On Saving Outliers for Better Clustering over Noisy Data

Shaoxu Song, Fei Gao, Ruihong Huang, Yihan Wang
Beijing Key Laboratory for Industrial Bigdata System and Application, School of Software, Tsinghua University
Beijing, China
sxsong@tsinghua.edu.cn,{gao-f16,hrh16,wangdh16}@mails.tsinghua.edu.cn

ABSTRACT

Clustering is often distracted by errors, frequently observed in almost all areas, ranging from online questionnaire to sensor reading in IoT. The dirty data values not only make themselves (the corresponding tuples) outlying, but also mislead the clustering of remaining tuples, e.g., mistakenly splitting a cluster into two or distorting the cluster center. The reason is that the traditional clustering methods either simply ignore the outliers such as DBSCAN or assign them to the closest clusters anyway, e.g., in K-Means. In this paper, we propose to save the outliers for better clustering. The idea is to adjust the erroneous values (often minimally) of the outlier in order to make it appear normally. That is, the tuples after adjusting values are no longer outlying, and thus will be clustered without distracting others. The outlier saving by value adjustment is designed to work with any clustering methods (e.g., DBSCAN or K-Means). Our technical contributions include: (1) showing NP-hardness of the outlier saving problem for clustering, (2) deriving lower and upper bounds of the optimal solutions, and (3) devising approximation algorithm with performance guarantees referring to the aforesaid bounds. Experiments on datasets with real-world outliers demonstrate the higher accuracy of our proposal, compared to the state-of-the-art approaches. Remarkably, we show that the adjusted data with outlier saving indeed improve significantly clustering, as well as other applications such as classification and record matching.

ACM Reference Format:
Shaoxu Song, Fei Gao, Ruihong Huang, Yihan Wang. 2021. On Saving Outliers for Better Clustering over Noisy Data. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3448016.3457271

1 INTRODUCTION

Noisy data have been widely recognized [32] and obviously damage the downstream applications such as clustering [31]. For example, consider the Iris dataset in Figure 1(a), to cluster iris plants according to the petal length and width. Errors may occur occasionally in the data, e.g., owing to mistakenly recording the width in *inch* instead of *cm*. As shown in Figure 1(b), due to the errors, the corresponding tuples become outlying denoted by red triangles, while

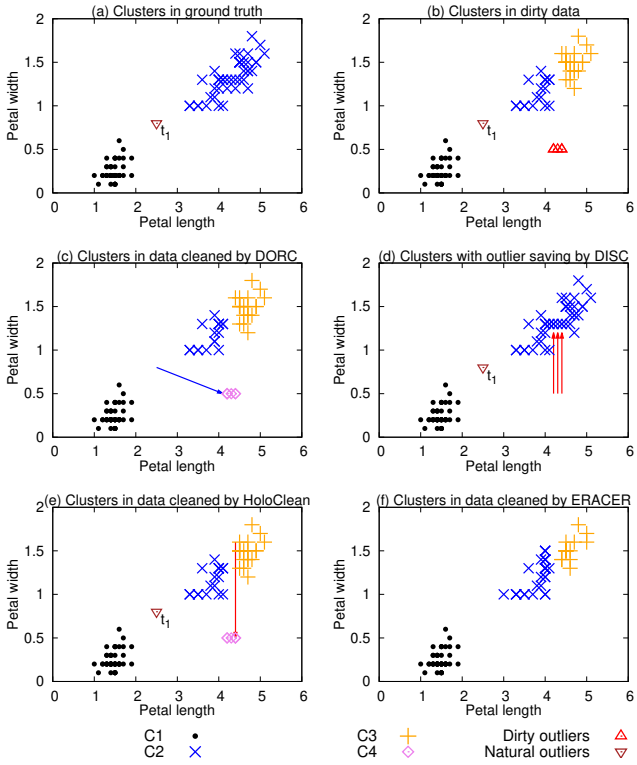

Figure 1: Clusters in (a) ground truth, (b) dirty data with outliers, (c)-(d) with outliers handled by various approaches

the remaining data are improperly split into three clusters, in the density-based DBSCAN clustering [21]. For K-Means clustering [26] with K=2, it is still possible to segment the clusters. For instance, in Figure 1(b), if the points at (3, 1) and (5, 1.5) are selected as the cluster centers, the points in C1 (black) and C2 (blue) will form a cluster whereas the points in C3 (orange) form another. That is, C2 and C3 belonging to the same cluster in ground truth are still split. Similar examples are also illustrated in Figure 1 in [43].

1.1 Outlier Saving for Clustering

To save the outlier for better clustering, in this paper, we propose to adjust the erroneous values of the outlier to make it an inlier. For example, as shown in Figure 1(d), by adjusting the erroneous width values in *inch* to *cm*, denoted by red arrows, the outlying points become inliers. Two clusters are returned correctly as the ground truth in Figure 1(a).

The existing method DORC [45] also proposes to find sufficient neighbors in a cluster by cleaning the noisy data. It substitutes

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457271

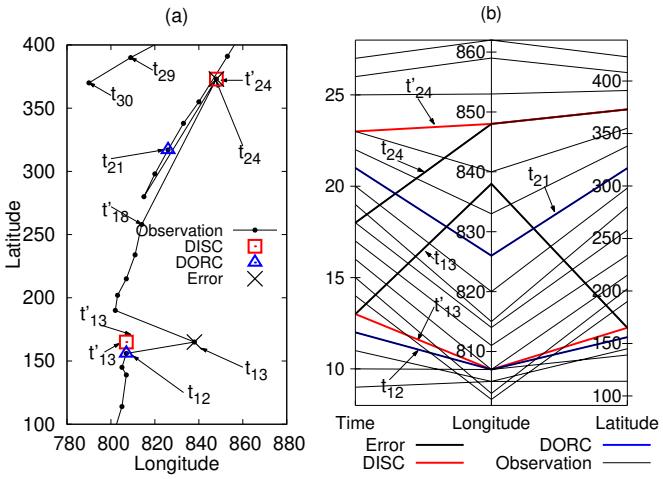

Figure 2: Saving outliers t_{13} and t_{24} by the proposed DISC with value adjustment (returning t'_{13} and t'_{24}) and the existing DORC [45] with tuple substitution (returning t_{12} and t_{21})

(error) tuples by the other (non-error) ones, in order to form clusters. That is, all the attribute values in the tuple are substituted by those of another existing tuple, known as *tuple substitution*. In practice, however, errors may occur only in some attributes, e.g., usually only one or several sensors are broken at a time among hundreds of sensors packed in a wind turbine. In Figure 1(c), the blue arrow denotes that by substituting both Petal width and length values, the natural outlier is moved to an existing dirty outlier tuple in the newly formed cluster C4. Unfortunately, owing to the excessive tuple substitution, the four-clusters results are still wrong.

Example 1. *Consider a real dataset of GPS readings with three attributes (Time, Longitude, Latitude), where real-world errors are observed owing to device issues. Figure 2(a) shows the latitude and longitude values of the observations, i.e., black dots connected by lines with consecutive timestamps. Figure 2(b) plots parallel coordinates, showing the tuples in three dimensions (Time, Longitude, Latitude).*

There is an obvious error occurring on the longitude value of tuple (point) t_{13}. A possible DISC adjustment of t_{13} is tuple t'_{13}, denoted by red square, which is produced by adjusting the longitude value of t_{13} from 838 to 807. The adjusted t'_{13} is no longer outlying. It will not split erroneously the trajectory into two segments (clusters).

Similarly, an error may also occur on the timestamp of tuple t_{24}, and again mistakenly splits the trajectory into segments (clusters). The DISC adjustment t'_{24} states that the erroneous timestamp value 18 of t_{24} should be adjusted to 23 to meet the distance constraints. It is worth noting that the longitude and latitude values do not need to change, as t'_{24} illustrated in Figure 2.

The existing DORC method [45] substitutes t_{13} and t_{24} by some other tuples t_{12} and t_{21}, respectively, i.e., over-changing all three attributes as shown in Figure 2(b). The excessive yet inaccurate cleaning does not avoid erroneously splitting the trajectory into segments (clusters), again similar to Figure 1(c).

Recall that the error in t_{24} occurs on the timestamp, while its longitude and latitude values are correct. That is, as illustrated in Figure 2(b), t_{24} should occur at Time 24, but was erroneously recorded with Time 18. The method DORC finds an existing tuple, which is closest to t_{24} on all the three attributes Time, Longitude and Latitude, i.e., t_{21} to substitute the erroneous tuple t_{24}. It is worth noting that DORC finds the closest tuple on all three attributes, not on the Longitude and Latitude attributes only as illustrated in Figure 2(a), since errors may occur on any attribute including Time.

Our proposal supports not only numeric data but also textual / categorical data. For instance, in the Restaurant dataset, typos in zip code `RH1O-OAG`, where letters `O` should be digits `0`, make the corresponding restaurants outlying and difficult to identify. After saving the outliers, e.g., by adjusting the erroneous `RH1O-OAG` to `RH10-0AG` as illustrated in Section 2.2, the record matching application is improved in Figure 8 in Section 4.2.5 of experiments.

The target of adjusting values in outliers is to form meaningful clusters. Adjusting values (essentially making new tuples) does not harm the analysis tasks, which usually do not focus on individual tuples. Therefore, as illustrated in Section 4.2, saving outlier by value adjustment is beneficial for not only clustering but also classification and record matching. However, the adjustment of individual tuples may not be one hundred percent accurate, as presented in Section 4.3, and thus not applicable to the transactional tasks, such as alarm system based on individual sensor readings.

1.2 Dirty Outliers versus Natural Outliers

To identify dirty versus natural outliers, we observe the number of attributes that result in the outlier and need adjustment to become an inlier. Intuitively, errors may occur only in some attributes, e.g., usually only one or several sensors are broken at a time among hundreds of sensors packed in a wind turbine. In contrast, natural outliers often show separability in a large number of attributes. e.g., a point from another wind farm or of extreme weather would be outlying with values on temperature, wind speed, deflection angle and so on all distant from the current observations.

