How Useful are Hand-crafted Data? Making Cases for Anomaly Detection Methods

Len Du
Australian National University
len.du.public@gmail.com

Marcus Hutter
Australian National University
www.hutter1.net
marcus.hutter@anu.edu.au

Abstract

While the importance of small data has been admitted in principle, they have not been widely adopted as a necessity in current machine learning or data mining research. Most predominantly, machine learning methods were typically evaluated under a "bigger is better" presumption. The more (and the more complex) data we could pour at a method, the better we thought we were at estimating its performance. We deem this mindset detrimental to interpretability, explainability, and the sustained development of the field. For example, despite that new outlier detection methods were often inspired by small, low dimensional samples, their performance has been exclusively evaluated by large, high-dimensional datasets resembling real-world use cases. With these "big data" we miss the chance to gain insights from close looks at how exactly the algorithms perform, as we mere humans cannot really comprehend the samples. In this work, we explore in the exactly opposite direction. We run several classical anomaly detection methods against small, mindfully crafted cases on which the results can be examined in detail. In addition to better understanding of these classical algorithms, our exploration has actually led to the discovery of some novel uses of classical anomaly detection methods to our surprise.

1. Introduction

While the exact phrase "Big Data" is no longer a buzzword in academic contexts, "bigger data" continue to be regarded as almost universally superior to "smaller data" when evaluating machine learning algorithms. We seem to always prefer a "more inclusive" dataset when talking about evaluation of algorithms. Authors rarely claim that they consciously remove some samples with justification when describing experiments, while the identification and removal of outliers (anomalies) have been the norm in more traditional sciences of statistical nature, such as medicine [1].

The importance of small data has been acknowledged in principle. But works addressing them have only occasional occurred. Interesting, we have seen quite a few talks about small data in top venues [2, 3, 4, 5, 6], but actual technical papers have been observed far less in proportion. We speculate that works directly addressing small data tend to be easily accused of "lacking comprehensive evaluation", because of the aforementioned hidden hypothesis that bigger data are always better.

The most usual arguments in favor of small data are probably quality and cost. Less good samples might train a better model than more bad samples Sometimes data simply come at a very high cost. It is also apparent that we want small data when trying to understand, interpret, explain and analyze how an algorithm works by examples[7]. Few people would deny these frequent arguments. But they may still not be enough to support the claim that small data should be widely adopted as part of an evaluation kit, which leads to our following not-so-obvious-but-important argument about sustainability of the field.

1.1. Potential next AI winter vs. small data

Despite criticism from various perspectives [8, 9], deep learning has taken over the (largely overlapping) fields of machine learning, data mining and artificial intelligence due to its effectiveness. The advantage largely comes with the large amount of data in benchmarks [10, 11]. The ubiquity of "big benchmarks" in turn came with the argument that more data is almost always better [12].

Moreover, if you focus on more traditional machine learning and think that automating deep learning is not very relevant, then you might be wrong. For example, K-Means, which does not even use gradient descent itself, can be cast into a strictly equivalent deep neural network [13]. ML algorithms outside deep learning could still have an equivalent, similar or superior version in deep learning.

URI: https://hdl.handle.net/10125/70716
978-0-9981331-4-0

And then here comes a warning sign. Not too long ago, two prominent organisations, literally at the ground zero of the explosive growth of deep learning, received substantial funding cuts from their local government [14, 15]. One was the revered CIFAR (as in the CIFAR datasets [16]). The other was the newly-founded "flagship" AI-specific Vector Institute led by the DL "godfather" Geoffrey Hinton himself.

Not only deep learning might plateau, the advance of Neural Architecture Search [17] erodes the human factor in machine learning in the other direction. Deep learning already have too much trial-and-error in its use. As long as we cling to "big benchmarks", neural architecture search might very well simply perform better without us knowing why, blocking works that inquires why in the process. Then machine learning would become truly about machines, where researchers' few leftover tasks are mindlessly monitoring the search process and interfacing with client requests.

1.2. Anomaly detection as a specimen

Anomaly detection has attracted many surveys and benchmarks, among which ones with the larger datasets are deemed more comprehensive. Such surveys provide a good estimate of the accuracy and performance of unsupervised anomaly detection algorithms in production settings, in which the input is large and high-dimensional. But when one tries to look closely into each case, and find inspirations for improved algorithms, the high-dimensional large datasets from real world are hard to comprehend by a human, let alone why exactly some points are wrongly identified. In this regard, development of better algorithms often relies on a trial-and-error approach, where one changes some aspect of an algorithm, and runs the algorithm against a benchmark to see whether some of the accuracy statistics become better. Even if an improved algorithm were found, it may still be hard to interpret and reason about why it had been improved at all, other than that it performed better on a benchmark.

When we look at the original papers proposing the algorithms [18, 19, 20, 21, 22, 23], we always find one or a few very small "motivating examples" exactly "crafted" by the authors to explain why inventing the new methods. In a sense, the fact that the proposed algorithms fared better than previous ones against the crafted "motivating examples" has been accepted as supporting the arguments for the proposed algorithms, yet we don't see them in an evaluation paper! We are exactly about to pit multiple algorithms against each other in the face of the same set of crafted small point configurations.

1.3. Not few-shot learning

Finally, we would like to draw a line between our idea of small data and the more popular few-shot learning [24], as "few-shot" literally means a small number of samples. Few-shot learning stemmed out of practical need in computer vision, and deals with a few samples given parameters carried over from previous training with a much larger dataset, effectively a kind of transfer learning. In contrast, we are talking about small data in their purest form, in a more Platonian sense. By "evaluating with small data" we mean evaluating a clean-slate model with only a small amount of data. In particular, the unsupervised anomaly detection algorithms we examine here are all stateless. It is impossible for them to carry over states from previous training. By "small" we also imply low-dimensionality.

