Good and Bad Neighborhood Approximations for Outlier Detection Ensembles

Author manuscript – the final publication is available at Springer via
https://doi.org/10.1007/978-3-319-68474-1_12

Evelyn Kirner[1], Erich Schubert[2], and Arthur Zimek[3]

[1] Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
evelyn.kirner@campus.lmu.de
[2] Heidelberg University
schubert@informatik.uni-heidelberg.de
[3] University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark
zimek@imada.sdu.dk

Abstract. Outlier detection methods have used approximate neighborhoods in filter-refinement approaches. Outlier detection ensembles have used artificially obfuscated neighborhoods to achieve diverse ensemble members. Here we argue that outlier detection models could be based on approximate neighborhoods in the first place, thus gaining in both efficiency and effectiveness. It depends, however, on the type of approximation, as only some seem beneficial for the task of outlier detection, while no (large) benefit can be seen for others. In particular, we argue that space-filling curves are beneficial approximations, as they have a stronger tendency to underestimate the density in sparse regions than in dense regions. In comparison, LSH and NN-Descent do not have such a tendency and do not seem to be beneficial for the construction of outlier detection ensembles.

1 Introduction

Any algorithm will have different points of optimization. More often than not, it is not the algorithm that needs to be optimized, but the actual implementation. Implementation details can yield substantial performance differences, in particular when scripting languages such as R and Python or just-in-time optimization such as in Java and Scala are used [29]. An implementation detail often not even mentioned in passing in publications describing a novel outlier detection algorithm is the computation of neighborhoods. Typical outlier detection algorithms compute some property for characterizing outlying behavior based on the nearest neighbors of some object and compare that property for a given object with the corresponding properties of some context of neighboring objects [42]. Because of its complexity, a central bottleneck for all these algorithms is usually the computation of object neighborhoods.

Table 1: Runtime breakdown of LOF (with $k = 100$) in ELKI 0.6.0

Data set	DBpedia 475.000 instances, 2 dimensions						
Index	linear scan		k-d tree		R*-tree		Theoretical
	(ms)	(%)	(ms)	(%)	(ms)	(%)	complexity
Load Ascii data	990	0.04	1057	4.82	1035	5.99	$\mathcal{O}(n)$
Bulk-load index	0	0.00	829	3.78	768	4.44	$\mathcal{O}(n \log n)$
kNN search	2672128	99.74	15740	71.72	11379	65.85	$\mathcal{O}(n^2)$, maybe $n \log n$
LOF	5879	0.22	4319	19.68	4099	23.72	$\mathcal{O}(nk)$
Data set	ALOI 75.000 instances, 27 dimensions						
Index	linear scan		k-d tree		R*-tree		Theoretical
	(ms)	(%)	(ms)	(%)	(ms)	(%)	complexity
Load Ascii data	2238	0.96	2232	0.50	2231	1.27	$\mathcal{O}(n)$
Bulk-load index	0	0.00	624	0.14	996	0.56	$\mathcal{O}(n \log n)$
kNN search	230030	98.84	446653	99.28	172791	97.99	$\mathcal{O}(n^2)$, maybe $n \log n$
LOF	468	0.20	372	0.08	321	0.18	$\mathcal{O}(nk)$

We demonstrate this in a motivating experiment, using the ELKI framework [2] since it offers many algorithms as well as several index structures for acceleration. In Table 1 we give runtime benchmark results running the LOF [10] algorithm (local outlier factor) on two larger data sets: the first data set contains all GPS coordinates from DBpedia [31], the second 27 dimensional color histograms for the Amsterdam Library of Object Images (ALOI, [17]). We expect the first to be more amiable to index acceleration using spatial indexes such as the k-d tree [9] and the R*-tree [19, 8]. For each data set, we report the runtime broken down into (i) loading the data from text files into memory, (ii) bulk-loading the index, (iii) searching the kNN of each object, and (iv) computing the LOF scores. We repeat the experiments using a linear scan, using a k-d tree, and using a bulk-loaded R*-tree; we also give theoretical results on the complexity of each step.

Both from a theoretical point of view as well as supported by the empirical results presented here, step (iii), computing the kNN of each object, is the main contributor to total runtime. However, it also becomes evident that the constant factors in the runtime analysis should probably not be as easily dismissed (see also the more extensive discussion by Kriegel et al. [29]). Bulk-loading the index is usually in $\mathcal{O}(\log n)$, while for kNN search with indexes an optimistic empirical estimate is $n \log n$, and the theoretical worst case supposedly is between $\mathcal{O}(n^{4/3})$ and $\mathcal{O}(n^2)$.[4] Effectively these values differ by two to three orders of magnitude, as constant factors with sorting are tiny. With a linear scan, finding the nearest neighbors is in $\Theta(n^2)$. Many implementations will also require $\Theta(n^2)$ memory because of computing a full distance matrix.