For the example in Figure 1(b), to handle the errors introduced by mistakenly recording the width in *inch* instead of *cm*, we only need to adjust the width value to make the (error) point an inlier. However, for a natural outlier of another petal type with both width and length values distinct from other observations, such as t_1, one has to adjust both width and length values to make it an inlier. To give another real-world example in Figure 2, only one error on the longitude value occurs in the dirty outlier t_{13}, while another dirty outlier t_{24} has erroneous timestamp value but correct longitude and latitude values. In contrast, a point from another trajectory (natural outlier) would be outlying with Time, Longitude and Latitude values all distant from the current observations, such as t_{29} or t_{30}.

Nevertheless, we observe the number of attributes that have to be adjusted to make outlying tuples inliers in the experiments in Section 4.3. For the Letter dataset in Figures 10 (c) and (d), only about 2 attributes need to be adjusted among 10, while the GPS dataset in Figure 9 only needs to adjust about 1 attribute.

In this sense, to avoid excessively adjusting the natural outliers, one can choose to return only the adjustment on no more than κ attributes (of the dirty outliers), following the minimum change criteria in data cleaning [23].

It is also suggested in [25] that if the outlier indicates a typographical error by an entry clerk then the entry clerk can be notified and

simply correct the error so the outlier will be restored to a normal record. However, for the natural outliers, it may be worth flagging for further verification. In this sense, we propose to adjust the dirty outliers while leaving the natural outliers unchanged.

1.3 Contribution

Our major contributions in this study are as follows.

We formalize the outlier saving problem, i.e., adjusting values under DIStance constraints for better Clustering (DISC), and show that finding the minimum adjustment is NP-hard in Theorem 1.

We capture the lower and upper bounds of the optimal adjustment. Efficient pruning and approximation are then devised upon these bounds in Algorithm 1. In particular, we show that the approximate algorithm is a constant-factor polynomial-time approximation in certain settings in Proposition 7.

We report an extensive experimental evaluation over several real datasets with real-world outliers. Compared to the state-of-the-art methods such as DORC [45], the proposed DISC shows clearly better clustering performances. Remarkably, outlier saving significantly improves not only clustering but also other applications like classification and record matching.

2 PROBLEM STATEMENT

We start by introducing some preliminaries on distance constraints for clustering in Section 2.1, and then formalize the problem of value adjustment under distance constraints for outlier saving in Section 2.2, together with the hardness analysis in Theorem 1. It leads to the development of approximation algorithm in Section 3.

2.1 Preliminaries on Distance Constraints

Let us first introduce some preliminaries on distance constraints in clustering and outlier detection.

2.1.1 Distance Function. Consider a relation scheme R with m attributes. For each attribute $A \in R$, we associate with a distance function, denoted by $\Delta(t_1[A], t_2[A])$, satisfying (1) non-negativity, $\Delta(t_1[A], t_2[A]) \geq 0$, (2) identity of indiscernibles, $\Delta(t_1[A], t_2[A]) = 0$ iff $t_1[A] = t_2[A]$, (3) symmetry, $\Delta(t_1[A], t_2[A]) = \Delta(t_2[A], t_1[A])$, (4) triangle inequality, $\Delta(t_1[A], t_3[A]) \leq \Delta(t_1[A], t_2[A]) + \Delta(t_2[A], t_3[A])$, where t_1, t_2, t_3 are tuples from R. For instance, we may use edit distance [36] for string values or absolute difference for numerical values.

The distance of t_1, t_2 on a set of attributes $X \subseteq R$, denoted by $\Delta(t_1[X], t_2[X])$, aggregates the distance on each attribute $A \in X$. For example, we may employ L^p metrics. L^1 norm is simply the sum of the distance on each attribute $A \in X$, $\Delta(t_1[X], t_2[X]) = \sum_{A \in X} \Delta(t_1[A], t_2[A])$. L^2 norm is the Euclidean length,

$$\Delta(t_1[X], t_2[X]) = \sqrt{\sum_{A \in X} \left(\Delta(t_1[A], t_2[A])\right)^2}. \tag{1}$$

In the following, by default, we consider L^2 norm as the distance function on multiple attributes. It is worth noting that all the four properties of the distance function on single attribute are also valid for multiple attributes. In addition, we have another monotonicity property, i.e., $\Delta(t_1[X], t_2[X]) \leq \Delta(t_1[X \cup \{A\}], t_2[X \cup \{A\}])$.

In consequence, $\Delta(t_1[R], t_2[R])$ is the distance of two tuples on all attributes in R, or simply $\Delta(t_1, t_2)$.

2.1.2 Distance Constraints. In order to save the outliers for better clustering, in this paper, we propose to study the distance constraints between tuples in clusters. Intuitively, a tuple in a cluster usually has a considerable number of neighbors, whereas data errors or abnormal behaviors rarely occur and thus are often outlying, e.g., as illustrated in Figure 1(b). Therefore, following the notations in clustering [21], we consider the neighbors of a tuple with distances no greater than ε, namely *ε-neighbors*. Statistically speaking, a tuple (object) keeping track of the number of nearest neighbors (with distance $\leq \varepsilon$) may notice an average number of k neighbors. Observing any particular neighbor does not affect the observation of future neighbors. In this sense, the appearance of nearest neighbors can be approximated with Poisson process [39]. It is not only theoretically analyzed [39] but also practically observed in real datasets as illustrated in Figure 5, That is, the number $N(\varepsilon)$ of ε-neighbors follows a Poisson distribution, having

$$p(N(\varepsilon) = k) = \frac{(\lambda\varepsilon)^k}{k!} e^{-\lambda\varepsilon}. \tag{2}$$

Referring to the cumulative distribution function, the probability of a tuple having at least a number η of ε-neighbors is

$$p(N(\varepsilon) \geq \eta) = 1 - e^{-\lambda\varepsilon} \sum_{i=0}^{\lfloor \eta - 1 \rfloor} \frac{(\lambda\varepsilon)^i}{i!}, \tag{3}$$

For the example in Figure 5(a) over the Letter dataset, the average number of neighbors with distances no greater than $\varepsilon = 3.0$ is 51.36, i.e., $\lambda\varepsilon = 51.36$. The probability of a tuple having at least a number $\eta = 18$ of ε-neighbors is 0.99.

Let r be a set of n tuples over R. For a tuple t over R, we denote

$$r_\varepsilon(t) = \{t_i \in r \mid \Delta(t, t_i) \leq \varepsilon\} \tag{4}$$

the ε-neighbors of t in r, i.e., the set of tuples $t_i \in r$ whose distances $\Delta(t, t_i)$ to t are no greater than a *distance threshold* ε. The number $N(\varepsilon)$ of ε-neighbors of t is thus $|r_\varepsilon(t)|$.

Definition 1 (Distance Constraints). *Referring to Formula 3, the distance constraints state that if a tuple t has ε-neighbors in r no less than a neighbor threshold η, i.e., $|r_\varepsilon(t)| \geq \eta$, then the tuple should belong to a cluster in a high probability $p(N(\varepsilon) \geq \eta)$.*

The determination of distance threshold ε and neighbor threshold η for clustering and outlier detection [27, 29] could also be interpreted by the probability $p(N(\varepsilon) \geq \eta)$. Referring to the Poisson distribution, i.e., in Figure 5, a larger ε or a smaller η leads to higher $p(N(\varepsilon) \geq \eta)$. The higher probability means more confident that the tuples satisfying the distance constraints indeed belong to a cluster. However, setting ε and η with too high probability would result in that those slightly low confidence outliers will not be detected as violations to the distance constraints. Consequently, by observing Figure 5, a distance threshold $\varepsilon = 3.0$ is preferred, where a limited number of data points in the left part of the blue line could be identified as outliers. It further determines a neighbor threshold $\eta = 18$, leading to a sufficiently high probability 0.99 of belonging to a cluster. Similar to Figure 5(a) over the Letter dataset, Figure 5(b) indicates that $\varepsilon = 5$ leads to too many outliers with limited neighbors, while $\varepsilon = 15$ fails to detect sufficient outliers given a

threshold of neighbors. A moderately large $\varepsilon = 10$ is preferred. Consequently, we have $\eta = 31$, referring to the high probability $p(N(\varepsilon) \geq \eta) = 0.99$, given the probabilistic distribution with $\varepsilon = 10$ in Figure 5(b).

2.2 Value Adjustment for Outlier Saving

An outlier tuple t_o does not meet the distance constraints of ε and η w.r.t. a tuple set r, i.e., $|r_\varepsilon(t_o)| < \eta$. To save the outlier for better clustering as mentioned in the Introduction, let t'_o be a *feasible adjustment* of t_o on some attribute values, having $|r_\varepsilon(t'_o)| \geq \eta$. That is, tuple t'_o after value adjustment is no longer outlying and should belong to a cluster with a high probability.

Intuitively, values are expected to be adjusted minimally, i.e., a lower *adjustment cost* $\Delta(t_o, t'_o)$. The rationale lies in two aspects. (1) Errors often occur minimally on only a fraction of attributes, since human or machines always try to avoid mistakes in practice [15]. For example, usually only one or several sensors are broken at a time among hundreds of sensors in a wind turbine, where each sensor corresponds to an attribute. (2) The value change is expected to be minimized, following the intuition that the adjustment should avoid losing information of the original data [23]. For instance, `RH10-0AG` is preferred as an adjustment of `RH1O-OAG`, rather than some arbitrary `RH1X-XAG`, given the smaller distance between letter `O` and digit `0` by the Needleman-Wunch measure [36, 37].

Definition 2 (Outlier Saving). *The problem of saving outlier t_o under DIStance constraints (ε, η) for Clustering (DISC) is to find the* optimal value adjustment t'_o *of t_o that meets the distance constraints $|r_\varepsilon(t'_o)| \geq \eta$ and has the minimum adjustment cost $\Delta(t_o, t'_o)$.*

In practice, there may exist multiple outlier tuples in a given dataset that do not meet the distance constraints for clustering. We split the dataset into two parts, r of non-outlying tuples and s of outliers. The non-outlying r satisfying the distance constraints are employed to save the outliers (violation tuples) in s one by one.