2. Outlier detection methods

We limit ourselves to methods based on k-nearest-neighbors, in particular those with no other hyperparameters than k itself. Methods based on k-nearest-neighbors, sometimes referred to as density-based methods [25], nearest-neighbor based methods [26], neighbour-based methods [27] or proximity-based methods [28], are among the most popular detection methods for their ease of deployment. Because these methods are local, our small-data cases are speculated to be generalizable to larger real-world cases. Note that these unsupervised methods are entirely different from applying a kNN-classifier trained with labeled inliers and outliers.

Our test policy is, briefly speaking, letting each algorithm do its best regardless of the necessary k. Comparing algorithms with different numbers of hyperparamters would not be as fair as comparing ones with the same set of hyperparameters.

To facilitate a precise notation, *duplicate points* need some attention. In general we would like to allow duplicate points in our analysis. However, allowing duplicate points prevents addressing "set of points" directly as a set cannot have duplicate members. This can be side-stepped by addressing the *index* of points throughout. It is not so hard to remove duplicates beforehand, so many algorithms were designed to work on datasets without duplicates. When using floating-point numbers, it may be problematic to decide whether any two points are equal. For generality, we will try to accommodate duplicates in our formulation as much as possible, but use only test cases without duplicates.

Assume that there are P points in $\mathbb{R}^n$ from which

we are to find outliers. As has been noted, we use $1, 2, \ldots, P \in \mathbb{N}$ to refer to the points. Then the actual coordinates are in a $n \times P$ matrix X, each column of which defines one point. It is usually assumed that $X \in \mathbb{R}^{n \times P}$, but we refrain from this constraint for maximum generality. Instead, we just write $X \in S^{n \times P}$, where S can be $\mathbb{R}$ or a set of other scalars.

In principle, any metric can be used as the distance function between points. However, not all algorithms were designed with this generality in mind. For now we just use $d : S^n \times S^n \to S$ to stand for the distance function. Although most of the time we can just assume it to be the Euclidean distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = (\sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2)^{\frac{1}{2}} .$$

A shorthand for distances between indexed points is

$$D_{i,j} = d(X_{\bullet,i}, X_{\bullet,j}) .$$

2.1. Basic kNN methods

k-nearest neighborhood is probably the most employed concept in outlier detection. To facility further discussion, it is better to define some common concepts first.

First of all we assume that there are enough points to establish the k-nearest neighbourhood, and that k is not zero, that is,

$$0 < k < P .$$

k-Nearest Neighborhood has not k points. With or without duplicate points, it is possible to encounter a situation where a point has *multiple neighbours with the same distance*, making it impossible to choose exactly k points solely based on distance. This could actually be frequent if we wish to consider Manhattan distance or fixed-point arithmetic. To work around this we define a point q being a member of the k-nearest neighbor of point p as that there are less than k points which are closer to p than q, that is,

$$\begin{aligned} \text{Ord}(p, q) &= |\{i \in \{1..P\} \mid p \neq i \neq q \wedge D_{p,i} < D_{p,q}\}| , \\ \text{NN}_k(p) &= \{q \in \{1..P\} \mid q \neq p \wedge \text{Ord}(p, q) < k\} , \\ \text{kNN}(p) &= \text{NN}_k(p) . \end{aligned}$$

We use both $\text{NN}_k(p)$ and $\text{kNN}(p)$ as k is usually understood from the context in discussions, but sometimes we wish to vary the k explicitly. As such it is true that

$$|\text{kNN}(p)| \geq k ,$$

where the equality may not hold if there are multiple neighbours with the same distance, which cannot really be prevented in a real program despite having an expectation of zero. This definition is compatible with that in [19], although defined in a different way, while many other publications ignored this issue, which may lead to inconsistencies, and possibly bugs in implementations.

k^{th}NN The *kth nearest neighbour* (k^{th}NN) [29] method is certainly one of the major approaches to outlier detection. In k^{th}NN , the outlierness of a point is its distance to its kth nearest neighbor. The original paper admits the use of any L_p metric as the distance. Here we only evaluate the algorithm with L_2, or Euclidean distance.

$$\text{OL}(p) = \text{k-th-dist}(p) = \max\{D_{p,q} \mid q \in \text{kNN}(p)\}$$

Note that even if the implementation ignored the possibility that $\text{kNN}(p) > k$, the value of $\text{k-th-dist}(p)$ would still be always correct. The implementation we use [30] differs from the original paper in the definition of k but it does not affect our results since we always try to find the best k.

kNNw Instead of the distance to the kth nearest neighbor, *kNNw* [22] defines the outlierness of a point as the sum of distances to its k nearest neighbors, that is,

$$\text{OL}(p) = \text{k-dist}(p) = \sum_{i=1}^{k} \max\{D_{p,q} \mid q \in \text{NN}_k(p)\} .$$

Note that we can't simply define it to be

$$\sum_{q \in \text{kNN}(p)} D_{p,q}$$

because even if $|\text{kNN}(p)| > k$ we still want to sum only k of the $|kNN(p)|$ numbers, otherwise the value may be biased. Instead, we have to iterate and sum through the i-th smallest neighbor-distance.

LIC One immediate way to invent another method would be to somehow combine k-th-dist and k-dist. LIC [20] did that and used other distance functions instead of the Euclidean one. We limit ourselves to Euclidean distances in this work, but the averaged version is worth attention. To be precise, the version of LIC in this paper is defined as

$$\text{OL}(p) = \text{LIC}(p) = \text{k-th-dist}(p) + \frac{\text{k-dist}(p)}{k} ,$$

so that k-dist(p) term does not shadow k-th-dist(p) when k is larger. This is also the default in [30].