[4] Results from computational geometry indicate that the worst case of nearest neighbor search in more than 3 dimensions cannot be better than $\mathcal{O}(n^{4/3})$ [16]. Empirical results with such indexes are usually much better, and tree-based indexes are often attributed a $n \log n$ cost for searching.

In particular on large data sets, finding the kNN is a fairly expensive operation, and traditional indexes such as the k-d tree and the R*-tree only work for low dimensionality (for our 27 dimensional example data set, the k-d tree has become twice as slow as the linear scan, and the R*-tree only yields small performance benefits, as opposed to 2 dimensions, where the speed-up was over 200 fold). Furthermore, neither the k-d tree nor the R*-tree are easy to parallelize in a cluster environment. Therefore, the use of approximate indexes is desirable to reduce runtime complexity.

While there are several attempts to optimize the neighborhood computation for outlier detection algorithms [23, 7, 27, 36, 3, 37, 46, 14, 43], these aim at computing the *exact* outlier score as fast as possible or at approximating the *exact* outlier score as closely as possible using approximate neighborhoods that are as close to the exact neighborhoods as possible. Note, however, that any outlier score is itself only an approximation of some imprecise statistical property and the "exact" outlier score is therefore an idealization that has probably no counterpart in reality.

Here, we argue that using approximate neighborhoods as such can be beneficial for outlier detection if the approximation has some bias that favors the isolation of outliers, especially in the context of ensemble techniques, that need some diversity among ensemble components anyway [48]. Using approximate neighborhoods as diverse components for outlier ensembles has not been discussed in the literature so far but it seems to be an obvious option. We show, however, that using the approximate neighborhoods can be beneficial or detrimental for the outlier detection ensemble, depending on the type of approximation. There are apparently good and bad kinds of neighborhood approximations for the task of outlier detection (and presumably also for clustering and for other data mining tasks). We take this point here based on preliminary results and suggest to investigate the bias of different neighborhood approximations methods further.

This paper is organized as follows: we review related work in Section 2, describe our approach in Section 3, and present our experimental results in Section 4. We conclude in Section 5.

2 Related Work

Existing outlier detection methods differ in the way they model and find the outliers and, thus, in the assumptions they, implicitly or explicitly, rely on. The fundamentals for modern, database-oriented outlier detection methods (i.e., methods that are motivated by the need of being scalable to large data sets, where the exact meaning of "large" has changed over the years) have been laid in the statistics literature. In general, statistical methods for outlier detection (also: outlier identification or rejection) are based on assumptions on the nature of the distributions of objects. The classical textbook of Barnett and Lewis [6] discusses numerous tests for different distributions. The tests are optimized for each distribution dependent on the specific parameters of the corresponding distribution, the number of expected outliers, and the space where to expect an

outlier. Different statistical techniques have been discussed by Rousseeuw and Hubert [40].

A broader overview for modern data mining applications has been presented by Chandola et al. [12]. Here, we focus on techniques based on computing distances (and derived secondary characteristics) in Euclidean data spaces.

With the first database-oriented approach, Knorr and Ng [26] triggered the data mining community to develop many different methods, typically with a focus on scalability. A method in the same spirit [39] uses the distances to the k nearest neighbors (kNN) of each object to rank the objects. A partition-based algorithm is then used to efficiently mine top-n outliers. As a variant, the sum of distances to all points within the set of k nearest neighbors (called the "weight") has been used as an outlier degree [4].

Aside from this basic outlier model, they proposed an efficient approximation algorithm, HilOut, based on multiple Hilbert-curves. It is a strongly database oriented technique capable of an efficient on-disk operation. It processes the data set in multiple scans over the data, maintaining an outlier candidate list and thresholds. For every point, its outlier score is approximated with an upper and lower point. Objects whose upper bound becomes less than the global lower bound can be excluded from the candidates. If after a certain number of scans the candidate set has not yet reached the desired size, a final refinement step will compute the pairwise distances from the candidates to the full data set. Hilbert-curves serve a twofold purpose in this method. First, they are used to find good neighbor candidates by comparing each object with its closest neighbors along the Hilbert-curve only. Second, the Hilbert-curves are used to compute lower bounds for the outlierness, as at least for a small radius they can guarantee that there are no missed neighbors. For subsequent scans, the Hilbert-curves are varied by shifting the data set with a multiple of $\frac{1}{d+1}$ on each axis to both create new neighbor candidates and to increase the chance of having a good guarantee on the close neighbor completeness.