2.3 Hardness Analysis

Unfortunately, the problem of finding the optimal value adjustment under distance constraints for clustering is generally hard.

Theorem 1. *Given an outlier tuple t_o with distance constraints (ε, η) over r of m attributes and a constant δ, the problem of determining whether there exists a value adjustment t'_o of t_o such that $|r_\varepsilon(t'_o)| \geq \eta$ and $\Delta(t_o, t'_o) \leq \delta$ is NP-complete.*

Proof sketch. To prove the NP-hardness, we show a reduction from the Minimum Valid Outlier Descriptor (MVOD) problem [19]. Given a set $\mathbb{O}$ of outliers, a set $\mathbb{N}$ of normal points, a set $\mathbb{P}$ of protected status variables where each $p \in \mathbb{P}$, taking on a value from {0,1}, specifies the coverage/explanation of points in $\mathbb{O}$ and $\mathbb{N}$, and an integer $k \leq |\mathbb{P}|$, the problem is to find a subset $P' \subseteq \mathbb{P}$ such that $|P'| \leq k$ and P' is a valid outlier descriptor for $\mathbb{O}$ and $\mathbb{N}$, i.e., P' covers only outliers and none of the normal points. We show that there is a value adjustment t'_o with $|r_\varepsilon(t'_o)| \geq \eta$ and $\Delta(t_o, t_i) \leq \delta = k$ if and only if the MVOD problem has a solution $P' \subseteq \mathbb{P}$ such that $|P'| \leq k$ and P' is a valid outlier descriptor for $\mathbb{O}$ and $\mathbb{N}$. □

Recognizing NP-hardness of the problem, a straightforward idea is to enumerate the possible value adjustments and return the optimal one. By considering all the values in each attribute, the approach needs $O(d^m n)$ time, where d is the size of attribute domain, m is the number of attributes in R, and n is the number of tuples in r. That is, for a fixed number m of attributes, the problem becomes tractable. For fixing the other number n of tuples, whether the problem becomes tractable is still open. We leave this interesting problem as the future study.

3 DISC ALGORITHM

Rather than enumerating the possible values in attributes of an outlier tuple, in this section, we first show the lower and upper bounds of desired adjustments in Sections 3.1 and 3.2. The developed bounds are then utilized in pruning and approximation in Section 3.3. In particular, the upper bound itself can serve as an approximate solution. It is a constant-factor polynomial-time approximation in certain settings, referring these bounds, in Section 3.4.

3.1 Lower Bound

Let t_1 be the η-th nearest neighbor of t_o in r, for example, in a two-dimensional space as illustrated in Figure 3. The outlier tuple t_o has $\Delta(t_o, t_1) > \varepsilon$. **Intuitively**, to "reach" t_1 (become an ε-neighbor), t_o needs to adjust at least a distance of $\Delta(t_o, t_1) - \varepsilon$. It serves as a lower bound of any feasible adjustment of t_o.

Lemma 2. *Any feasible adjustment t''_o of t_o always has*

$$\Delta(t_o, t''_o) \geq \Delta(t_o, t_1) - \varepsilon,$$

where t_1 is the η-th nearest neighbor of t_o in r.

Proof. For any tuple t'_o having $\Delta(t_o, t'_o) < \Delta(t_o, t_1) - \varepsilon$, we show that it cannot be a feasible adjustment, having a number η of ε-neighbors.

Consider any tuple $t_2 \in r$ with $\Delta(t_o, t_2) \geq \Delta(t_o, t_1) > \varepsilon$. Referring to triangle inequality of the distance function, we have $\Delta(t'_o, t_2) \geq \Delta(t_o, t_2) - \Delta(t_o, t'_o) \geq \Delta(t_o, t_1) - \Delta(t_o, t'_o) > \varepsilon$.

Since t_o does not meet the distance constraints, there are at least a number $n - \eta + 1$ of tuples $t_2 \in r$ with $\Delta(t_o, t_2) \geq \Delta(t_o, t_1) > \varepsilon$ (including t_1). Referring to the aforesaid derivation, we have $\Delta(t'_o, t_2) > \varepsilon$ for these at least $n - \eta + 1$ of tuples t_2. In other words, tuple t'_o having $\Delta(t_o, t'_o) < \Delta(t_o, t_1) - \varepsilon$ is not a feasible adjustment. □

Example 2 (Example 1 continued). *Consider tuple t_{13} in Figure 2. Let ε =0.28 and η =2. The second nearest neighbor of t_{13} is t_{10}, having $\Delta(t_{13}, t_{10})$ =0.903. We compute a lower bound $\Delta(t_{13}, t_{10}) - \varepsilon$ =0.623. Any feasible adjustment, e.g., t'_{13} =(13, 807, 165), always has $\Delta(t_{13}, t'_{13})$ =0.838 no less than the lower bound 0.623.*

Let X denote the set of attributes that are not adjusted. We are interested in the bounds of all feasible adjustments with unadjusted values $t_o[X]$ on attributes X, i.e., adjusting only the attributes $R \setminus X$. (Such bounds are useful in pruning and approximating in Section 3.3, by traversing unadjusted attributes X rather than enumerating the huge body of values on attributes.)

Similar to the definition in Formula 4, let $r_\varepsilon(t_o[X])$ denote all the tuples in r whose distances to t_o on attributes X are $\leq \varepsilon$. For

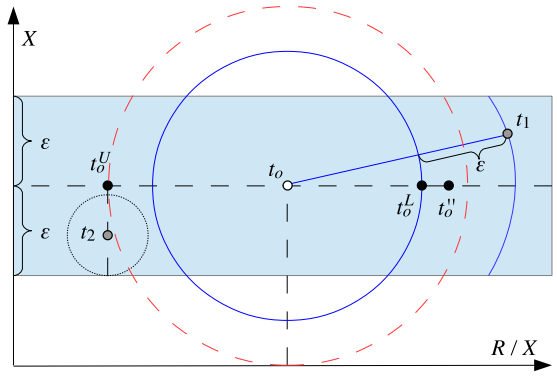

Figure 3: Lower (t_o^L) and Upper (t_o^U) bounds of any feasible adjustment (t_o'') with unadjusted attributes X. As defined in Proposition 3, t_1 is the η-th nearest neighbor of t_o in $r_\varepsilon(t_o[X])$ having $\Delta(t_o, t_1) > \varepsilon$. It determines the lower bound by t_o^L with adjustment cost no less than $\Delta(t_o, t_1) - \varepsilon$, i.e., the blue circle. Moreover, as defined in Proposition 5, $t_2 \in r_\varepsilon(t_o[X])$ is a tuple having a number η of $(\varepsilon - \Delta(t_o[X], t_2[X]))$-neighbors on attributes R. The little circle in black around t_2 denotes a number η of neighbors with distances $\leq \varepsilon - \Delta(t_o[X], t_2[X])$. It determines the upper bound by t_o^u, i.e., the red dashed circle.

instance, in a two-dimensional space as illustrated in Figure 3, it denotes all the tuples (points) in the shaded area.

Proposition 3. *Let t_1 be the η-th nearest neighbor of t_o in $r_\varepsilon(t_o[X])$ having $\Delta(t_o, t_1) > \varepsilon$. Any feasible adjustment t_o'' with unadjusted attributes X ($t_o''[X] = t_o[X]$) always has*

$$\Delta(t_o, t_o'') \geq \Delta(t_o, t_1) - \varepsilon.$$

Proof. For any tuple $t_o^\#$ with $t_o^\#[X] = t_o[X]$ and $\Delta(t_o, t_o^\#) < \Delta(t_o, t_1) - \varepsilon$, we show that it cannot be a feasible adjustment (having a number η of ε-neighbors).

First, since $t_o^\#[X] = t_o[X]$, any tuple $t \in r$ with $\Delta(t[X], t_o[X]) > \varepsilon$ cannot be a ε-neighbor of $t_o^\#$. Thereby, it is sufficient to consider $r_\varepsilon(t_o[X])$ for potential neighbors of $t_o^\#$.

Following the same line of proving Lemma 2, we can show that tuple $t_o^\#$ with $\Delta(t_o, t_o^\#) < \Delta(t_o, t_1) - \varepsilon$ cannot be a feasible adjustment with neighbors from $r_\varepsilon(t_o[X])$. □

Referring to Proposition 3, any feasible adjustment of t_o with unadjusted attributes X must have an adjustment cost no less than $\Delta(t_o, t_1) - \varepsilon$, where t_1 is the η-th nearest neighbor of t_o in $r_\varepsilon(t_o[X])$. That is, in Figure 3, any point with unadjusted attributes X and lying inside the blue solid circle is not possible to have a number η of ε-neighbors, and thus cannot be a feasible adjustment. The point t_o^L with $t_o^L[X] = t_o[X]$ serves as a lower bound of any feasible adjustment whose X value is the unadjusted $t_o[X]$.

In this sense, Lemma 2 is indeed a special case of Proposition 3, where $X = \emptyset$, i.e., all the attributes can be adjusted and we search t_1 in r (define $\Delta(t_i[X], t_j[X]) = 0$ for $X = \emptyset$).

3.2 Upper Bound

Recall that any tuple in r meets the distance constraints and thus could be a potential feasible adjustment. **Intuitively**, the one with the minimum distance to tuple t_o would serve as an upper bound of the minimum adjustment.

Lemma 4. *Given an outlier-free set r of tuples meeting distance constraints, there always exists a feasible adjustment t_o' of t_o having $\Delta(t_o, t_o') \leq \Delta(t_o, t_2)$, where $t_2 = \arg\min_{t \in r} \Delta(t_o, t)$.*

Proof. Obviously, t_2 is a feasible modification of t_o, which itself satisfies $\Delta(t_o, t_2) \leq \Delta(t_o, t_2)$. □

While the existing DORC [45] substitutes outliers directly by the nearest existing tuple, in this study, we consider the existing tuple t_2 as the upper bound of the optimal adjustment. In this sense, it is not surprising that DISC proposed in this paper is more precise than DORC, as illustrated in the experiments in Section 4.