ODIN Outlier Detection using Indegree Number (ODIN) [18] employs a kNN graph, which we will see later that we can understand the method without. To construct the graph, each data point p corresponds to a vertex with directed edges the corresponding vertices of k-nearest neighbours of p. Formally, we can define the (directed) kNN graph as

$$\begin{aligned} \text{kNND}(p) &= (V_{\text{kNN}}(p), E_{\text{kNND}}(p)) \,, \\ V_{\text{kNN}}(p) &= \text{kNN}(p) \cup \{p\} \,, \\ E_{\text{kNN}}(p) &= \{(p,q) \mid q \in \text{kNN}(p)\} \,. \end{aligned}$$

Data points whose corresponding vertices have the least indegrees are considered outliers, where the number of indegrees can be computed as

$$\begin{aligned} \text{Ind}(p) &= |\{q \in \{1..P\} \mid (q,p) \in E_{\text{kNN}}(q)\}| \\ &= |\{q \in \{1..P\} \mid p \in \text{kNN}(q)\}| \,. \end{aligned}$$

Or we can equivalently define the outlierless by any strictly decreasing function of the number of indegrees. A definition that is consistent with the implementation under test is

$$\text{OL}(p) = \text{ODIN}(p) = \frac{-\text{Ind}(p)}{k} \,,$$

whose linearity also arguably provides ease for both computers and human readers. As can be seen from the simplified computation of Ind(p), this method can actually be understood without knowing about the graph at all.

2.2. Methods based on LOF

A large class of unsupervised anomaly detection algorithms are based on the well-known *Local outlier factor (LOF)* [19] method.

LOF LOF has only one hyperparameter k. Given k, the outlierness of of point p, LOF(p), is defined as

$$\text{OL}(p) = \text{LOF}(p) = \frac{1}{|\text{kNN}(p)|} \sum_{q \in \text{kNN}(p)} \frac{\text{lrd}(q)}{\text{lrd}(p)} \,,$$

where the local reachability density of p, lrd(p), is defined as

$$\begin{aligned} \text{lrd}(p) &= \frac{|\text{kNN}(p)|}{\sum_{q \in \text{kNN}(p)} \text{reach-dist}(p,q)}, \\ \text{reach-dist}(p,q) &= \max\{\text{k-th-dist}(q), D_{p,q}\} \,. \end{aligned}$$

Although named *reachability distance*, reach-dist is not symmetric, and thus not a proper distance.

LDOF As with LOF, LDOF has only one hyperparameter k. Instead of the k-th smallest neighbor-distance, average kNN distance k-a-dist(p) and kNN inner distance k-i-dist(p) are used, defined as

$$\begin{aligned} \text{k-a-dist}(p) &= \frac{1}{|\text{kNN}(p)|} \sum_{q \in \text{kNN}(p)} D_{p,q} \,, \\ \text{k-i-dist}(p) &= \frac{\sum_{i,j \in \text{kNN}(p) \wedge i \neq j} D_{i,j}}{|\text{kNN}(p)|(|\text{kNN}(p)|-1)} \\ &= \frac{\sum_{i,j \in \text{kNN}(p)} D_{i,j}}{|\text{kNN}(p)|(|\text{kNN}(p)|-1)} \,. \end{aligned}$$

In the original literature [21] it is k instead of $|\text{kNN}(p)|$. Using $|\text{kNN}(p)|$ is consistent with [19] as well as the implementation, and more general. Then the outlierness is given by

$$\text{OL}(p) = \text{LDOF}(p) = \frac{\text{k-a-dist}(p)}{\text{k-i-dist}(p)} \,.$$

sLOF sLOF is said to be a method which happens frequently in practice by mistaking the reachability "distance" with the actual distance. [31] It is mildly faster as well. The exact definition is

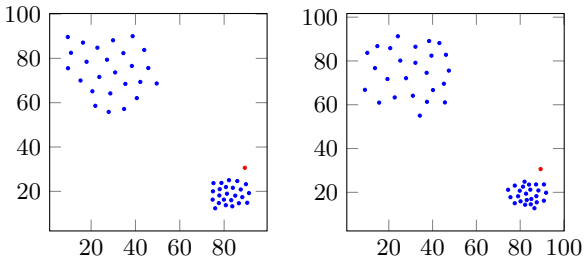
$$\begin{aligned} \text{s-lrd}(p) &= \frac{|\text{kNN}(p)|}{\sum_{q \in \text{kNN}(p)} D_{p,q}}, \\ \text{OL}(p) = \text{sLOF}(p) &= \frac{1}{|\text{kNN}(p)|} \sum_{q \in \text{kNN}(p)} \frac{\text{s-lrd}(q)}{\text{s-lrd}(p)} \,. \end{aligned}$$

3. Challenging the detection algorithms

All algorithms introduced, as well as most outlier detection algorithms in general, output a scalar "outlierness" score for each point and require further thresholding. To avoid tuning for the threshold, we consider an algorithm to have succeeded on a case as long as the range of scores over inliers (by label) can be separated from that of outliers. That is, either the maximum score of labeled inliers is less than the minimum score of labeled outlier(s), or the maximum score of labeled outlier(s) is less than the minimum score of labeled inlier(s). All test results are aggregated in Table 1 for quick reference once the reader have understood the rest of the paper.