The so-called "density-based" approaches consider ratios between the local density around an object and the local density around its neighboring objects, starting with the seminal LOF [10] algorithm. Many variants adapted the original LOF idea in different aspects [42]. Despite those many variants, the original LOF method is still competitive and state of the art [11].

As for other approaches, also for several of the variants of LOF, approximate variants have been proposed. For example the LOCI method [38] came already in the original paper with an approximate version, aLOCI. For aLOCI, the data are preprocessed and organized in (multidimensional) quadtrees. These have the benefit of allowing a simple density estimation based on depth and occupancy numbers alone, i.e., when an object is contained in an area of volume V which contains n objects, the density is estimated to be $\frac{n}{V}$. Since this estimation can be quite inaccurate when an object is close to the fringe of V, aLOCI will generate multiple shifted copies of the data set, and always use the quadtree area where the object is located most closely to the center. Furthermore, as aLOCI considers multiple neighborhood sizes, the algorithm will check multiple such boxes,

which may come from different trees. This makes the parallelization of aLOCI hard, while the required random accesses to the quadtree make this primarily an algorithm for data that fits into main memory. Shifting is done by moving the data set along a random vector in each dimension, cyclically wrapping the data within the domain (which may, in turn, cause some unexpected results).

Several approximate approaches use random projection techniques [33, 1, 45] based on the Johnson-Lindenstrauss lemma [24], especially in the context of high dimensional outlier detection [51]. Wang et al. [46] propose outlier detection based on Locality Sensitive Hashing (LSH) [22, 18, 13]. The key idea of this method is to use LSH to identify low-density regions, and refine the objects in these regions first, as they are more likely to be in the top-n global outliers. For local outlier detection methods there may be interesting outliers within a globally dense region, though. As a consequence, the pruning rules this method relies upon will not be applicable. Zhang et al. [47] combine LSH with isolation forests [32]. Projection-indexed nearest-neighbours (PINN) [14] shares the idea of using a random projection to reduce dimensionality. On the reduced dimensionality, an exact spatial index is then employed to find neighbor candidates that are refined to k nearest neighbors in the original data space.

Improving efficiency of outlier detection often has been implemented by focussing on the top-n outliers only and pruning objects before refinement that do not have a chance to be among the top-n outliers [23, 7, 27, 36, 3]. A broad and general analysis of efficiency techniques for outlier detection algorithms [37] identifies common principles or building blocks for efficient variants of the so-called "distance-based" models [26, 39, 4]. The most fundamental of these principles is "approximate nearest neighbor search" (ANNS). The use of this technique in the efficient variants studied by Orair et al. [37] is, however, different from the approach we are proposing here in a crucial point. Commonly, ANNS has been used as a filter step to discard objects from computing the *exact* outlier score. The exact kNN distance could only become smaller, not larger, in case some neighbor was missed by the approximation. Hence, if the upper bound of the kNN distance, coming along with the ANNS, is already too small to possibly qualify the considered point as a top-n outlier, the respective point will not be refined. For objects passing this filter step, the *exact neighborhood* is still required in order to compute the *exact outlier score*. All other efficiency techniques, as discussed by Orair et al. [37], are similarly based on this consideration and essentially differ in the exact pruning or ranking strategies. As opposed to using approximate nearest neighborhoods as a filter step, we advocate to *directly* use the resulting set of an approximate nearest neighbor search to compute outlier scores, without any refinement. Schubert et al. [43] based a single outlier model on combinations of several approximate neighborhoods, studying space-filling curves and random projections. Here, we compute the outlier score on each of the k *approximate* nearest neighbors directly, without any refinement, instead of on the exact neighborhood and combine them only afterwards. In addition, we compare different approximate neighborhood search methods: aside from space filling curves we also study LSH (see above) and NN-Descent [15]. The basic

idea of NN-Descent is an iterative refinement of neighborhoods, checking the *approximate* neighbors of the *approximate* neighbors (using both forward and reverse neighborhoods). Starting from random neighborhoods, the iteration approximates surprisingly quickly and well the true neighborhoods.

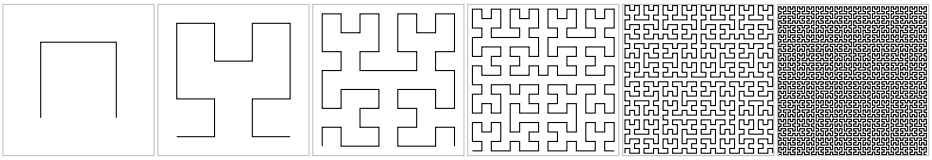
Many more approaches for the computation of approximate neighborhoods could be tested and compared on their suitability for outlier detection (as well as for other data mining tasks). However, most of them focus on a near-perfect recall, and therefore may be unsuitable for our purposes. K-d-trees can be parameterized to give approximation guarantees even when not exploring all branches [5]. For example, randomized k-d-trees [44] build multiple k-d-trees (randomly choosing the split axis amongst the best candidates) and search them in parallel with an approximate search, while the priority search k-means tree [35] uses recursive clustering.