Example 3 (Example 1 continued). *Consider tuple t_{13} in Figure 2 given $\varepsilon = 0.28$ and $\eta = 2$. The nearest neighbor of t_{13} is t_{12}, having $\Delta(t_{13}, t_{12}) = 0.846$. It serves as an upper bound. For instance, for $t_{13}' = (13, 807, 165)$, we have $\Delta(t_{13}, t_{13}') = 0.838 < 0.846$.*

Again, we extend the upper bound in Lemma 4 for feasible adjustments with unadjusted attributes X.

Proposition 5. *If there exists a tuple $t_2 \in r_\varepsilon(t_o[X])$ having a number η of $(\varepsilon - \Delta(t_o[X], t_2[X]))$-neighbors on attributes R (full-space), then t_o must have a feasible adjustment t_o'' with unadjusted attributes X ($t_o''[X] = t_o[X]$) and*

$$\Delta(t_o, t_o'') \leq \Delta(t_o, t_o^u),$$

where t_o^u is an upper bound adjustment having $t_o^u[X] = t_o[X]$, $t_o^u[R \setminus X] = t_2[R \setminus X]$.

Proof. For any $(\varepsilon - \Delta(t_o[X], t_2[X]))$-neighbor t_3 on attributes R (full-space) of t_2, it has $\Delta(t_2, t_3) \leq \varepsilon - \Delta(t_o[X], t_2[X])$. Referring to the aforesaid definition of t_o^u, we have $\Delta(t_2, t_o^u) = \Delta(t_o[X], t_2[X])$. According to triangle inequality of the distance function, it follows $\Delta(t_3, t_o^u) \leq \Delta(t_3, t_2) + \Delta(t_2, t_o^u) = \varepsilon$.

That is, all the η neighbors of t_2 with distances no greater than $\varepsilon - \Delta(t_o[X], t_2[X])$ are also ε-neighbors of t_o^u. In other words, t_o^u is a feasible adjustment and can serve as an upper bound. □

For instance, in Figure 3, suppose that t_2 has a number η of neighbors with distances $\leq \varepsilon - \Delta(t_o[X], t_2[X])$, i.e., neighbors of t_2 inside the black dotted circle. All these neighbors have distances to t_o^u no greater than ε, i.e., ε-neighbors of t_o^u. By finding such a t_2 with the minimum distance $\Delta(t_o[R \setminus X], t_2[R \setminus X])$ to t_o, the corresponding t_o^u is an upper bound. The minimum adjustment of t_o, whose X value is the unadjusted $t_o[X]$, must be inside the red dashed circle.

Indeed, if we define $\Delta(t_i[X], t_j[X]) = 0$ for $X = \emptyset$, Proposition 5 states that any tuple t_2 in r has a number η of ε-neighbors could be an upper bound of the optimal adjustment of t_o. We are interested in the one with the minimum distance to t_o, i.e., exactly Lemma 4. In other words, Lemma 4 is a special case of Proposition 5, which illustrates the rationale of Proposition 5.

3.3 Approximation with Bounds

As stated in Proposition 5, the upper bound t_o^u itself is a feasible adjustment (if exists). **Intuitively**, instead of enumerating the huge body of all values as possible adjustments, we may directly return this upper bound t_o^u as the approximate solution. Remarkably, the approximation performance is guaranteed under certain settings in Propositions 6 and 7, referring to the aforesaid bounds.

Algorithm 1 presents the pseudocode of approximation. The major steps are as follows.

Algorithm 1: DISC(r, X, t_o^*)

Input: r a set of non-outlying tuples,
X a set of unadjusted attributes,
t_o^* the currently known best solution of outlier tuple t_o
Output: t_o^* the near optimal adjustment of tuple t_o

/* lower bound */
1 t_1 := the η-th nearest neighbor of t_o in r;
2 **if** $\Delta(t_o, t_1) - \varepsilon \geq \Delta(t_o, t_o^*)$ **then**
3 **return** t_o^*; // Proposition 3
/* upper bound */
4 $r_2 := \{t \in r \mid \delta_\eta(t) \leq \varepsilon - \Delta(t_o[X], t[X])\}$;
5 **if** $|r_2| > 0$ **then**
6 $t_2 := \arg\min_{t \in r_2} \Delta(t_o[R \setminus X], t[R \setminus X])$;
7 $t_o^u[X] = t_o[X], t_o^u[R \setminus X] = t_2[R \setminus X]$;
8 **if** $\Delta(t_o, t_o^u) < \Delta(t_o, t_o^*)$ **then**
9 $t_o^* := t_o^u$; // Proposition 5
10 **for** *each* $A \in R \setminus X$ **do**
11 $t_o^* :=$ DISC($r, X \cup \{A\}, t_o^*$);
12 **return** t_o^*;

3.3.1 Enumerating Unadjusted Attributes X. To utilize and obtain more tight upper bounds in Proposition 5, we recursively enumerate various $X \subseteq R$ as unadjusted attributes, starting from $X = \emptyset$. To avoid redundant computations, the same attribute set X will be processed at most once. The upper bound t_o^u for each $X \subseteq R$ is computed. Among them, the minimum one is returned as the approximate adjustment.

3.3.2 Upper Bounds as Approximate Adjustment. Let t_o^* be the currently known best solution. To initialize t_o^* for $X = \emptyset$, we assign $t_o^* = \arg\min_{t \in r} \Delta(t_o, t)$, according to Lemma 4. With the recursive augmentation of the unadjusted attributes X, t_o^* is gradually improved. If t_o^u of the present X in Proposition 5 has adjustment cost lower than the currently known best solution t_o^*, the lower cost t_o^u is returned as the new best solution thus far (Line 8).

3.3.3 Lower Bounds for Pruning. Referring to the lower bound of feasible adjustments with unadjusted attributes X in Proposition 3, if the lower bound, $\Delta(t_o, t_1) - \varepsilon$, is already higher than the currently known best solution t_o^*, all the adjustments with unadjusted X can be pruned. That is, the recursive augmentation on X stops (Line 2).

Example 4 (Example 1 continued). *Algorithm 1 begins with $X = \emptyset, t_o = t_o' = t_{13}, t_o^* = t_{12}$. Let $A =$ Time be the unadjusted attribute in Line 11 in Algorithm 1. In the next recursion, it computes $r_2 = \{t_{12}\}$. We have $\Delta(t_o, t_o^u) < \Delta(t_o, t_o^*)$, where*

$$t_o^u[Time] = t_o[Time] = 13,$$

$$t_o^u[Longitude, Latitude] = t_{12}[Longitude, Latitude] = (838, 165).$$

Referring to Line 7, t_o^u =(13, 838, 165) is assigned as the new currently known best solution. Finally, a result (13, 807, 156) will be returned.

The approximation algorithm considers only the unadjusted attributes X in R rather than enumerating all the possible values on all the attributes as adjustments. For the brute-force enumeration starting from $X = \emptyset$, the algorithm runs in $O(2^m n)$ time. It is more efficient than $O(d^m n)$ of the straightforward idea presented at the end of Section 2.3. Moreover, as discussed in the Introduction, in some scenarios, the user may be confidence on certain attributes. For instance, a wind turbine would be automatically switched off, if there are more than κ sensors broken. For this case, one knows confidently at most κ dirty attribute values in a tuple, and does not trust an adjustment on more than κ attributes. We may only return those adjustments with at least $m - \kappa$ unadjusted attributes. That is, we can generate bounds only for X with $|X| \geq m - \kappa$. Rather than initializing $X = \emptyset$, Algorithm 1 runs for each $X \subseteq R$ having $|X| = m - \kappa$. By avoiding the redundant computation on the same X, the time complexity of approximation becomes $O(m^{\kappa+1} n)$.

3.4 Approximation Performance

Let t_o' be the approximate adjustment returned by Algorithm 1, and t_o^* be the optimal solution. We utilize again the lower and upper bounds to derive the approximation factor. It is notable that given distance metrics with discrete distance values, Algorithm 1 is indeed a constant factor approximation (Proposition 7).

Proposition 6. *For any tuple t_o with $\Delta(t_o, t_2) \geq c\varepsilon$, where $t_2 = \arg\min_{t \in r} \Delta(t_o, t)$ and $c > 1$, we have*

$$\frac{\Delta(t_o, t_o')}{\Delta(t_o, t_o^*)} \leq \frac{c}{c-1}.$$

In Proposition 6, $t_2 = \arg\min_{t \in r} \Delta(t_o, t)$ denotes the nearest (non-outlying) neighbor of the outlier t_o. It is often distant from the outlying t_o. For instance, in Figure 1(b), given $\varepsilon = 0.1$, we have $\Delta(t_o, t_2) \geq 0.5$ for the outliers, i.e., $c = 5$. The approximation factor is thus $\frac{\Delta(t_o, t_o')}{\Delta(t_o, t_o^*)} \leq \frac{c}{c-1} = 1.25$. Indeed, the more the outlier t_o is distant from the non-outlying tuples in r, i.e., larger c, the closer the approximate adjustment is to the optimal solution, a smaller approximation factor.

Proposition 7. *If the unit of distance value is 1, we have approximation factor*

$$\frac{\Delta(t_o, t_o')}{\Delta(t_o, t_o^*)} \leq \varepsilon + 1,$$

where distance threshold ε is an integer as well.

In Proposition 7, we consider the distance metrics with discrete distance values. It is often employed in evaluating string values, such as edit distance [36], where $\Delta(t_1, t_2) > \varepsilon$ implies $\Delta(t_1, t_2) \geq \varepsilon + 1$ and the distance threshold ε is also an integer. For example, as discussed in Section 1.1, typos in zip code RH10-0AG, where letters O should be digits 0, have edit distance equal to 2. With such a distance metric, the approximation factor is given by $\frac{\Delta(t_o, t_o')}{\Delta(t_o, t_o^*)} \leq \varepsilon + 1$.