Table 1. Aggregated test results

	k^{th}NN	kNNw	LIC	ODIN	LOF	sLOF	LDOF
Case SD.1	✗	✗	✗	✓	✓	✓	✓
Case SD.2	✗	✗	✗	✓	✓	✓	✓
Case SD.3	✗	✗	✗	✓	✓	✓	✓
Case SD.4	✗	✗	✗	✓	✓	✓	✓
Case L1.1	✓	✓	✓	✓	✓	✓	✓
Case L1.2	✗	✗	✗	✗	✗	✗	✗
Case L1.3	✓	✓	✓	✓	✓	✓	✗
Case L1.4	✓	✓	✓	✓	✓	✓	✗
Case L1.5	✓	✓	✓	✓	✓	✓	✗
Case L1.6	✓	✓	✓	✓	✓	✓	✗
Case L1.7	✗	✗	✗	✗	✓	✗	✗
Case L1.8	✗	✗	✗	✗	✗	✗	✗
Case L1.9	✗	✗	✗	✓	✗	✗	✗
Case LD	✓	✓	✓	✗	✓	✓	✓
Case LD/1	✓	✓	✓	✗	✓	✓	✓
Case LD/2	✓	✓	✓	✗	✓	✓	✓
Case LD/3	✓	✓	✓	✓	✓	✓	✓
Case G.1	✗	✗	✗	✗	✓	✗	✗
Case G.2	✓	✗	✓	✓	✓	✗	✗
Case G.3	✓	✓	✓	✓	✓	✓	✗
Case G.4	✗	✗	✗	✗	✓	✗	✗
Case G.5	✗	✗	✗	✗	✓	✗	✗



(a) Case SD.1 (b) Case SD.2

(c) Case SD.3 (d) Case SD.4

Figure 1. Sparse and dense clusters

3.1. Sparse and dense clusters

A sparse cluster and a dense cluster in each of Figure 1. The distance from the outlier to the dense cluster, is about the distance between inliers of the sparse cluster. Variations of actual clusters have been produced to check the robustness of results.

This kind of situations was exactly one of the inspirations behind LOF. Not surprisingly, all LOF-based methods succeeded in these cases, while k^{th}NN and kNNw failed. The results were exactly the same across the perturbed versions, that is, Case SD.1 to Case SD.4.

It is worth mentioning that ODIN also succeeded in disambiguating these outliers, despite being a "global"

Table 2. Test results of sparse and dense clusters

	k^{th}NN	kNNw	LIC	ODIN	LOF	sLOF	LDOF
Case SD.1	✗	✗	✗	✓	✓	✓	✓
Case SD.2	✗	✗	✗	✓	✓	✓	✓
Case SD.3	✗	✗	✗	✓	✓	✓	✓
Case SD.4	✗	✗	✗	✓	✓	✓	✓

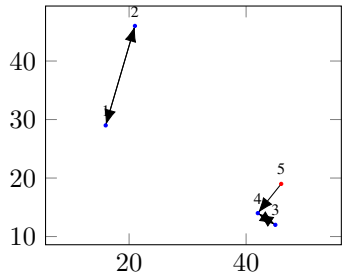


Figure 2. Minimal sparse-and-dense case

method along with k^{th}NN and kNNw, in contrast to LOF-inspired ones [31]. ODIN essentially used only the information of whether a point was in another point's kNN set. The reason is that the inliers point their outbound edges to other inliers in the same cluster, rather than the outlier. And the outlier has to point its outbounding edges to inliers, although no inbound edge received. In other words, without graph theory jargon, the kNN set of both inliers and outliers are composed of inliers, so inliers are likely to be in at least one point's kNN set, yet the outlier can be in no point's kNN set. To illustrate, we show a vastly simplified example in Figure 2, with $k = 1$ and

$$X = \begin{bmatrix} 16 & 21 & 45 & 42 & 46 \\ 29 & 46 & 12 & 14 & 19 \end{bmatrix},$$

$$\text{kNN}(1) = \{2\}\,, \text{kNN}(2) = \{1\}\,, \text{kNN}(3) = \{4\}\,,$$
$$\text{kNN}(4) = \{3\}\,, \text{kNN}(5) = \{4\}\,.$$

It is clear from Figure 2 that the outlier, point 5, is in no point's kNN set, but other points are always in another point's kNN set. So ODIN is able to separate the inliers from the outliers in this case. Also note that the distance between point 1 and 2 is larger than that between 5 and 4, rendering k^{th}NN and kNNw ineffective.

3.2. Straight line and outlier

As shown in Figure 3, Case L1.1 to Case L1.8 represent a class of configurations where the inliers form (approximately) a line (segment) or more generally, lie in a subspace of lower dimensions.

The first case (Case L1.1) is the easiest, where the distance from the outlier to the line of inliers

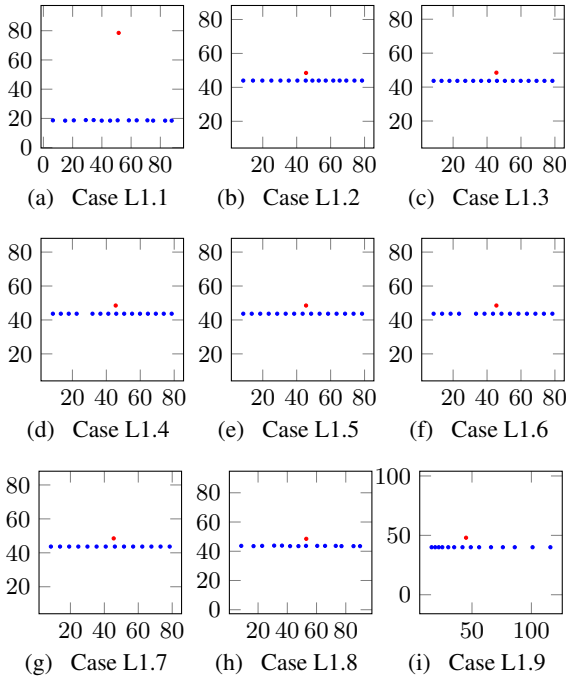


Figure 3. Straight line and outlier

Table 3. Test results of straight line and outlier

	k^{th}NN	kNNw	LIC	ODIN	LOF	sLOF	LDOF
Case L1.1	✓	✓	✓	✓	✓	✓	✓
Case L1.2	✗	✗	✗	✗	✗	✗	✗
Case L1.3	✓	✓	✓	✓	✓	✓	✗
Case L1.4	✓	✓	✓	✓	✓	✓	✗
Case L1.5	✓	✓	✓	✓	✓	✓	✗
Case L1.6	✓	✓	✓	✓	✓	✓	✗
Case L1.7	✗	✗	✗	✗	✓	✗	✗
Case L1.8	✗	✗	✗	✗	✗	✗	✗
Case L1.9	✗	✗	✗	✓	✗	✗	✗