Isolation forests [32] can be seen as an approximate density estimation ensemble, which constructs multiple trees on different samples of the data, where the height of a leaf (which determines "isolation") is implicity used as a kind of density estimate. As it does not find neighbors, but directly estimates density, it cannot be used with methods such as LOF. ALOCI [38] uses a quadtree for a similar purpose. Nevertheless, the idea of building an ensemble of simple outlier detectors is a common idea with our approach, and our observations may yield further insight into this method, too.

Our work here is thus to be seen as a first step towards embracing imprecision of approximate nearest-neighbor search as a source of diversity for ensemble construction.

3 Outlier Detection Ensembles Based on Approximate Neighborhoods

The conclusion we draw from the discussion of related work is to emphasize that, for certain outlier detection models, it does not seem to be of the utmost importance to work on exact neighborhoods. Although the use of approximate neighborhoods for outlier detection was usually an intermediate step, before ultimately neighborhoods are refined to be exact or at least as good as possible, we maintain that approximate neighborhoods can be sufficient or even beneficial (if the approximations exhibit a bias that favors the isolation of outliers), to estimate and compare local densities, in particular if we combine outlier models learned on approximate data to an ensemble. The same reasoning relates to several existing ensemble methods for outlier detection [48], where a better overall judgment is yielded by diversified models. Models are diversified using approximations of different kinds: the results for outlier detection ensembles have been improved by computing neighbors in subsets of features [30], in subsets of the dataset [50], or even by adding *noise* components to the data points in order to yield diverse density-estimates [49]. All these variants can in some sense also be seen as using approximate neighborhoods directly for density estimates (in subspaces, on subsets, or on noisy data), and for some of these approximations



(a) Depth 1 (b) Depth 2 (c) Depth 3 (d) Depth 4 (e) Depth 5 (f) Depth 6

Fig. 1: Hilbert curve approximations at different recursion depth.

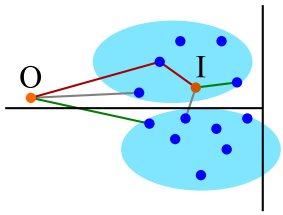


Fig. 2: Approximation error caused by a space filling curve (illustration): black lines indicate neighborhoods not preserved by the space filling curve. Shaded areas are discovered clusters, red lines are approximate 2NN distances, green lines are the real 2NN distances. By the loss of true neighbors, the density estimated based on approximate neighbors will have a stronger tendency to be underestimated for outliers than for cluster points, where the distances do not grow that much by missing some true neighbors.

it has been argued why the particular approximation technique could even prove beneficial for increasing the gap between outlier and inlier scores [50].

Among neighborhood approximation methods, space-filling curves have a particular property w.r.t. outliers that seems to act beneficial. A space-filling curve is recursively cutting the space as visualized in Figure 1. Neighbors being close in the full space but being separated by such a cut will not be well preserved. In Figure 2, we showcase why this is of minimal effect on density estimates within a cluster, while the density around outliers is more likely to be underestimated more strongly: losing some neighbor in a low-density area (as around outliers) will incur the identification of approximate neighbors that exhibit larger distances (and thus much smaller local density estimates for the outlier) as compared to losing some neighbor in some high-density area (such as a cluster), where the approximate neighbors will still be rather close. Space-filling curves do exhibit a bias that is actually helpful for outlier detection.

Neither LSH nor NN-Descent have a similar bias favoring relative underestimation of density around outliers. By using reverse neighborhoods together with forward neighborhoods, NN-Descent is naturally adaptive to different local densities. Kabán [25] pointed out that random projection methods according to the Johnson-Lindenstrauss lemma preserve distances approximately and thus also

preserve the distance concentration. Accordingly, LSH, being based on random-projections, tends to preserve distances without bias on higher or lower densities.

In an outlier ensemble setting, we propose to use some basic outlier detector that takes local neighborhoods as input (context set) to compute some local model and that compares this model with the model of the neighbors as reference set. Context set and reference set are not necessarily identical, but typically they are. See the discussion by Schubert et al. [42] on the general design of local outlier detection methods. As we have seen in the overview on related work, typically exact neighborhoods are used. However, in ensemble approaches [48] often special techniques are applied to diversify the models, e.g. by using neighborhood computations in subspaces [30], in subsets of the data [50], or after adding a noise component on the data [49].

