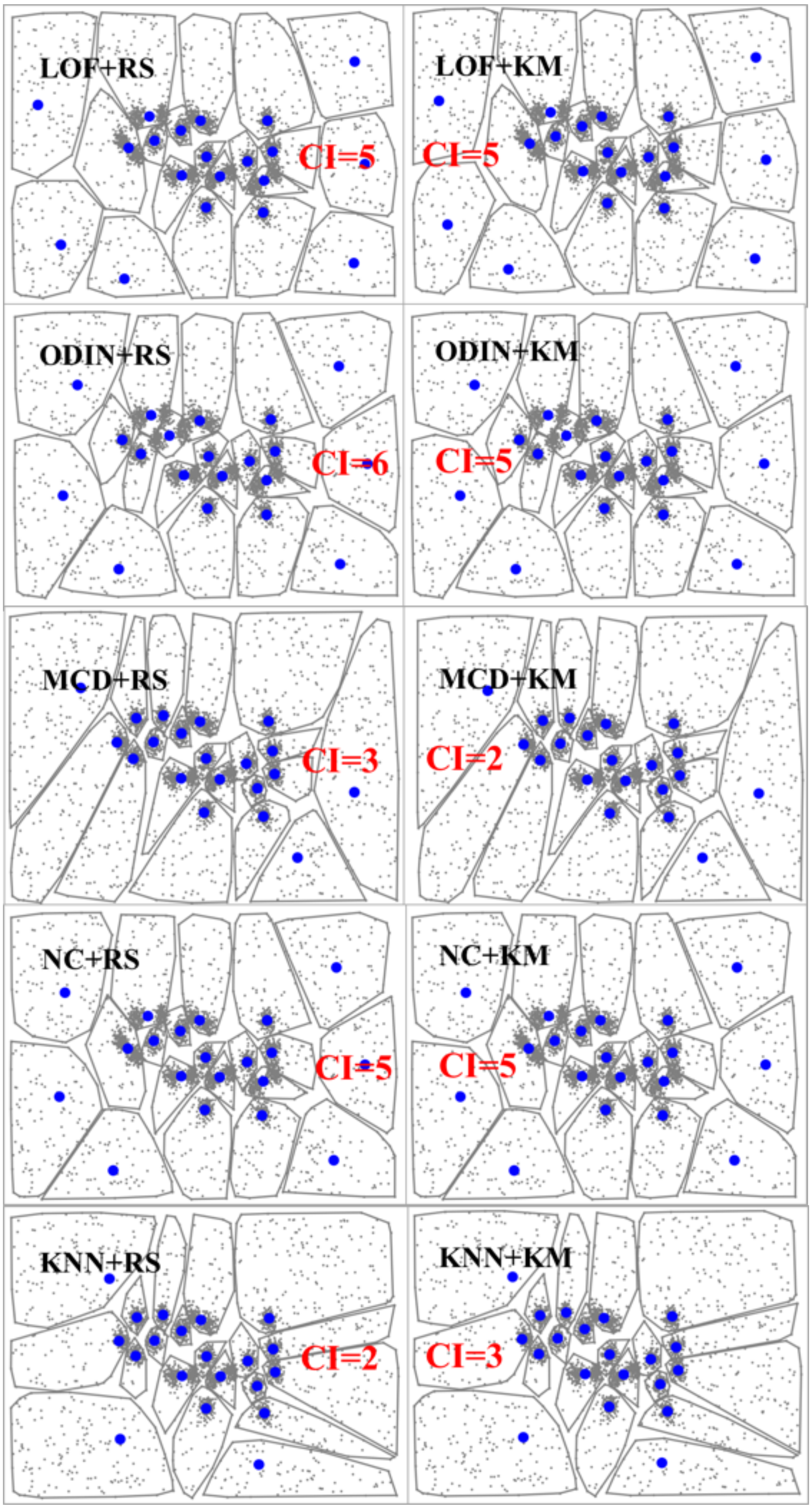


Fig. 25. Noise removal + clustering results on 8% noisy A1.