Table 1: Real data sets used in experiments

Dataset	#tuple	#attribute	#class	#outlier	|domain|
Iris [1]	150	4	3	15	23.25
Seeds [3]	210	7	4	12	182.3
WIFI [4]	2000	7	4	156	42.14
Yeast [5]	1299	8	4	39	36.63
Letter [2]	20000	16	26	1920	16
Flight [7]	200000	3	5	19920	1272
Spam [6]	4601	57	2	457	32.81
GPS	8125	3	3	837	3844
Restaurant [8]	864	5	752	86	864

4 EXPERIMENTS

The experimental evaluation proposes to validate two aspects. (1) Could outlier saving truly improve clustering? i.e., application performance. (2) Is the adjustment of potential errors in outliers close to the truth? a.k.a. cleaning accuracy.

4.1 Experimental Settings

Table 1 lists the real datasets with outliers. We quantify the number of outliers detected by DBSCAN. Small clusters in the Yeast dataset are identified as outliers by DBSCAN in ground truth. To avoid confusing, we do not consider such small clusters in evaluation.

4.1.1 Clustering Evaluation. We employ various clustering algorithms such as the classical DBSCAN [21] and K-Means [26]. While DBSCAN handles both clustering and outliers, K-Means- [13] also extends K-Means by considering k clusters and l outliers. To address the unbalanced clusters and outliers, CCKM [43] introduces an auxiliary outlier cluster. SREM [40] proposes to learn hybrid models to reduce the sensitivity of initial points. KMC [14] extracts a small kernel set for approximating K-Means. For the clustering methods such as K-Means with the cluster number as the input, we set the number of clusters the same as classes, presented in Table 1. For the other methods like DBSCAN, the number of returned clusters may be different from the given classes.

Clustering accuracy is measured by F1-score, NMI and ARI [38]. The F1-score [33] is given by $\text{F1-score} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$, $\text{Precision} = \frac{TP}{TP+FP}$, $\text{Recall} = \frac{TP}{TP+FN}$, where TP is the number of true positives, i.e., the number of pairs of points that are clustered together in the predicted partition and in the ground truth partition, FP is the number of false positives, i.e., the number of pairs of points that are clustered together in the predicted partition but not in the ground truth partition, and likewise FN is false negative.

4.1.2 Classification Evaluation. Note that besides clustering, other applications also benefit by saving outliers. To evaluate the classification application over the data without/with outlier saving, we employ the scikit-learn implementation[9] of decision tree [20]. Classifiers are trained with default parameters. We use 5-fold cross-validation. Classification performance is measured by F1-score.

In particular, we do not use the cluster value in the aforesaid clustering experiment as the class label in this classification experiment. Instead, only outlier saving (but not clustering) is employed in the classification setup. Specifically, we train the existing classifiers over the datasets with/without outlier saving. While the training datasets are different (under various outlier handling methods), the prediction of class labels is done by the aforesaid trained classifiers. Therefore, the motivation of the experiment is to show that outlier saving (but not clustering) is beneficial to the downstream tasks such as classification and record matching.

4.1.3 Record Matching Evaluation. Similar to classification, another record matching application could also be improved by outlier saving. A rule-based record matching method [24] is implemented, where two tuples having similar values on all attributes are considered as matched. By "similar", we mean the normalized n-gram similarity of two values larger than a threshold (0.7). The record matching accuracy is measured by F1-score.

4.1.4 Compared Methods. The proposed DISC for outlier saving is compared to the following existing methods that may clean the errors in outliers. DORC [45] shares the same parameters of our proposal, i.e., distance threshold ε and neighbor threshold η. The parameters of ERACER [34] are histograms for convolution and coefficients for regression, which can be learned directly from the data as suggested in [34]. Holistic [17] uses denial constraints to steer data cleaning. The constraints are discovered from the data [16]. HoloClean [41] have parameters of its probabilistic model, which are computed using empirical risk minimization (ERM). As suggested in [41], variables that correspond to clean cells are treated as labeled examples to learn the parameters of the model.

The choosing of parameters is independent of method. That is, different methods may use different parameter settings for the highest cluster accuracy. As illustrated in Figures 4, 8 and 10, we search the settings of distance threshold ε and neighbor threshold η with the best performance for DORC and DISC. For ERACER, Holistic and HoloClean, as aforesaid, their parameters are directly learned/computed from data.

4.2 Application of Saving Outliers

This experiment demonstrates how clustering as well as other applications are improved by saving outliers.

4.2.1 Clustering. Table 2 presents the DBSCAN clustering results over the raw data without outlier saving, the data with outlier saving by the proposed DISC, and the data cleaned by various existing methods DORC, ERACER, HoloClean and Holistic. The results on F1-score, NMI and ARI measures are generally similar. As shown, the clustering accuracy after outlier saving by DISC is higher than directly performing over the raw data. The results verify the rationale of outlier saving for better clustering.

The general purpose data cleaning methods do not consider the improvement of the downstream applications. For instance, as illustrated in Figure 1, while the data cleaning methods modify the data to meet the cleaning criteria, the modified data may not always be accurate and thus not help in improving clustering (one cluster still splits into two). Indeed, the methods in data cleaning may modify all the outliers either introduced by errors (dirty outliers) or simply true abnormal behaviors (natural outliers). For example, as introduced in Section 1.2, t_1 in Figure 1, t_{29} and t_{30} in Figure 2 are natural outliers *not* introduced by errors but modified by the

Table 2: Clustering over raw data without / with outlier saving or data cleaning by various approaches

Data	NMI (DBSCAN)						ARI (DBSCAN)					
	Raw	DISC	DORC	ERACER	HoloClean	Holistic	Raw	DISC	DORC	ERACER	HoloClean	Holistic
Iris	0.6123	**0.6947**	0.6611	0.5023	0.6612	0.6426	0.5615	**0.6511**	0.6104	0.4880	0.6308	0.6052
Seeds	0.5098	**0.5908**	0.5228	0.5076	0.5098	0.5021	0.4242	**0.5220**	0.4538	0.4016	0.4299	0.4381
WIFI	0.7245	**0.7891**	0.6778	0.6536	0.7349	0.7310	0.5990	**0.6975**	0.5730	0.5296	0.5933	0.5655
Yeast	0.2728	**0.3237**	0.2794	0.2936	0.2628	0.2416	0.2573	**0.2914**	0.2391	0.2197	0.1894	0.1793
Letter	0.6696	**0.7153**	0.6725	0.5927	0.6194	0.6272	0.5674	**0.6849**	0.6285	0.4837	0.5048	0.5367
Flight	0.5245	**0.5721**	0.5425	0.5199	0.5238	0.5172	0.5151	**0.5715**	0.5315	0.4948	0.5061	0.5017
Spam	0.6618	**0.7059**	0.6961	0.6231	0.6602	0.6552	0.5918	**0.6527**	0.6258	0.5523	0.5893	0.5819
GPS	0.6089	**0.6721**	0.6402	0.5504	0.5782	0.5828	0.5825	**0.6293**	0.5709	0.5014	0.5376	0.5483

Data	F1-score (DBSCAN)						Time cost (s) (DBSCAN)					
	Raw	DISC	DORC	ERACER	HoloClean	Holistic	Raw	DISC	DORC	ERACER	HoloClean	Holistic
Iris	0.7784	**0.8508**	0.7862	0.7003	0.7461	0.7509	0.0009	0.0735	0.0153	0.0316	0.1813	1.0281
Seeds	0.7054	**0.7963**	0.6388	0.6312	0.7054	0.6813	0.0012	0.0945	3.5618	0.0527	0.2371	3.0393
WIFI	0.8441	**0.9089**	0.7742	0.7222	0.8544	0.8517	0.0112	0.3235	23.613	0.4692	0.2251	25.391
Yeast	0.6453	**0.7162**	0.6551	0.5700	0.5548	0.5417	0.0022	0.2903	19.275	0.2866	0.2765	18.256
Letter	0.7377	**0.7739**	0.7260	0.6285	0.6529	0.6812	0.0353	1.9921	844.76	0.7727	0.3056	70.375
Flight	0.7157	**0.7487**	0.7239	0.6936	0.6498	0.6706	0.0563	1.8911	1583.2	0.8327	0.2659	150.13
Spam	0.7509	**0.8305**	0.7722	0.6952	0.7959	0.8174	0.0236	5.0317	2646.3	1.1201	0.2176	125.42
GPS	0.6715	**0.7331**	0.7078	0.5845	0.6012	0.6304	0.0393	1.6139	1267.9	0.8173	0.1903	130.52

Table 3: Clustering by various methods over raw data without / with outlier saving

F1-score	DBSCAN		K-Means		K-Means-		CCKM		SREM		KMC	
Data	Raw	DISC	Raw	DISC	Raw	DISC	Raw	DISC	Raw	DISC	Raw	DISC
Iris	0.7784	**0.8508**	0.8206	**0.8677**	0.8375	**0.8685**	0.8253	**0.8712**	0.8307	**0.8919**	0.8213	**0.8567**
Seeds	0.7054	**0.7963**	0.5682	**0.6564**	0.5634	**0.6611**	0.5645	**0.6804**	0.6388	**0.7265**	0.5455	**0.6436**
WIFI	0.8441	**0.9089**	0.9152	**0.9247**	0.8964	**0.9186**	0.9137	**0.9226**	0.9279	**0.9391**	0.9085	**0.9287**
Yeast	0.6453	**0.7162**	0.6668	**0.6781**	0.6684	**0.7126**	0.6728	**0.7458**	0.7078	**0.7605**	0.6515	**0.6991**
Letter	0.7377	**0.7739**	0.6318	**0.6419**	0.6258	**0.6317**	0.6320	**0.6778**	0.6857	**0.7285**	0.6352	**0.6735**
Flight	0.7157	**0.7487**	0.6517	**0.6946**	0.6676	**0.6803**	0.6712	**0.7096**	0.7065	**0.7682**	0.6514	**0.6859**
Spam	0.7509	**0.8305**	0.7626	**0.8164**	0.7598	**0.8061**	0.7519	**0.8275**	0.7929	**0.8738**	0.7565	**0.8168**
GPS	0.6715	**0.7331**	0.6212	**0.7563**	0.5807	**0.7349**	0.6742	**0.7488**	0.6774	**0.7819**	0.6077	**0.7678**

data cleaning methods. Therefore, as also indicated in [31], data cleaning does not necessarily improve the quality of downstream ML models. It explains why the state of the art methods in data cleaning may perform worse on downstream tasks after cleaning.