Here we propose to not artificially diversify exactly computed neighborhoods but rather to stick to approximate neighborhoods in the first place, which comes obviously with a considerable computational benefit in terms of efficiency. We demonstrate that this approach can also come with a considerable benefit in terms of effectiveness, although this depends on the approximation method chosen. We conjecture that also different outlier detection methods used as ensemble components might react differently to the use of appoximations. In this study, however, we focus on the sketched approximation techniques (space-filling curves, LSH, and NN-Descent) in building outlier ensembles, using LOF [10] as basic outlier detection technique. Outlier scores computed on various approximations are then combined with standard procedures [48], using score normalization [28] and ranking of average scores.

4 Experiments

For experiments, we use LOF as well as the neighborhood approximation methods in the implementation available in the ELKI framework [41].

As data set, we use a 27 dimensional color histogram representation of the Amsterdam Library of Object Images (ALOI) [17], as used before in the outlier detection literature (cf. the collection of benchmark data by Campos et al. [11] and previous usage documented therein). We also take orientation on the results reported by Campos et al. [11] for parameter selection (neighborhood size for LOF), where values larger than 20 do not seem to be beneficial on this data set. We thus test $k = 1, \ldots, 20$.

As space filling curves we use the Z-order [34] and a window size equal to the number of requested neighbors as used by Schubert et al. [43]. We chose the simplest curve because it produces more diversity, and the Hilbert curve [20] is substantially more expensive to compute, although recently some progress has been made on sorting data without transforming it to Hilbert indices [21]. For LSH we use 3 projections based on p-stable distributions [13], 3 hash tables and a projection width of 0.1. For NN-Descent, we restrict the number of iterations to 2 in order to force some diversity in the results. All of these parameters are deliberately chosen to provide a fast, and not overly precise result. Too

precise results will obviously be detrimental to building an ensemble afterwards, as ensembles rely on diversity in the ensemble members.

We measure the recall of the delivered neighborhoods (not counting the query point itself) as well as the performance of the outlier detection methods (LOF with exact neighborhoods and ensemble of LOF on approximate neighborhoods) in terms of the area under the ROC curve (ROC AUC).

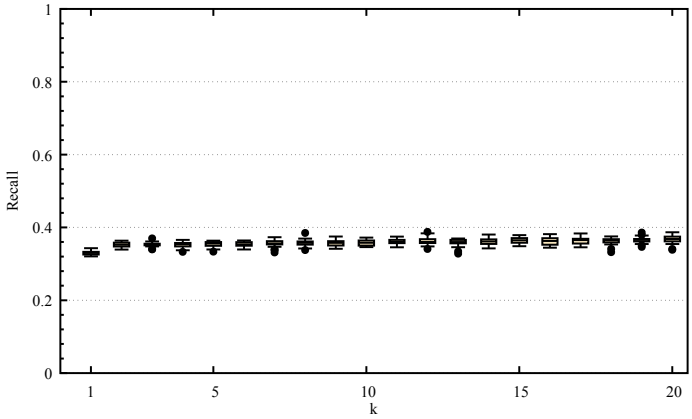
In Figure 3, we depict the recall of the three approximation methods as distribution over 25 runs, varying the size k of the requested neighborhood. For NN-Descent we see a strong tendency to achieve better recall for larger neighborhoods (due to the larger neighbors-of-a-neighbor candidate set). Z-order only shows a slight tendency in the same direction, LSH has the opposite tendency, however not very strongly (because of the increasing distance to the k nearest neighbor, these are less likely to be in the same hash bucket). More remarkable is the difference in the variance of achieved recall: Z-order always has a considerable variance, the variance in LSH seems also to depend on the neighborhood size, while NN-Descent has very stable recall over the different runs. If we allowed NN-Descent to perform more iterations, its recall would further improve, but the variance would become even smaller. Note that for the purpose of ensemble method, variance is related to diversity, and therefore desirable.

If we were to compare the approximation methods as such, we would easily notice that LSH achieves very high recall compared to the others, and therefore may be considered to be the best choice. However, as we want to use approximate neighborhoods as input for ensemble members, a low recall might already be sufficient to get good results [43] and the variance is of greater importance.