6.2 Outlier detection results

The outlier detection results are summarized in Table VI (2- dimensional datasets), Table VII (high dimensional datasets), and Table VIII (string dataset). For 2-d datasets, Table VI, it shows that with the increase of the noise level, the performance of all methods tend to drop but my proposed methods and KNN are always far better than the others regardless the noise level. The exception is MCD, which is better than my proposed methods from 32% to 128%. My proposed methods are equally good with KNN with low noise level (0.025% to 16%), but KNN is better when the noise level is high.

TABLE VI

AVERAGE OUTLIER DETECTION RESULTS FOR ALL 2-D DATASETS IN TABLE II WITH DIFFERENT NOISE LEVEL. *K* NEAREST NEIGHBOR IS SET TO 30.

ROC AUC

	Noise level										
Method	**0.025%**	**0.05 %**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
DOD	**1.00**	**1.00**	0.99	0.99	0.99	0.97	0.95	0.92	0.88	0.82	0.95
MOD	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.98**	0.95	0.90	0.83	**0.96**
LOF	0.99	0.97	0.95	0.91	0.86	0.68	0.53	0.42	0.36	0.32	0.70
ODIN	0.98	0.96	0.94	0.90	0.81	0.71	0.65	0.61	0.58	0.57	0.77
MCD	0.95	0.94	0.95	0.96	0.96	0.95	0.95	**0.96**	**0.96**	**0.96**	0.95
NC	0.96	0.92	0.88	0.83	0.72	0.62	0.54	0.48	0.45	0.44	0.68
KNN	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**

For real-word high dimensional datasets in Table VII, the best result per method is shown with *k* ranging from 2 to 100. We can see that the proposed methods outperform the others in KDDCup99, SpamBase, and Parkinson datasets.

TABLE VII

OPTIMAL RESULTS FOR KDDCUP99, SPAMBASE AND PARKINSON DATASETS WITH *k* NEAREST NEIGHBORS RANGING FROM 2 TO 100.

ROC AUC

Method	**KDDCup99 (outlier: 0.42%)**		**SpamBase (outlier: 39.91%)**		**Parkinson (outlier: 75.38%)**	
	ROC	k	ROC	k	ROC	k
DOD	0.94	87	**0.60**	2	**0.71**	3
MOD	**0.99**	99	**0.61**	2	**0.67**	2
LOF	0.85	100	0.50	2	0.61	8
ODIN	0.81	100	0.52	41	0.53	2
MCD	0.97	-	0.45	-	0.65	-
NC	0.69	80	0.55	2	0.61	21
KNN	**0.99**	85	0.56	93	0.64	5

For String (Country) dataset, the results based on Edit distance are summarized in Table VII. We can see that the proposed DOD with ROC = 0.85 clearly outperforms KNN with ROC= 0.60 and ODIN with ROC = 0.50.

TABLE VIII
OPTIMAL RESULTS FOR COUNTRY DATASET WITH DIFFERENT NOISE LEVEL
WITH *k* NEAREST NEIGHBORS RANGING FROM 2 TO 100
BASED ON EDIT DISTANCE.
ROC AUC

	Noise level							
Method	**10%**		**20%**		**30%**		**40%**	
	ROC AUC	k	ROC AUC	k	ROC AUC	k	ROC AUC	k
DOD	**0.86**	2	**0.84**	4	**0.84**	5	**0.83**	5
KNN	0.59	3	0.60	3	0.61	3	0.60	3
ODIN	0.48	2	0.49	2	0.50	2	0.50	2

To sum up, our proposed methods outperform others despite of the dataset type or dimensional. It performs very well in real-word datasets with high outlier level.

6.3 Effect of *k* in mean-shift process

We also tested the effect of the parameter *k*, namely *k* in *k*-nearest neighbor, during the mean-shift process. We decrease and increase the *k* value to see how it affects the results and found that it has no much effect on mean-shift but has slight effect on medoid-shift. This is because it produces plenty of duplicates in medoid-shift process. Further, we found that we can improve the DOD results by increasing *k* value during medoid-shift process, because it can bring new objects into the nearest neighborhood to help outliers to move bigger distance, especially when duplicates exist in *k* nearest neighborhood. The tested results are shown in Table IX and Table X.

