AnyCORE - An Anytime Algorithm for Cluster Outlier REmoval

Andreas Lohrer[1], Anna Beer[2], Maximilian Hünemörder[1], Jennifer Lauterbach[2], Thomas Seidl[2] and Peer Kröger[1]

[1]*Christian-Albrechts-University Kiel, Germany, Christian-Albrechts-Platz 4, 24118 Kiel, Germany*
[2]*Ludwig-Maximilians-Universität München, Germany, Oettingenstraße 67, 80538 München, Germany*

Abstract
We introduce AnyCORE (Anytime Cluster Outlier REmoval), an algorithm that enables users to detect and remove outliers at anytime. The algorithm is based on the idea of MORe++[1], an approach for outlier detection and removal that iteratively scores and removes $1d$-cluster-outliers in n-dimensional data sets. In contrast to MORe++, AnyCORE provides continuous responses for its users and converges independent of cluster centers. This allows AnyCORE to perform outlier detection in combination with an arbitrary clustering method that is most suitable for a given data set. We conducted our AnyCORE experiments on synthetic and real-world data sets by benchmarking its variant with k-Means as the underlying clustering method versus the traditional batch algorithm version of MORe++. In extensive experiments we show that AnyCORE is able to compete with the related batch algorithm version.

Keywords
anytime algorithm, outlier removal, outlier detection, clustering

1. Introduction

In many domains, e.g., in sports, health, or mobility, data-driven applications and the connected business processes benefit from algorithms capable of detecting and removing outliers at any time. However, since most outlier detection algorithms work in batch mode[1] and the amounts of most commonly massive and high-dimensional data sets are likely to be computationally expensive, applications and users face long response times, avoiding to take fast decisions. Furthermore, outliers in the data set can have a distortive impact on the accuracy of results and cause additional unnecessary computational efforts. As digitalization is essential for various domains and business areas, an increase in the amount of structured and unstructured data generated by related applications is the typical consequence. Leveraging these amounts of data means finding insightful patterns which can be used for subsequent information reasoning or to derive recommendations for further actions. Such patterns can be, e.g., clusters of similar data points or anomalous data points, which we call outliers. Finding clusters and identifying

LWDA'21: Lernen, Wissen, Daten, Analysen September 01–03, 2021, Munich, Germany
alo@informatik.uni-kiel.de (A. Lohrer); beer@dbs.ifi.lmu.de (A. Beer); mah@informatik.uni-kiel.de (M. Hünemörder); jenny.lauterbach@fs.lmu.de (J. Lauterbach); seidl@dbs.ifi.lmu.de (T. Seidl); pkr@informatik.uni-kiel.de (P. Kröger)
0000-0001-7834-301X (A. Lohrer); 0000-0002-4861-1412 (T. Seidl); 0000-0001-5646-3299 (P. Kröger)

[1]returning results only if the computation has been completely finished

outliers are tasks that become increasingly cumbersome, in terms of runtime, with an increasing number of observations and feature dimensions. In addition to that, noise observations that are neither part of any cluster nor a significant outlier are not only irrelevant for data analysts[2], they can also have negative effects on the results of clustering algorithms. Thus, excluding them can optimize the resulting cluster structures. Since clustering and outlier removal are already challenging tasks for low-dimensional data, batch mode[1] algorithms cannot provide the related responses at any time to its users for nowadays often large or high-dimensional data.

Thus, we introduce in this paper AnyCORE, an **Any**time **C**luster **O**utlier **RE**moval algorithm, which allows users to request intermediate results at any time, even before the algorithm has finished. Optimizing the algorithm MORe++[1] our algorithm not only alternates between the originally consecutive clustering and outlier removal steps of MORe++, it also provides intermediate results at any time while allowing the user to select clustering and outlier decision methods most suitable according to the provided data set.