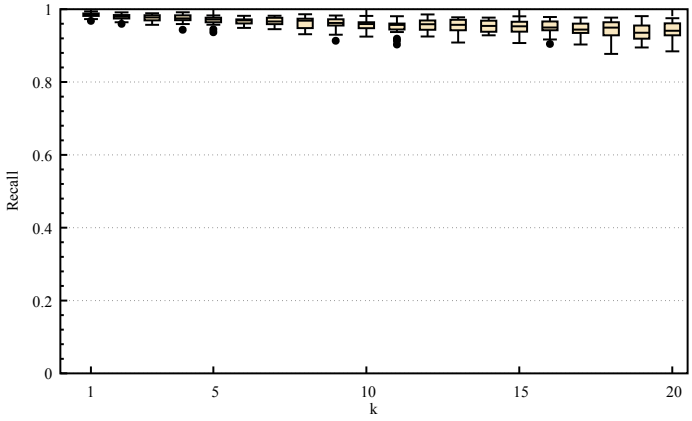
In Figure 4 we depict the performance of the resulting ensembles, based on each of the approximation methods and each k. We plot the score distribution of the individual ensemble members using a boxplot, and the performance of the ensemble resulting from the combination. In order to visualize the relationship to recall of the true nearest neighbors, we use the mean recall of the ensemble on the x axis. For comparison, at recall 1, we also plot the results obtained with exact nearest neighbors (multiple points due to multiple choices of k).

For all of the methods, we can observe that the ensemble performs at least as good as 75% of the ensemble members, indicated by the upper quartile of the boxplot. As expected, we see that the combination of LOF based on LSH (high recall) and NN-Descent (low recall), both not exhibiting a beneficial bias for outlier detection, does not improve over the single LOF result based on exact neighborhoods,[5] while the combination of LOF based on space-filling curve approximations (intermediate recall, large variance, beneficial bias) improves also over the exact LOF and shows the best results overall, similar to the observation by Schubert et al. [43].

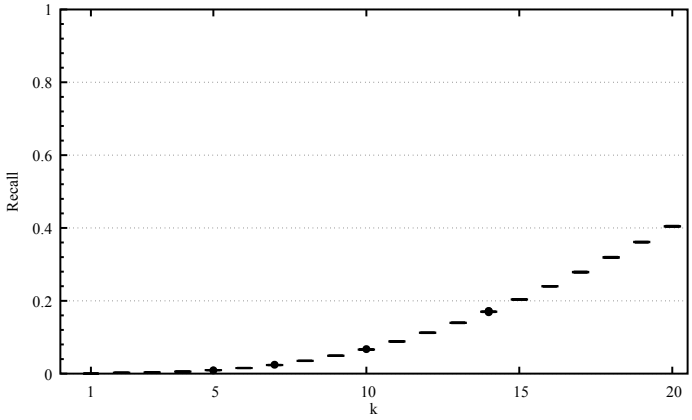
[5] But there may be a performance improvement by nevertheless using these methods.



(a) Z-order

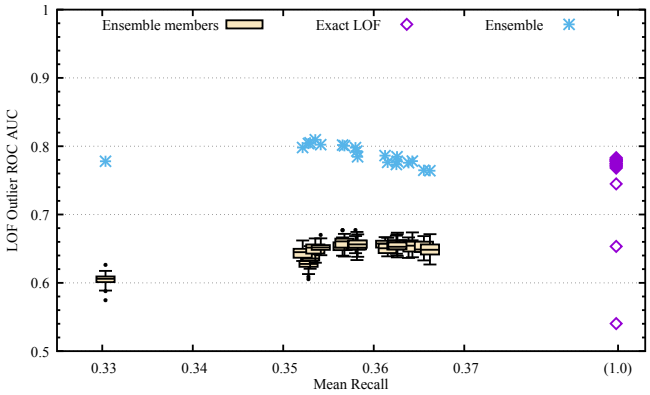


(b) LSH

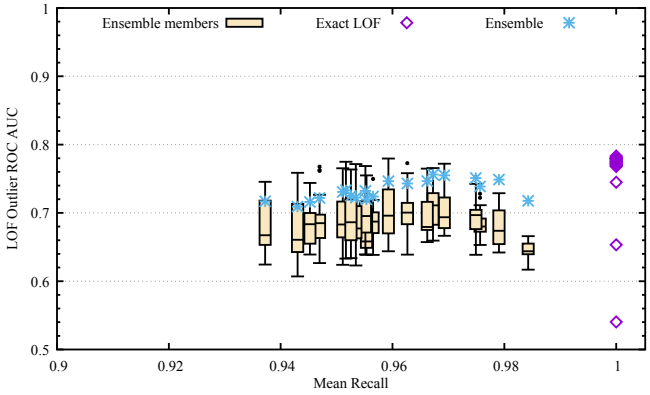


(c) NN-Descent

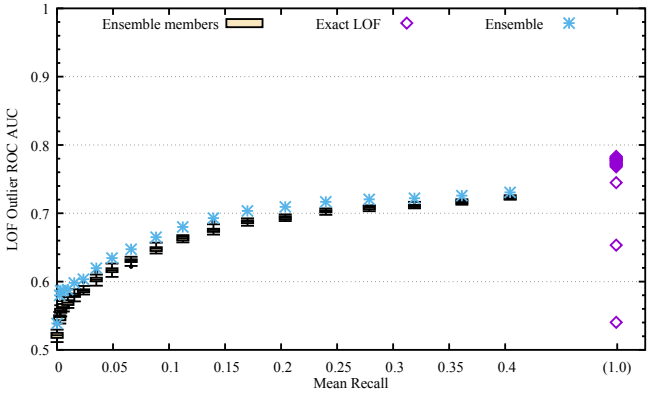
Fig. 3: Recall of the true k nearest neighbors for approximate neighborhood search (distribution over 25 runs), depending on the neighborhood size k. (The query point is not counted as hit in the result.)