TABLE IX
OPTIMAL DOD RESULTS FOR COUNTRY DATASETS WITH DIFFERENT NOISE LEVEL
BASED ON EDIT DISTANCE
ROC AUC

	Noise level			
k in medoid-shift (1st, 2nd, 3rd) iteration	**10% (k=2)**	**20% (k=4)**	**30% (k=5)**	**40% (k=5)**
(k, 2k, 3k)	0.85	**0.84**	**0.84**	**0.84**
(k, k, k)	**0.86**	**0.84**	**0.84**	0.83
(k, k/2, k/3)	**0.86**	0.83	0.82	0.83

TABLE X

AVERAGE DOD RESULTS FOR ALL 11 2-D DATASETS WITH DIFFERENT NOISE LEVEL IN TABLE II.

BASED ON EUCLIDEAN DISTANCE (*K* = 30).

ROC AUC

	Noise level										
k in medoid-shift (1st, 2nd, 3rd) iteration	**0.025%**	**0.5%**	**1%**	**2%**	**4%**	**8%**	**16%**	**32%**	**64%**	**128%**	**Avg.**
(k, 2k, 3k)	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.97**	**0.94**	**0.90**	**0.84**	**0.96**
(k, k , k)	**1.00**	**1.00**	0.99	**0.99**	**0.99**	0.97	0.95	0.92	0.88	0.82	0.95
(k, k/2, k/3)	**1.00**	0.99	0.99	**0.99**	0.98	0.96	0.94	0.90	0.86	0.80	0.94

6.4 Effect of mean-shift *iteration*s

We also study the relationship between noise level and shifting iterations. We repeat the experiment 100 times on S1 dataset with random noise on each level. The average ROC AUC results are shown in Fig. 10 and Table XI. We can see that with the increase of the noise level, it needs more shifting iterations to reach the optimal measure, which are 3 iterations for MOD and 5 iterations for DOD.

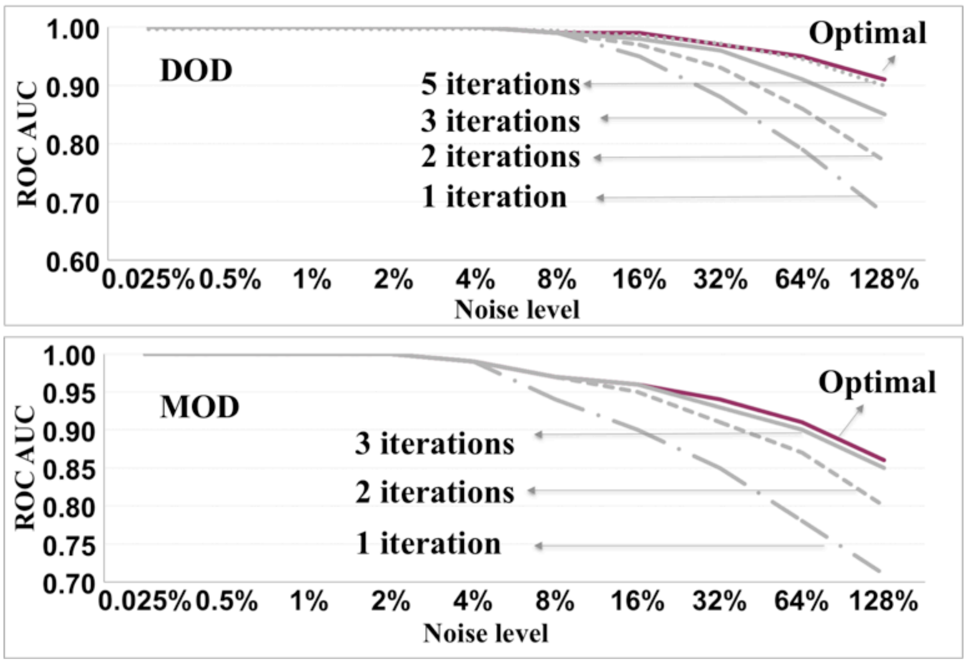


Fig. 26. The effect of the noise level on the number of shifting iterations needed on S1 dataset. Results are shown for medoid-shift (above) and mean-shift (below).