Our DISC proposes to distinguish the dirty and natural outliers. As mentioned in Section 1.2, we choose to only adjust minimally a few attributes of the dirty outliers (introduced by errors such as t_{13} and t_{24} in Figure 2). In contrast, we leave the natural ones (of true abnormal behaviors, e.g., t_{29} and t_{30} in Figure 2) unchanged, which need to adjust a large number of attributes to become inliers.

For datasets with larger domain sizes of attributes, such as Flight and GPS illustrated in Table 1, the corresponding performances are not high in Table 2. The reason is that with more candidates in the domain of attributes, it is more difficult to determine the right adjustment for outliers.

As illustrated in Table 2, while the differences on time costs over small datasets are not clear, the density-based DORC and the

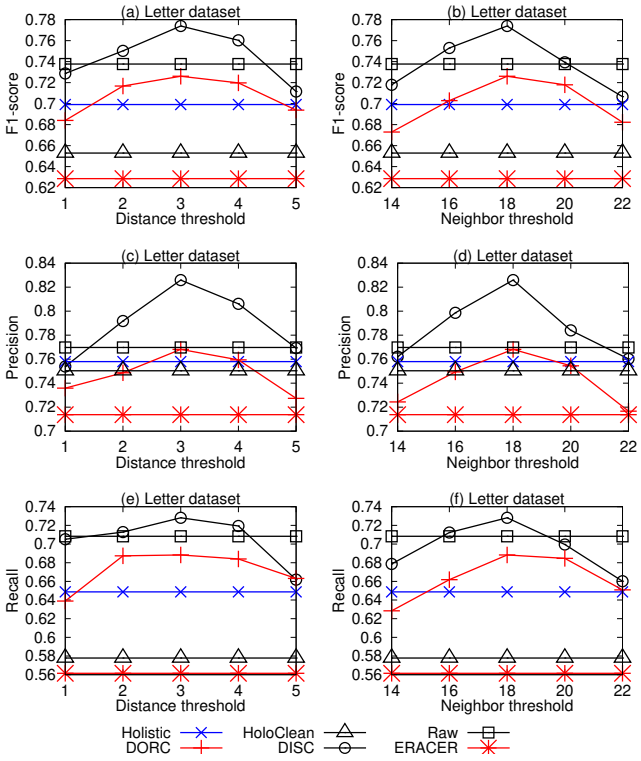

Figure 4: Clustering over raw data without/with outlier saving or data cleaning, where m=16, n=1000, (a) η=18, (b) ε=3

Table 4: Performance of parameter determination

Data	Sample		Time (s)		ε, η			Clustering F1-score		
	Rate	Tuples	DISC	DB	DISC	DB	Optimal	DISC	DB	Optimal
Letter	1%	200	0.015	0.014	3, 18	0.41, 24	3, 18	0.774	0.319	0.774
Letter	10%	2k	0.167	0.168	3, 18	0.42, 24	3, 18	0.774	0.326	0.774
Letter	100%	20k	1.494	1.514	3, 18	0.43, 24	3, 18	0.774	0.335	0.774
Flight	0.1%	200	0.066	0.061	10, 31	60.1, 240	10, 35	0.732	0.291	0.749
Flight	1%	2k	0.612	0.615	10, 31	62.7, 240	10, 35	0.732	0.294	0.749
Flight	100%	200k	53.202	53.172	10, 31	62.2, 240	10, 35	0.732	0.293	0.749

constraint-based Holistic show significantly higher time costs in larger datasets, such as Letter. Our proposal DISC has time performance comparable to the statistics-based ERACER and HoloClean.

It is worth nothing that our proposal of outlier saving is complementary to the existing clustering (as well as classification and record matching) methods, i.e., first apply adjustment for outlier saving, and then perform clustering over the adjusted data. As shown in Table 3, the clustering performance is more or less improved by first applying outlier saving (the DISC columns), compared to directly conducting clustering over the raw data without outlier saving (the corresponding Raw columns). The clustering algorithm with better performance over raw data, e.g., SREM+Raw, generally shows higher accuracy as well after outlier saving, i.e., SREM+DISC.

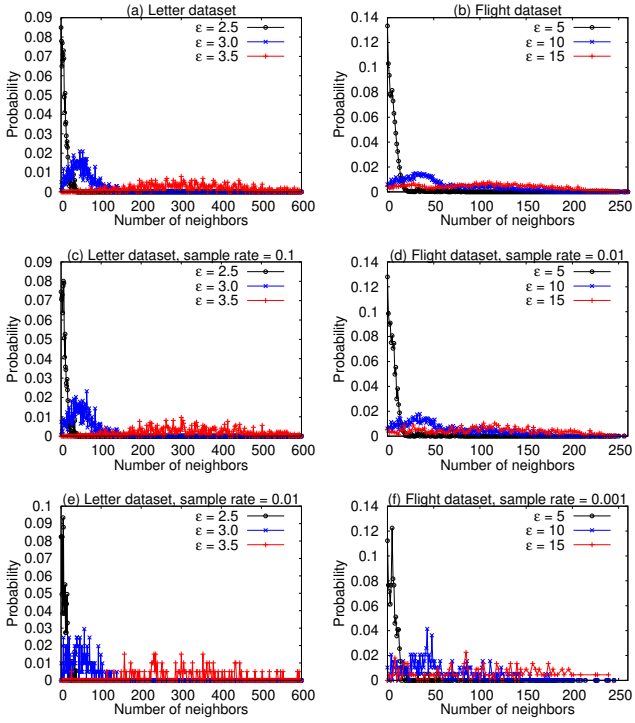

Figure 5: Distribution on the number of ε-neighbors over (a) Letter and (b) Flight, with different sampling rates

4.2.2 Parameter Determination. As discussed at the end of Section 2.1, the determination of parameters η and ε in distance constraints is to observe the probability $p(N(\varepsilon) \geq \eta)$ in the distribution, e.g., in Figure 5(a) over the Letter dataset.

First, let us present the F1-score, precision and recall of clustering under various ε and η settings in Figure 4. These parameters also affect DORC which considers the number of neighbors in density-based clustering and cleaning. The results of other data cleaning approaches ERACER, Holistic and HoloClean are irrelevant to these parameters and reported as baselines.

For the distance threshold ε, as illustrated in Figure 5(a), when ε is small, e.g., $\varepsilon = 2.5$, the probability of a tuple having limited neighbors is high. In other words, a large number of tuples would have insufficient neighbors. As a result, many tuples without errors are adjusted, i.e., over-change, and the clustering accuracy is low in Figure 4(a). On the other hand, when ε is large, such as $\varepsilon = 3.5$, most tuples have a large number of neighbors as observed in Figure 5(a). Only a few tuples will be adjusted, and thus the overall clustering accuracy is low. Consequently, as discussed at the end of Section 2.1, a moderately large $\varepsilon = 3$ is preferred to detect and clean dirty outliers in a high probability.

For the neighbor threshold η, again, referring to Figure 5(a), for a fixed $\varepsilon = 3$, a larger threshold η on the number of neighbors means that more tuples will have insufficient neighbors and thus be adjusted. Similar to Figure 4(a), a large η leads to over-change in Figure 4(b), while a small η fails to identify erroneous tuples for adjustment. Consequently, a threshold $\eta = 18$ is selected with $p(N(\varepsilon) \geq \eta) = 0.99$ as discussed at the end of Section 2.1.

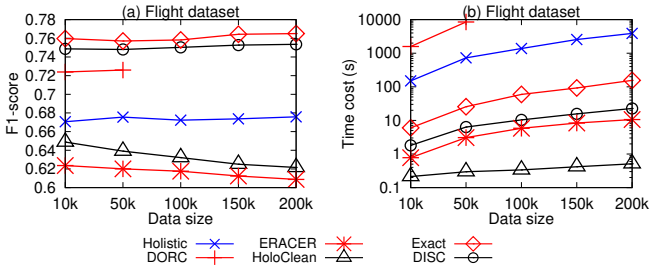

Figure 6: Scalability on the number of tuples n, where $m = 3$, $\varepsilon = 10$ and $\eta = 35$

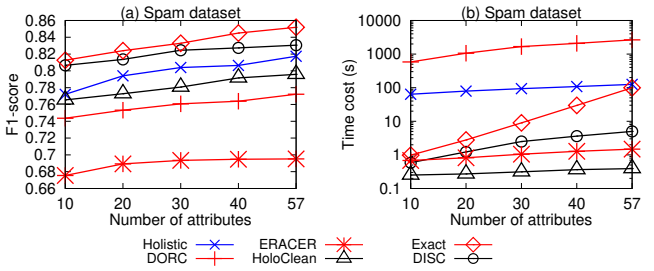

Figure 7: Scalability on the number of attributes m, where $n = 5000$, $\varepsilon = 5$ and $\eta = 10$

Moreover, we show that sampling is applicable to obtain the probability distribution in Figure 5(a). As presented in Figure 5(c), with a sampling rate 0.1, i.e., counting the ε-neighbors for only 10% tuples, a similar distribution is still observed. The corresponding time cost (for determining parameters) is significantly reduced in Table 4. Indeed, by observing about 2k samples, we could clearly obtain the distribution for parameter determination, which is also observed in another dataset Flight in Figure 5(d), i.e., a sampling rate 0.01 over 200k tuples.