is much larger than those between successive inliers. The feature of the second case (Case L1.2) is that there are significant variations in the distances between successive points on the line. The four cases from Case L1.3 to Case L1.7 are visually similar. Case L1.3 and Case L1.5 are different in that, in Case L1.3, the projection of the outlier on the line of inliers is very close to an inlier, while in Case L1.5 the outlier is projected to the middle of a gap. Case L1.4 and Case L1.6 are modified from Case L1.3 and Case L1.4 respectively, each with an inlier removed for a larger gap. Case L1.7 differs from Case L1.3 by having two less points and larger gaps between successive inliers. Points in Case L1.8 are significantly perturbed. Case L1.9 has large variations of gap sizes.

The results are shown in Table 3. It is clear from the result of Case L1.1 that as long as the distance from an outlier to a line is much larger than the gaps between points in the line, any unsupervised anomaly detection algorithm works reasonably well. So only the rest of the

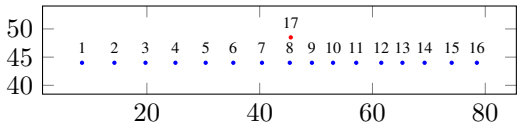


Figure 4. Case L1.2

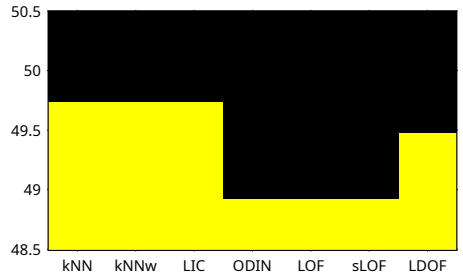


Figure 5. Test result represented by yellow (failure) and black (success) pixel strips.

cases, where the outlier-to-line distance and gap sizes are comparable, are interesting.

In Case L1.2, all methods failed, which was quite intriguing. On the first sight, it was tempting to scale the gaps between the inliers and see what would happen. However, unsupervised anomaly detection algorithms are usually assumed to be scale-invariant, so it should be sufficient to move the outlier around instead. For ease of discussion, we number the points as in Figure 4.

All the inliers had the same y-coordinate 44, while the y-coordinate of the outlier in the original Case L1.2 was 48.5. Then we moved the outlier vertically to see how each unsupervised anomaly detection algorithm would work. The configuration being simple, we could afford to run a three-digit equal-step scanning of the outlier's y-coordinate from 48.50 to 50.49 with a 0.01 increment, 200 test cases in total. Luckily, the result turned out to exhibit mononicity. That is, we observed that if any one method succeeded at discriminating the outlier with y-coordinate y, it would then succeed at all the cases where the outlier has y-coordinate $y' \geq y$. To showcase this, the result is plotted in Figure 5, where each case is represented by a yellow or black strip of pixels. The monotocity is evident from the fact that there is no distinguishable strips but only two color blocks in the plot.

3.3. Case where LDOF was claimed to have advantages over LOF

We recovered the precise configuration used as the "motivating example", in the LDOF paper [21] by parsing the PDF file. As a result, there may be some

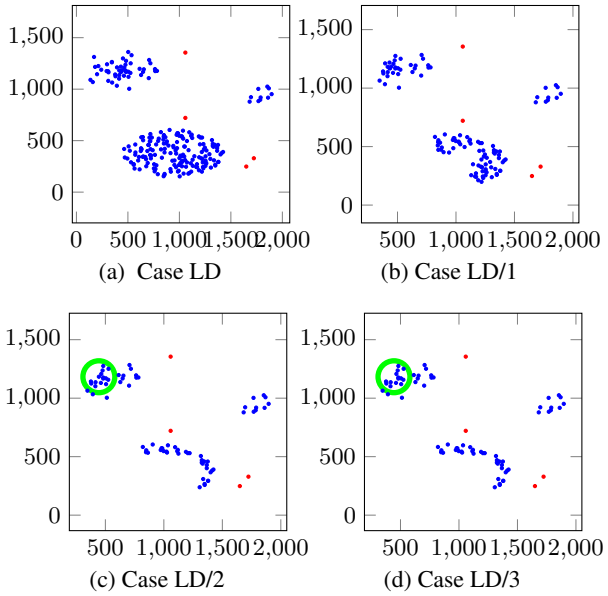


Figure 6. The LDOF case and its modifications (lower two differing only by the circled point)

Table 4. Test results of the LDOF case and its modifications

	k^{th}NN	kNNw	LIC	ODIN	LOF	sLOF	LDOF
Case LD	✓	✓	✓	✗	✓	✓	✓
Case LD/1	✓	✓	✓	✗	✓	✓	✓
Case LD/2	✓	✓	✓	✗	✓	✓	✓
Case LD/3	✓	✓	✓	✓	✓	✓	✓