From Table XI, the average of optimal iteration is around 3. As there is no much gap between 3 and 5 iterations for DOD and also for the purpose of saving computing time, we set it as 3 iterations as default.

However, even it's good enough to use 3 iterations for all tested cases. If the noise level is very low (below 4%), we can use 1 or 2 iterations to save computing.

TABLE XI
BEST NUMBER ITERATIONS FOR AVERAGE RESULTS OF S1 DATA SET WITH NOISE LEVELS (REPEATED 100 TIMES)
(**ITERATIONS**)

	Noise level										
Method	**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

For string datasets, the iteration relationship is shown in Table XII. We can see the result of 3 iterations is better than 1 iteration, except with low outlier level (10%). The gap between 1 iteration and 2 iterations is big but small between 2 iterations and 3 iterations, hence it is reasonable to use 3 iterations.

TABLE XII
THE EFFECT OF MEDOID-SHIFT ITERATIONS ON DOD RESULTS
FOR **COUNTRY** DATASETS WITH DIFFERENT NOISE LEVEL
BASED ON EDIT DISTANCE
ROC AUC

	Noise level			
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**

6.5 Running time

All methods based on *k*-NN require O(N^2) calculations. To address this slowness of the brute-force approach, we use the KD-tree technique [30] with all algorithms. It works fast in low dimensional but can become inefficient with higher dimensions (D>20). In this case, faster approximate like *NNDES* [31] or the *Random pair divisive* (RP-div) [32] can become more efficient. Running times of the 9 datasets are summarized in Table XIII. The proposed method is similar fast as KNN, LOF and MCD. However, DOD is slower than MOD.

TABLE XIII
RUNNING TIMES ON SPAMBASE DATASETS (K=100)

Method	DOD	MOD	LOF	ODIN	MCD	NC	KNN
Time (s)	17	11	12	3	12	48	15

7 Conclusions

Mean-shift and medoid-shift were proposed as a separate noise filtering process before clustering. The results show that mean-shift is more effective than medoid-shift and that they improve both *k*-means and random swap clustering. The proposed approach outperforms four existing outlier filtering methods: KNN, LOF, MCD, NC, and ODIN.

Mean-shift outlier detection (MOD) was proposed. The results show that mean-shift variant (MOD) is slightly more effective than the medoid-shift (DOD). For the studied noise patterns, the proposed approach clearly outperforms existing outlier detection methods: LOF, MCD, NC, ODIN and outperforms than KNN in real-word datasets and string datasets but in not goo d as KNN in some 2-d dataset cases. The most important property of the proposed method is that it can effectively detect string outliers based on Edit distance and suitable for the cases when the amount of outliers is big.