(a) Z-order



(b) LSH



(c) NN-Descent

Fig. 4: ROC AUC of ensemble members (LOF on approximations), ensemble, and exact LOF for different $k = 1 \ldots 20$ (not labeled). Boxplots indicate the ensemble members, the stars indicate the performance of the complete ensemble, diamonds indicate the performance of exact nearest neighbors for comparison.

5 Conclusion

We studied outlier detection ensembles based on approximate neighborhoods, using LOF as outlier detector and space-filling curves (Z-order), LSH, and NN-Descent as approximate methods of nearest neighbor search. Our results demonstrate that higher recall in the neighborhood search is not necessarily better for building ensembles, as for building ensembles, the variance over the ensemble members is an important ingredient. And indeed, in theory, a method with 0% recall in the true k-nearest-neighbors can nevertheless achieve 100% accuracy in finding the true outliers. The neighborhood approximation with intermediate recall, Z-order, delivers the best results for the outlier ensemble, beating exact methods. NN-Descent (with only 2 iterations to have more diversity) reaches from very poor recall to a slightly better recall, compared to Z-order. The recall here is clearly depending on the size of the requested neighborhood (as expected from the nature of the approximation method). But the variance is surprisingly small and does not give sufficient variety to improve in an ensemble. LSH, on the other hand, shows a very strong performance in terms of recall. The performance of the outlier ensemble is in the upper half of the distribution of the individual outlier detectors based on individual approximations, but does not reach the performance of the exact method. For the purpose of using this for outlier detection ensembles, a key challenge is to construct approximation that are both good enough, and diverse enough.

We offer as an additional explanation that space-filling curves exhibit a bias that is particularly helpful to distinguish low-density areas (i.e., outliers) from high-density areas (i.e., clusters). We therefore suggest to study more thoroughly the bias of different neighborhood approximation methods with respect to different application tasks.

References

1. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. JCSS 66, 671–687 (2003)
2. Achtert, E., Kriegel, H.P., Schubert, E., Zimek, A.: Interactive data mining with 3D-Parallel-Coordinate-Trees. In: Proc. SIGMOD. pp. 1009–1012 (2013)
3. Angiulli, F., Fassetti, F.: DOLPHIN: an efficient algorithm for mining distance-based outliers in very large datasets. ACM TKDD 3(1), 4:1–57 (2009)
4. Angiulli, F., Pizzuti, C.: Outlier mining in large high-dimensional data sets. IEEE TKDE 17(2), 203–215 (2005)
5. Arya, S., Mount, D.M.: Approximate nearest neighbor queries in fixed dimensions. In: Proc. SODA. pp. 271–280 (1993)
6. Barnett, V., Lewis, T.: Outliers in Statistical Data. John Wiley&Sons, 3rd edn. (1994)
7. Bay, S.D., Schwabacher, M.: Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: Proc. KDD. pp. 29–38 (2003)
8. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: An efficient and robust access method for points and rectangles. In: Proc. SIGMOD. pp. 322–331 (1990)

9. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM 18(9), 509–517 (1975)
10. Breunig, M.M., Kriegel, H.P., Ng, R., Sander, J.: LOF: Identifying density-based local outliers. In: Proc. SIGMOD. pp. 93–104 (2000)
11. Campos, G.O., Zimek, A., Sander, J., Campello, R.J.G.B., Micenková, B., Schubert, E., Assent, I., Houle, M.E.: On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. Data Min. Knowl. Disc. 30, 891–927 (2016)
12. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM CSUR 41(3), Article 15, 1–58 (2009)
13. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. ACM SoCG. pp. 253–262 (2004)
14. de Vries, T., Chawla, S., Houle, M.E.: Density-preserving projections for large-scale local anomaly detection. KAIS 32(1), 25–52 (2012)
15. Dong, W., Charikar, M., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proc. WWW. pp. 577–586 (2011)
16. Erickson, J.: On the relative complexities of some geometric problems. In: Proceedings of the 7th Canadian Conference on Computational Geometry, Quebec City, Quebec, Canada, August 1995. pp. 85–90 (1995)
17. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The Amsterdam Library of Object Images. Int. J. Computer Vision 61(1), 103–112 (2005)
18. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proc. VLDB. pp. 518–529 (1999)
19. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proc. SIGMOD. pp. 47–57 (1984)
20. Hilbert, D.: Ueber die stetige Abbildung einer Linie auf ein Flächenstück. Math. Ann. 38(3), 459–460 (1891)
21. Imamura, Y., Shinohara, T., Hirata, K., Kuboyama, T.: Fast hilbert sort algorithm without using hilbert indices. In: Proc. SISAP. pp. 259–267 (2016)
22. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proc. STOC. pp. 604–613 (1998)
23. Jin, W., Tung, A.K., Han, J.: Mining top-n local outliers in large databases. In: Proc. KDD. pp. 293–298 (2001)
24. Johnson, W.B., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. In: Conference in Modern Analysis and Probability, Contemporary Mathematics, vol. 26, pp. 189–206. American Mathematical Society (1984)
25. Kabán, A.: On the distance concentration awareness of certain data reduction techniques. Pattern Recognition 44(2), 265–277 (2011)
26. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proc. VLDB. pp. 392–403 (1998)
27. Kollios, G., Gunopulos, D., Koudas, N., Berchthold, S.: Efficient biased sampling for approximate clustering and outlier detection in large datasets. IEEE TKDE 15(5), 1170–1187 (2003)
28. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Interpreting and unifying outlier scores. In: Proc. SDM. pp. 13–24 (2011)
29. Kriegel, H.P., Schubert, E., Zimek, A.: The (black) art of runtime evaluation: Are we comparing algorithms or implementations? KAIS pp. 1–38 (2016)
30. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: Proc. KDD. pp. 157–166 (2005)

31. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web J. (2014)
32. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM TKDD 6(1), 3:1–39 (2012)
33. Matoušek, J.: On variants of the Johnson–Lindenstrauss lemma. Random Structures & Algorithms 33(2), 142–156 (2008)
34. Morton, G.M.: A computer oriented geodetic data base and a new technique in file sequencing. Tech. rep., International Business Machines Co. (1966)
35. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE TPAMI 36(11), 2227–2240 (2014)
36. Nguyen, H.V., Gopalkrishnan, V.: Efficient pruning schemes for distance-based outlier detection. In: Proc. ECML PKDD. pp. 160–175 (2009)
37. Orair, G.H., Teixeira, C., Wang, Y., Meira Jr., W., Parthasarathy, S.: Distance-based outlier detection: Consolidation and renewed bearing. PVLDB 3(2), 1469–1480 (2010)
38. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: LOCI: Fast outlier detection using the local correlation integral. In: Proc. ICDE. pp. 315–326 (2003)
39. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Proc. SIGMOD. pp. 427–438 (2000)
40. Rousseeuw, P.J., Hubert, M.: Robust statistics for outlier detection. WIREs DMKD 1(1), 73–79 (2011)
41. Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K.A., Zimek, A.: A framework for clustering uncertain data. PVLDB 8(12), 1976–1979 (2015)
42. Schubert, E., Zimek, A., Kriegel, H.P.: Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. Data Min. Knowl. Disc. 28(1), 190–237 (2014)
43. Schubert, E., Zimek, A., Kriegel, H.P.: Fast and scalable outlier detection with approximate nearest neighbor ensembles. In: Proc. DASFAA. pp. 19–36 (2015)
44. Silpa-Anan, C., Hartley, R.I.: Optimised kd-trees for fast image descriptor matching. In: Proc. CVPR (2008)
45. Venkatasubramanian, S., Wang, Q.: The Johnson-Lindenstrauss transform: An empirical study. In: Proc. ALENEX Workshop (SIAM). pp. 164–173 (2011)
46. Wang, Y., Parthasarathy, S., Tatikonda, S.: Locality sensitive outlier detection: A ranking driven approach. In: Proc. ICDE. pp. 410–421 (2011)
47. Zhang, X., Dou, W., He, Q., Zhou, R., Leckie, C., Kotagiri, R., Salcic, Z.: LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis. In: Proc. ICDE (2017)
48. Zimek, A., Campello, R.J.G.B., Sander, J.: Ensembles for unsupervised outlier detection: Challenges and research questions. SIGKDD Explor. 15(1), 11–22 (2013)
49. Zimek, A., Campello, R.J.G.B., Sander, J.: Data perturbation for outlier detection ensembles. In: Proc. SSDBM. pp. 13:1–12 (2014)
50. Zimek, A., Gaudet, M., Campello, R.J.G.B., Sander, J.: Subsampling for efficient and effective unsupervised outlier detection ensembles. In: Proc. KDD. pp. 428–436 (2013)
51. Zimek, A., Schubert, E., Kriegel, H.P.: A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Min. 5(5), 363–387 (2012)