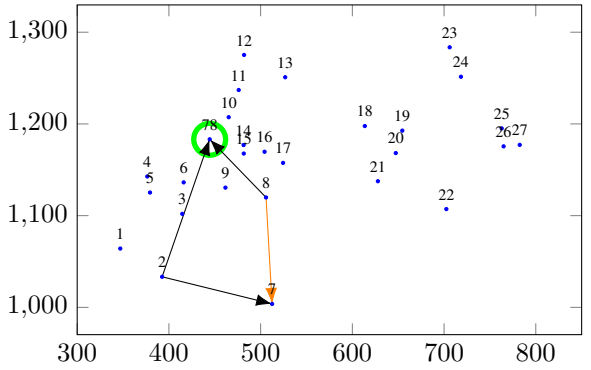


Figure 7. Local view of Case LD/2

uniform scaling but all methods covered in this paper should work invariantly under uniform scaling. This case, along with some modified versions which are about to be explained in this section, are plotted in Figure 6. As can be seen from Table 4, although the case was designed to show the superiority of LDOF, all methods except ODIN passed the test. Actually, in the original paper [21], it was only claimed that this case defeated k^{th}NN , kNNw and LOF when $k > 10$. And in our tests all the algorithms did pass with $k < 10$ (in particular, $k = 2$ for both k^{th}NN and kNNw, and $k = 3$ for LOF). So our results do not technically conflict with those in [21]. In retrospect, the test case did seem somewhat lax in terms of supporting LDOF, as there was no obvious reason why we should prefer some $k > 10$ over $k = 2$ when encountered by this case.

That said, it is accidentally interesting why ODIN alone failed. To ease the analysis, we gradually removed some of the inlier points from the configuration until the result of any method changed. The modified cases are displayed in Figure 6, while the results are listed in Table 4 alongside the result of the original case. We were also lucky enough to find two cases Case LD/2 and Case LD/3, where the only difference was the latter having one less point from the former, while ODIN failed in the former but succeeded in the latter.

Case LD/3 was passed by ODIN with $k = 9$. For a reasonably small k, including $k = 9$ of course, the two outliers in the middle have an indegree number of 0, while the two outliers in the lower right corner have an indegree number of 1, being the nearest neighbor of each other, which we have manually verified to be true for both Case LD/1 and Case LD/2, forall $1 \leq k \leq 9$. So as long as all the inliers have an indegree number no less than 2, ODIN can successfully disambiguate.

Now, for Case LD/2 and $k = 9$ we have

$$7 \in \text{kNN}(2) = \{1, 3, 4, 5, 6, 7, 8, 9, 78\} ,$$
$$7 \notin \text{kNN}(8) = \{3, 6, 9, 10, 14, 15, 16, 17, 78\} ,$$
$$\text{Ind}(7) = 1 .$$

And for Case LD/3,

$$7 \in \text{kNN}(2) = \{1, 3, 4, 5, 6, 7, 8, 9, 15\} ,$$
$$7 \in \text{kNN}(8) = \{3, 6, 7, 9, 10, 14, 15, 16, 17\} ,$$
$$\text{Ind}(7) = 2 .$$

That is, as displayed in Figure 7, when point 78 is not present, point 7 is in the kNN set of point 8 and thus has two indegrees, which is why ODIN passed Case LD/3. When point 78 is present, as is the case in Case LD/2, it takes the place of point 7 in the kNN set of point 8 (for k=9), so point 7 has only one indegree and becomes indistinguishable from the outliers. That said, in this local view, this point does look like a local outlier to a certain extent. If we decreased k below 9, point 7 would be no longer in the kNN set of point 8 by being exactly the 9-th nearest neighbour, so neither Case LD/2 or Case LD/3 could be passed when k¡9. For

Table 5. Test results of grid patterns

	k^{th}NN	kNNw	LIC	ODIN	LOF	sLOF	LDOF
Case G.1	✗	✗	✗	✗	✓	✗	✗
Case G.2	✓	✗	✓	✓	✓	✗	✗
Case G.3	✓	✓	✓	✓	✓	✓	✗
Case G.4	✗	✗	✗	✗	✓	✗	✗
Case G.5	✗	✗	✗	✗	✓	✗	✗

completeness, all points would have indegree number 0 when k=0, disabling the ODIN method completely. If we increased k to 10, the indegree number of one of the outlier would suddenly increase to 8, a value greater than those of a significant portion of the inliers, which means k was already too large and the method started to lose their disambiguation power globally.

We take the finding this way. The ODIN method is scale-invariant within one configuration, it is equally likely to recognise small, local features and larger-scale features, similar to a fashion in fractals. On the other hand, the indegree number of any point p increases monotonically from $\text{Ind}(p) = 0$ when $k = 0$ to $\text{Ind}(p) = (P-1)$ when $k = (P-1)$, since $\text{kNN}(q) = \{1..P\} \setminus \{q\}$ for any q, but the way $\text{Ind}(p)$ grows with k depends on the outlierness of p. The scale-invariance may be desirable sometimes, but when it is unwanted, which is the case for general use of outlier-detection, we cannot separate small-scale features from large-scale ones using the indegree numbers under a single k. Considering the curve of $\text{Ind}(p)$ versus k rather than a single value could lead to more advanced methods. Alternatively, combining $\text{Ind}(p)$ with other methods to filter out small-scale "outliers" also seems like a viable approach. More extensive study on ODIN itself can be conducted with data patterns generated with fractals in mind.

3.4. Grid patterns

While being "outliers" in a literal sense, irregularities in grid-like patterns as demonstrated by Case G.1 to Case G.5 displayed in Figure 8 are probably not intended for most unsupervised anomaly detection algorithms. We initially prepared these patterns with the expectation that they are very challenging for unsupervised anomaly detection algorithms, only to find that these methods work reasonably well on these configurations. The results are shown in Table 5.