References

1. A.M. Ali, P. Angelov, "Anomalous behavior detection based on heterogeneous data and data fusion," *Soft Computing*, 2018, 3187–3201.
2. T.V. Pollet, L. van der Meij, "To remove or not to remove: the impact of outlier handling on significance testing in testosterone data," *Adaptive Human Behavior and Physiology*, 3 (1), 43-60, Mars 2017.
3. H.-P. Kriegel, P. Kröger, and A. Zimek, “Outlier detection techniques,” *13th Pacific-Asia Conf. Knowledge Discovery Data Mining*, 1–73, 2009.
4. J.W. Yang，S. Rahardja, and P. Fränti, "Outlier detection: How to Threshold Outlier Scores?", manuscript, submitted.
5. M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Int. Conf. on Knowledge Discovery and Data Mining* (KDD), 226-231, 1996.
6. J.W.Yang，S. Rahardja, and P. Fränti, "Mean-shift outlier detection," *Int. Conf. Fuzzy Systems and Data Mining* (FSDM), Nov, 2018.
7. V. Barnett, "The study of outliers: purpose and model," *Applied Statistics*, 27(3), 242–250, 1978.
8. T. Johnson, I. Kwok, and Ng. Raymond, “Fast computation of 2-dimensional depth contours”, In Proc. *Int. Conf. on Knowledge Discovery and Data Mining* (KDD), New York, NY, 1998
9. P. Fränti, J. Kivijärvi, and O. Nevalainen: "Tabu search algorithm for codebook generation in VQ," *Pattern Recognition*, 31 (8), 1139-1148, August 1998.
10. S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Record*, 29 (2), 427-438, June 2000.
11. V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using k-nearest neighbor graph," *Int. Conf. on Pattern Recognition* (ICPR), 430-433, August 2004.
12. M.M. Breunig, H. Kriegel, R.T. Ng and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Int. Conf. on Management of Data*, 29 (2), 93-104, May 2000.
13. G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenkova, E. Schubert, I. Assent, and M. E. Houle, “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”, *Data Mining and Knowledge Discovery*, 30 (4), 891–927, 2016.
14. M.R. Brito, E.L. Chavez, A.J. Quiroz, J.E. Yukich, "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection," *Statistics & Probability Letters*, 35 (1), 33-42, 1997.
15. P. J. Rousseeuw, "Least median of squares regression," *J. Am Stat Ass*, 79:871, 1984
16. M. Hubert, and M. Debruyne, "Minimum covariance determinant," *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1), 36-43. doi:10.1002/wics.61, 2009.
17. X. Li, J. Lv, and Z. Yi, "An efficient representation-based method for boundary point and outlier detection," *IEEE Trans. on Neural Networks and Learning Systems*, 29 (1), January 2018.
18. D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24 (5), 603–619, May 2002.
19. Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17 (8), 790–799, August 1995.
20. D.-M. Tsai and J.-Y. Luo, "Mean shift-based defect detection in multicrystalline solar wafer surfaces," *IEEE Trans. on Industrial Informatics*, 7 (1), 125-135, February 2011.
21. Y.A. Sheikh, E.A. Khan, T. Kanade, "Mode-seeking by Medoidshifts," *IEEE Int. Conf. on Computer Vision* (ICCV), October 2007.
22. Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17 (8), 790–799, August 1995.
23. H.V. Nguyen, V. Gopalkrishnan, "Feature extraction for outlier detection in high-dimensional spaces," *J Mach, Learn Res Proc Track* 10:66–75, 2010
24. V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen and P. Fränti, "Improving k-means by outlier filtering," *Scand. Conf. on Image Analysis* (SCIA), Lecture Notes of Computer Science, LNCS vol. 3540, 978-987, June 2005.
25. E. Forgy, "Cluster analysis of multivariate data: efficiency vs. interpretability of classification," *Biometrics*, 21, 768-780, 1965.
26. P. Fränti, "Efficiency of random swap clustering," *Journal of Big Data*, 5:13, 1-29, 2018.

27. P. Fränti, J. Kivijärvi, and O. Nevalainen: "Tabu search algorithm for codebook generation in VQ," *Pattern Recognition*, 31 (8), 1139-1148, August 1998.
28. P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, 2018.
29. P. Fränti, M. Rezaei and Q. Zhao, "Centroid index: Cluster level similarity measure", *Pattern Recognition*, 47 (9), 3034-3045, 2014.
30. R. Kannan, H. Woo, C. C. Aggarwal and H. Park. "Outlier Detection for Text Data," *International Conference on SIAM Data Mining*, 2017.
31. J.L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, 1975
32. W. Dong, C. Moses, K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," *ACM Int. Conf. on World Wide W*eb, 577–586, 2011.
33. S. Sieranoja and P. Fränti, "Fast random pair divisive construction of kNN graph using generic distance measures," *Int. Conf. on Big Data and Computing* (ICBDC), April 20