Finally, we consider an existing parameter determination method (DB) based on Normal distribution [27, 29]. As shown in Table 4, time cost of DB with Normal distribution estimation is similar to our DISC using Poisson distribution. The corresponding F1-score of clustering, using the parameters ε and η determined by our proposal, is higher, almost close to the optimal one (found by testing various ε and η combinations in Figure 4).

4.2.3 Scalability. We evaluate the impact of large data sets, Flight and Spam, in Figures 6 and 7. The results are generally similar to Table 2 over other datasets.

Figure 6 reports clustering F1-score and the corresponding time cost on dataset Flight with up to 200k tuples. The results are generally similar to Table 2. While our DISC scales well as the statistics-based ERACER and HoloClean, the density-based DORC cannot obtain a result in more than one hour with data sizes larger than 50k.

Figure 7 considers dataset Spam with up to 57 attributes. Again, our DISC shows good scalability w.r.t. the number of attributes, thanks to the effective approximation in Section 3.4.

Moreover, we implement the exact algorithm by considering the values in each attribute as possible adjustments, introduced at the end of Section 2.3 after analyzing the intractability in Theorem 1. As presented in Figures 6 and 7, while the Exact algorithm shows better clustering F1-score than the approximate DISC, the corresponding time cost is higher. It illustrates the trade-offs between the improvement in time performance and the loss in accuracy by the approximation algorithm DISC.

It is not surprising that the time cost of the Exact algorithm increases proportionally as the approximate DISC in Figure 6(b), given its time complexity $O(d^m n)$, i.e., linear w.r.t. the number of tuples. On the other hand, the time cost of the Exact algorithm increases exponentially in the number of attributes m, as shown in Figure 7(b). It also illustrates the boundaries in terms of resources when the Exact method is applied to large datasets, i.e., on a larger number of attributes.

Table 5: Classification over raw data without / with outlier saving or data cleaning by various approaches

Data	F1-score (Decision Tree)					
	Raw	DISC	DORC	ERACER	HoloClean	Holistic
Iris	0.9357	**0.9576**	0.9470	0.8605	0.9477	0.9262
Seeds	0.8407	**0.9687**	0.8541	0.8189	0.8541	0.8272
WIFI	0.9598	**0.9777**	0.9675	0.9646	0.9672	0.9634
Yeast	0.8946	**0.9152**	0.8972	0.86134	0.8523	0.8372
Letter	0.9345	**0.9421**	0.9381	0.9053	0.8923	0.8789
Flight	0.9207	**0.9356**	0.9317	0.9068	0.9132	0.9171
Spam	0.9033	**0.9296**	0.9178	0.8969	0.9068	0.9015

4.2.4 Classification. In addition to clustering, Table 5 presents the classification results performed over the raw data without outlier saving and the data with outliers saved by DISC and cleaning by various approaches. The results are generally similar to the clustering in Table 2. It is not surprising that dirty data would also distract classifier training while saving outliers helps. DISC with the best clustering performance again reaches the highest classification accuracy, in all the experiments.

Inaccurate cleaning may mislead not only clustering but also other downstream tasks [31] such as the training and scoring of classifiers. Therefore, as illustrated in Table 5, the general purpose data cleaning methods may show worse classification results in some tests compared to the Raw data without cleaning. By precisely saving outliers introduced by errors but not true abnormal behaviors, our DISC helps in improving the training and scoring of classifiers as well.

4.2.5 Record Matching. Figure 8 reports the record matching results over again the raw data without outlier saving and the data with outlier saving / data cleaning. The results may vary under different distance and neighbor threshold settings. Our DISC again significantly improves the matching accuracy, compared to matching over the raw data without outlier saving. It is analogous to the improvement of clustering and classification in Tables 2 and 5, respectively.

4.3 Accuracy of Saving Outliers

As discussed in Section 5, the existing Subspace Separability Explanation (SSE) [35] identifies the attributes of an outlier that show

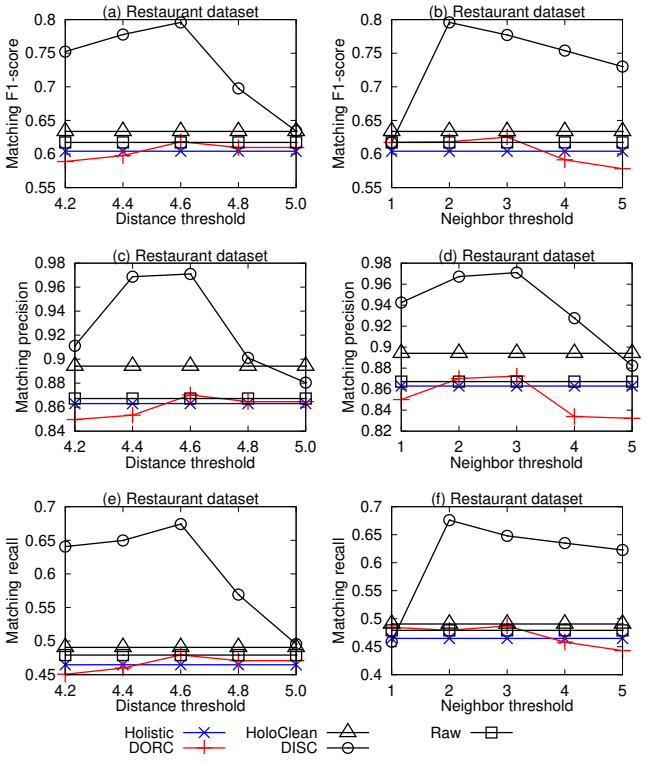

Figure 8: Record matching over the raw data without / with outlier saving or data cleaning by various methods, where $m = 5, n = 864$, (a) $\eta = 3$, (b) $\varepsilon = 4.6$ (ERACER supports only numerical values and does not apply)

separability, to explain why the tuple is outlying. However, SSE does not indicate how to save and utilize the outliers.

Nevertheless, to show that the value adjustment for outlier saving is indeed reasonable, we compare the attributes with values adjusted by our DISC and the attributes showing separability as explained by SSE. Let T be the set of attributes with errors in a tuple and P be the set of attributes adjusted by DISC, or returned by SSE. Following the same line of [35], we consider $\text{Jaccard}(T, P) = \frac{|T \cap P|}{|T \cup P|}$. The larger the Jaccard index is, the more accurate the attributes are explained by SSE / adjusted by DISC.

We consider the GPS dataset, where both dirty and natural outliers are observed and labeled. As illustrated in Figure 9(a), the rates of dirty and natural outliers are similar (about 0.1). That is, about 9% points are dirty outliers with errors in some attributes, while other 10% points are natural outliers with distinct Time, Longitude and Latitude values (occasionally collected for device testing in different time at various places).

Moreover, Figure 10 presents the accuracy of explanations on attributes for outliers with randomly injected errors. Since the adjustments on values are more strong evidence of being outliers than simply the separability on attributes, the accuracy of attribute explanations by DISC is a bit higher than the existing SSE. The results demonstrate as well the rationale of our DISC. In addition, we also report the attributes that are cleaned by various approaches.

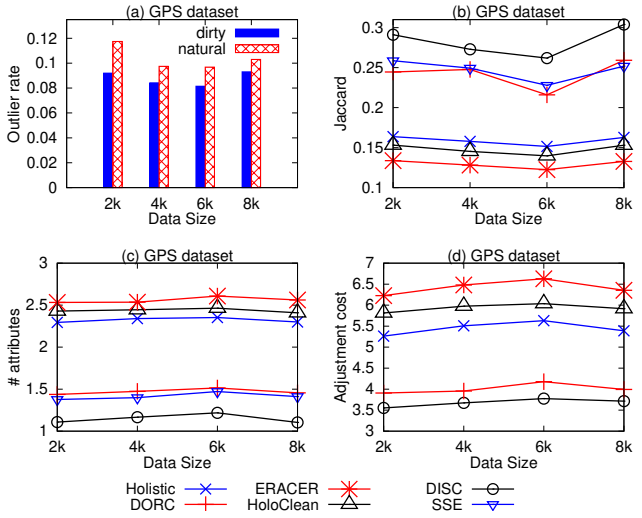

Figure 9: Accuracy of attribute adjustment/explanation for outliers where $m = 4$, $\eta = 3$, $\varepsilon = 15$

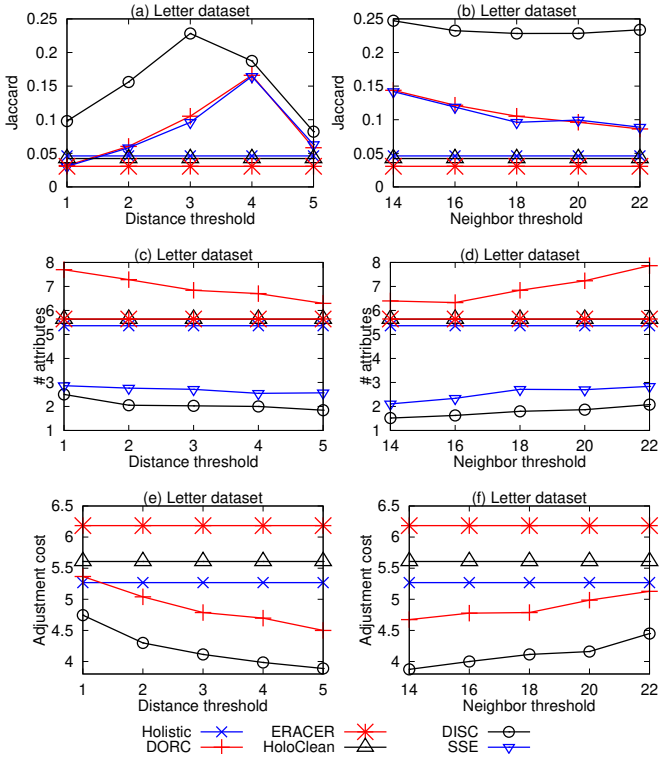

Figure 10: Accuracy of attribute adjustment/explanation for outliers where $n = 1000$, $m = 10$, (a) $\eta = 18$, (b) $\varepsilon = 3$

Since we randomly replace the values in attributes, it is not guaranteed to be separable attributes after introducing synthetic errors. Indeed, all the methods will not detect non-separable attributes (Jaccard=1 is probably unlikely). In this sense, while non-separable attributes are useless in evaluation, it does not hurt the comparison of approaches in identifying other separable attributes.