At first sight, it is tempting to consider the effectiveness coming from the fact that the outliers are unusually close to or far away from its surrounding points, compared to the inliers. However, the presumed outliers in grid-less density patterns such as Case B.1 are harder to detect. So the unsupervised anomaly detection algorithms seem to be actually picking up the pattern and need a closer look.

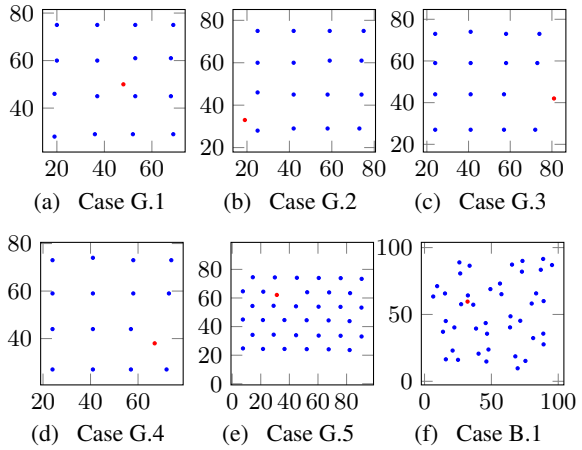


Figure 8. Grid patterns

From the results, vanilla LOF stands out as being particularly effective for the grid-like patterns, failing on only 1 case.

For Case G.1 we have used

$$X = \begin{bmatrix} 48 & 19 & 36 & 52 & 69 & 19 & 37 & 53 & 68 & 68 & 53 & 37 & 20 & 20 & 37 & 53 & 69 \\ 50 & 28 & 29 & 29 & 29 & 46 & 45 & 45 & 45 & 61 & 61 & 60 & 60 & 75 & 75 & 75 & 75 \end{bmatrix},$$

the first column being the presumed outlier,and

$$P = 17\ , k = 16\ ,$$

For LOF we have

$$\text{lrd}(1) = 0.0176\ , \qquad \text{LOF}(1) = 1.0218\ ,$$
$$\text{lrd}(7) = 0.0177\ , \qquad \text{LOF}(7) = 1.0143\ .$$

In this case $k = P - 1$, and the outlier seems to be identified by it being at the most "central" location. It could be argued that this is just a special case.

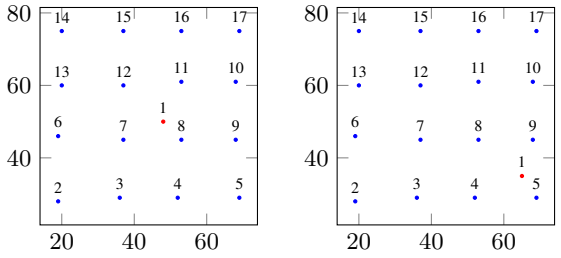
An obvious next step would be to move the "outlier" point somewhere else so that the special case cannot hold, as shown in Figure 9b. But in this case we have found that LOF was able to distinguish the outlier with $k = 2$. More specifically, for the case shown in Figure 9b, we have

$$X = \begin{bmatrix} 65 & 19 & 36 & 52 & 69 & 19 & 37 & 53 & 68 & 68 & 53 & 37 & 20 & 20 & 37 & 53 & 69 \\ 35 & 28 & 29 & 29 & 29 & 46 & 45 & 45 & 45 & 61 & 61 & 60 & 60 & 75 & 75 & 75 & 75 \end{bmatrix},$$

and

$$P = 17\ , \qquad \text{LOF}(1) = 1.1815\ ,$$
$$\text{LOF}(5) = 0.9344\ , \quad \text{LOF}(8) = 1.0809\ ,$$

where point 8 has the greatest LOF score among inliers. In this case, since the LOF scores were seperated



(a) Case G.1 (b) Modified Case G.1

Figure 9. Case G.1 and its modification

between point 8 and point 1, rather than between point 5 and point 1, it can still be argued to be a special case.

However, LOF is observed to succeed on most "grid-like" cases, which is not yet well explained by the one special case. It is not hard to generate a series of similar cases with different grid sizes and different "outlier" positions. But before that we should check other grid-like patterns in Table 5 first. And in Case G.2, Case G.4 and Case G.5, LOF all successfully identified outliers with $k = 2$.

It would then be tempting to postulate that LOF works on grid-like patterns generally with either $k = P - 1$ or $k = 2$. We went on to extensively generate grid-like patterns of various sizes and apply LOF against them.

The experiment setup was to generate square grids of different sizes, and cells of different heights as we consider the method to be scale- and rotation-invariant but not necessarily invariant with uneven scaling on two axes. Some perturbations were also applied to each grid points. We then inserted an outlier around a grid point with random perturbation greater than that of any regular grid points. Then the "original" grid point around which the outlier was added could either be removed or not, and we considered both cases.

For both types of cases, four cell aspect ratios 1, 2, 4 and 8 were used. And for each combination of aspect ratio and grid size, 40 experiments with different pseudo-randomness seeds for perturbations were generated. Grid sizes of 4, 5, 8, 9, 13, 16 and 32 were used, resulting in a total of 2240 cases. Examples are shown in Figure 10.

We found that in more than half of the 2240 cases, LOF was able to identify the outlier successfully, with a small $k < 5$ most of the time. There were no more apparent patterns regarding whether a configuration could or could not be successfully detected by LOF. The LOF method certainly has great potential to be adapted for grid-like patterns, and we will need a more dedicated future work in this regard.

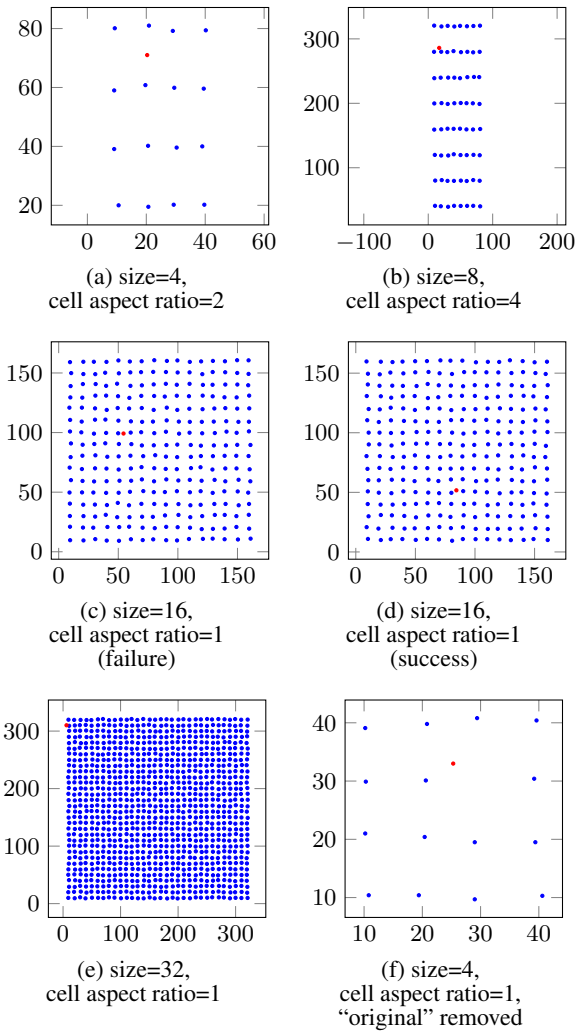


Figure 10. Examples of extensive cases for LOF in grid-like patterns

4. Discussion

In this work we have tested several classical unsupervised anomaly detection algorithms with manually designed cases. Both the size of each case and the total number of cases are much smaller than a typical review. And yet we have found an unexpected use case for outlier detection methods, that is the grid patterns. We do learn more of algorithms with less data and more attention. While objective, automated and extensive evaluation of algorithms are certainly desirable, at present the human attention is still valuable. For data-mining practitioners, this paper also provides some useful heuristics for generic outlier detection algorithms.

This paper is by no means exhaustive in terms of exhibiting the effectiveness of small data in machine learning research. The main point here is to rehearse the

perspective stressing the small, well designed examples that is sometimes assumed with existing research but rarely elaborated or made a focus of research. We hope this perspective could be used across many different classes of data-mining/machine-learning algorithms.

References

[1] P. L. Canner, Y. Huang, and C. L. Meinert, "On the detection of outlier clinics in medical and surgical trials: I. practical considerations," *Controlled clinical trials*, vol. 2, no. 3, pp. 231–240, 1981.

[2] Y. W. Teh, "On big data learning for small data problems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (New York, NY, USA), p. 3, Association for Computing Machinery, 2018.

[3] C. Neumann, "Using "big data" to solve "small data" problems," in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013* (I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, eds.), p. 1140, ACM, 2013.

[4] F. Geerts, "Scale independence: Using small data to answer queries on big data (invited talk)," in *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016* (W. Martens and T. Zeume, eds.), vol. 48 of *LIPIcs*, pp. 2:1–2:2, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[5] Z. Li, H. Yao, and F. Ma, "Learning with small data," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, (New York, NY, USA), p. 884–887, Association for Computing Machinery, 2020.

[6] O. Kennedy, D. R. Hipp, S. Idreos, A. Marian, A. Nandi, C. Troncoso, and E. Wu, "Small data," in *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pp. 1475–1476, IEEE Computer Society, 2017.

[7] J. J. Faraway and N. H. Augustin, "When small data beats big data," *Statistics & Probability Letters*, vol. 136, pp. 142–145, 2018.

[8] M. Hutson, "Artificial intelligence faces reproducibility crisis," *Science*, vol. 359, no. 6377, pp. 725–726, 2018.

[9] G. Marcus, "Deep learning: A critical appraisal," 2017.

[10] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences*, 2020.

[11] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

[12] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[13] L. Du, "Shallow deep learning: Embedding verbatim k-means in deep neural networks," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov 2019.

[14] T. C. Press, "Ontario government cuts $24 million in ai research funding," *CBCnews*, May 2019.

[15] K. Allen, "Ford government slashes funding to research institutes focused on artificial intelligence," *thestar.com*, May 2019.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[17] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[18] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," ICPR '04, (Washington, DC, USA), pp. 430–433, IEEE Computer Society, 2004.

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, pp. 93–104, May 2000.

[20] B. Yu, M. Song, and L. Wang, "Local isolation coefficient-based outlier mining algorithm," *2009 International Conference on Information Technology and Computer Science*, vol. 2, pp. 448–451, 2009.

[21] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," (Berlin, Heidelberg), pp. 813–822, Springer Berlin Heidelberg, 2009.

[22] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," (Berlin, Heidelberg), pp. 15–27, Springer Berlin Heidelberg, 2002.

[23] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: challenges and research questions a position paper," *Acm Sigkdd Explorations Newsletter*, vol. 15, no. 1, pp. 11–22, 2014.

[24] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[25] E. Schubert, A. Zimek, and H.-p. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, pp. 190–237, 01 2014.

[26] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLOS ONE*, vol. 11, pp. 1–31, 04 2016.

[27] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[28] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, "Generative adversarial active learning for unsupervised outlier detection," *CoRR*, vol. abs/1809.10816, 2018.

[29] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, pp. 427–438, May 2000.

[30] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek, "A framework for clustering uncertain data," *PVLDB*, vol. 8, no. 12, pp. 1976–1979, 2015.

[31] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, 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*, vol. 30, pp. 891–927, Jul 2016.