Figures 10 (c) and (d) report the number of modified attributes. Methods like HoloClean modify about 6 attributes among 10. The reason is that data cleaning methods follow the minimum change idea, and avoid excessively modifying all the attributes.

We also present the magnitude of the adjustment, i.e., the adjustment cost $\Delta(t_o, t'_o)$ in Definition 2 on how the adjusted t'_o is distant from the original t_o, in Figures 10 (e) and (f). As shown, the adjustment costs of data cleaning approaches such as HoloClean are high, i.e., over-changed. Therefore, it is not the case that they modify all attributes just a little.

5 RELATED WORK

Outlier Explanation. Given an outlier detected by existing methods, the outlier explanation [28, 35] identifies a set of attributes (subspace) in which the given outlier shows separability from the inliers. Unfortunately, it fails to explain how the values should be adjusted to become non-separable in the attributes. Therefore, the outlier explanations cannot be directly utilized in improving clustering. Our outlier saving method DISC further indicates the potential values the attributes should be adjusted to so that it is no longer outlying and thus improves clustering.

Sarawagi [44] uses the values approximated by other (parent) tuples to explain errors. The approach explains only the problems in aggregates but not for outliers, and thus is not directly applicable in this study. Similarly, Daniel et al. [18] study data quality-aware reports in business intelligence (BI) applications. Again, the proposal manages quality metadata for BI applications but not outliers.

Clustering and Cleaning. To perform clustering over noisy data, DBSCAN [21] investigates the number η of neighbors whose distances are no greater than the distance threshold ε. It considers a tuple as the core point in a cluster if at least η neighbors lie in a distance no greater than ε. Variations are derived under this intuition for clustering such as OPTICS [10] and AMTICS [42]. In this study, we use a similar distance-based notation as constraints to detect and save outliers, referring to the rationale of Poisson process in Section 2.1.2.

In addition to the core points with a number η of ε-neighbors, DBSCAN [21] further categorizes the points (tuples) without sufficient neighbors into (density-)reachable points and outliers. Only those tuples having ε-neighbors less than η and not the ε-neighbor of any core point will be detected as outliers. Unlike K-Means [26] that directly clusters all points including outliers, DORC [45] proposes to clean the noises by tuple substitution for clustering. For instance, in Figure 2(b), t_{24} is substituted by the existing observation t_{21}, denoted by blue triangle/line. All the attributes (Time, Longitude, Latitude) are substituted, i.e., over-changed. Therefore, the experiments in Section 4 show that our proposed DISC has better performance than DORC [45].

The novelty of our proposal, compared to the EM method [40], is the consideration of both the distance constraints for clustering and the minimum change for cleaning. In particular, maximizing the number of neighbors may excessively change the values of an outlier to the densest cluster, which is distant from and not related to the original tuple. For instance, in Figure 1(b), the red points (errors) may be mistakenly moved to the densest cluster C1 (black dots).

General Purpose Data Cleaning. Rather than targeting on saving outliers for better clustering, data cleaning techniques have been widely studied for general purpose and can be used to clean the errors in outliers. Integrity constraints are often employed to advise data cleaning [22]. In this study, we consider denial constraints (DC) based Holistic cleaning [17] as the competitor since DC can cover the semantics of several other constraint types, such as functional dependencies [30] or conditional functional dependencies [12]. Unfortunately, precise data dependencies may not always be available [11]. By declaring a very weak constraint so that it holds in the dataset, an error tuple such as t_{13} in Figure 2(a) is not identified with any violation and thus will not be cleaned. That is, the denial constraint specifies a constraint on the walking speed of a person. The small error on longitude value in t_{13} does not violate the constraint. Consequently, with insufficient detection, the accuracy of cleaning under denial constraints is low in Section 4.

Statistical approaches employ statistic models for cleaning, e.g., the linear regression model in ERACER [34]. SCARE [46] proposes to maximize the data likelihood w.r.t. the statistical model, which does not show better performance than ERACER [34] as reported in the paper. Since the statistical model is iteratively learned from the dirty and partially cleaned data, the data could be over-changed to fit the model. Moreover, owing to data heterogeneity and sparsity, the learned statistical models are often inaccurate [34]. Therefore, statistical method ERACER, as well as HoloClean [41] using both DC rules and statistical results, may not perform.

It is worth noting that these data cleaning methods are proposed for general purpose without particular consideration for clustering in advance. In contrast, our proposal aims to save outliers for better clustering under the distance constraints in the first place. Hence, the accuracy of clustering with the proposed outlier saving is higher than the existing methods in the experiments in Section 4.

6 CONCLUSION

In this paper, we propose to adjust the values of outliers under distance constraints for better clustering. The distance constraints are declared referring to the Poisson process of nearest neighbor appearance. That is, if a tuple belongs to a cluster, it should have a high probability (e.g., 0.99) to observe a certain number of neighbors. The outlier saving is thus to minimally adjust the outlier values in order to meet the distance constraints for clustering. To find the minimum adjustment, we (1) analyze its NP-hardness, (2) obtain the lower and upper bounds of optimal solutions, and (3) devise pruning and approximation techniques. It is notable that the algorithm is a constant-factor polynomial-time approximation in certain settings (Proposition 7). Experiments on real datasets show that the clustering (as well as classification and record matching) performances are clearly improved after saving the outliers. Our proposal is more effective than applying the general purpose data cleaning methods which do not consider the clustering target in advance.

Acknowledgement

This work is supported in part by the National Key Research and Development Plan (2019YFB1705301), the National Natural Science Foundation of China (62072265, 61572272, 71690231), and the MIIT High Quality Development Program 2020.

REFERENCES

[1] http://archive.ics.uci.edu/ml/datasets/iris.
[2] http://archive.ics.uci.edu/ml/datasets/letter+recognition.
[3] http://archive.ics.uci.edu/ml/datasets/seeds.
[4] http://archive.ics.uci.edu/ml/datasets/wireless+indoor+localization.
[5] http://archive.ics.uci.edu/ml/datasets/yeast.
[6] https://archive.ics.uci.edu/ml/datasets/spambase.
[7] https://figshare.com/articles/dataset/flights+csv/9820139.
[8] http://www.cs.utexas.edu/users/ml/riddle/data.html.
[9] scikit-learn. *http://scikit-learn.org/*, 2018.
[10] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 49–60, 1999.
[11] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin. On the relative trust between inconsistent data and inaccurate constraints. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 541–552, 2013.
[12] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 746–755, 2007.
[13] S. Chawla and A. Gionis. k-means-: A unified approach to clustering and outlier detection. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*, pages 189–197, 2013.
[14] K. Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.
[15] J. Chomicki and J. Marcinkowski. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance [result from a Dagstuhl seminar]*, pages 119–150, 2005.
[16] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.
[17] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 458–469, 2013.
[18] F. Daniel, F. Casati, T. Palpanas, O. Chayka, and C. Cappiello. Enabling better decisions through quality-aware reports in business intelligence applications. In *Proceedings of the 13th International Conference on Information Quality, MIT, Cambridge, MA, USA, 2008*, pages 310–324, 2008.
[19] I. Davidson and S. S. Ravi. A framework for determining the fairness of outlier detection. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, pages 2465–2472, 2020.
[20] A. Dobra. Decision tree classification. In *Encyclopedia of Database Systems*, pages 765–769. 2009.
[21] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.
[22] W. Fan. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 159–170, 2008.
[23] W. Fan. Constraint-driven database repair. In *Encyclopedia of Database Systems*, pages 458–463. 2009.
[24] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995.*, pages 127–138, 1995.
[25] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, 2004.
[26] X. Jin and J. Han. *K*-means clustering. In *Encyclopedia of Machine Learning and Data Mining*, pages 695–697. 2017.
[27] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 392–403, 1998.
[28] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 211–222, 1999.
[29] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB J.*, 8(3-4):237–253, 2000.
[30] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 53–62, 2009.
[31] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang. Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis]. *CoRR*, abs/1904.09483, 2019.
[32] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(2):97–108, 2012.
[33] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.
[34] C. Mayfield, J. Neville, and S. Prabhakar. ERACER: a database approach for statistical inference and data cleaning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 75–86, 2010.
[35] B. Micenková, R. T. Ng, X. Dang, and I. Assent. Explaining outliers by subspace separability. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 518–527, 2013.
[36] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.
[37] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.
[38] X. V. Nguyen, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, 2010.
[39] Y. Noh, F. C. Park, and D. D. Lee. Diffusion decision making for adaptive k-nearest neighbor classification. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1934–1942, 2012.
[40] C. K. Reddy, H. Chiang, and B. Rajaratnam. Stability region based expectation maximization for model-based clustering. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 522–531, 2006.
[41] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.
[42] F. Richter, Y. Lu, D. Kazempour, and T. Seidl. "show me the crowds!" revealing cluster structures through AMTICS. *Data Sci. Eng.*, 5(4):360–374, 2020.
[43] N. Rujeerapaiboon, K. Schindler, D. Kuhn, and W. Wiesemann. Size matters: Cardinality-constrained clustering and outlier detection via conic optimization. *SIAM J. Optim.*, 29(2):1211–1239, 2019.
[44] S. Sarawagi. Explaining differences in multidimensional aggregates. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 42–53, 1999.
[45] S. Song, C. Li, and X. Zhang. Turn waste into wealth: On simultaneous clustering and cleaning over dirty data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1115–1124, 2015.
[46] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 553–564, 2013.