The main contributions of this paper are:

- The *k*-Means Outlier Removal algorithm gets enhanced to provide users response with intermediate results at any time during its runtime.
- AnyCORE utilizes removed outliers to get more precise clustering results by alternating between clustering and outlier removal.
- AnyCORE allows users to interrupt, to check the results, and to stop the algorithm at any time if the results are considered as satisfactory.
- The user can select the underlying clustering and outlier scoring method suitable to the analyzed data set.

The remaining paper is structured as follows. Section 2 describes related work for anytime algorithms, clustering and outlier removal algorithms. In Section 3, we describe our algorithm AnyCORE and evaluate its competitiveness in Section 4. Section 5 concludes the paper.

2. Related Work

In this section, we provide an overview of the related literature. This includes the description of related anytime clustering and outlier detection algorithms, as well as the introduction of the underlying outlier removal algorithm MORe++[1] which got optimized by our AnyCORE approach.

MORe++. MORe++[1] is a k-**M**eans++ based algorithm for **O**utlier **Re**moval which is also capable to perform on high-dimensional data. It performs *k*-Means++[3] until convergence to unify similar observations within clusters. For the subsequent step of outlier detection the algorithm analyzes the given samples in each single dimension by creating 1d-projections separately for each single cluster (see Figure 1). The resulting histograms of each cluster in each single dimension allow even visually the identification of outliers by gaps or significant outlying rises or falls among its frequencies. Counting the single dimensions in which a sample got detected as cluster outlier allows one to calculate an overall outlier score for a sample and

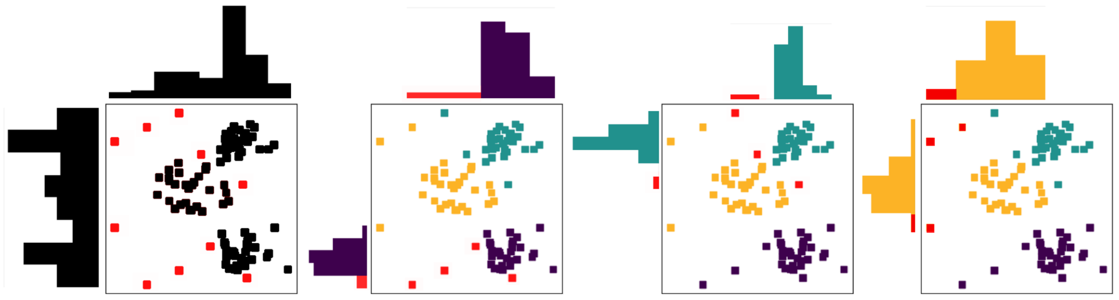

Figure 1: This illustration visualizes outliers (red) detected by MORe++ [1] during its clusterwise 1-dimensional outlier analysis. The black histogram on the left shows the aggregated result of the single cluster-based outlier analysis on the right. (based on Fig. 1.[1])

distinguish normal from abnormal samples by a given threshold. This enables the algorithm to flag samples as outliers and additionally for their removal.

Since MORe++ does not consider the beneficial effect of outlier removals upfront of its clustering step we address this potential within our AnyCORE approach in Section 3.

***k*-means−−** The concept of combining clustering and outlier detection by removing outliers before the clustering step was also introduced by, e.g., *k*-Means−−[4]. The authors provide an extension to *k*-Means, where a fixed number l of points are removed in each iteration, before the centroids are computed. This is achieved by ranking the points in the input data set X by distance from the current centroids. Then the next centroids are computed from the set difference of X and the l points furthest away from the centroids. For the next iteration the furthest points are recomputed and again removed from the original data set X. At convergence the algorithm results in a set of cluster centers and a set of exactly l outliers. Note that *k*-Means−− uses the removal of the furthest points in order to optimize clustering results of *k*-Means, where our algorithm AnyCORE is designed to find outliers in high-dimensional space, working independently of the cluster algorithm. AnyCORE does not need necessarily any cluster centroids or means as provided by *k*-Means. Due to the curse of dimensionality, the Euclidean distance used by *k*-Means−− can not be used to discern distances to the cluster centers for very high dimensionalities any more, where our histogram-based solution is suitable also for large numbers of attributes in the data set.

Anytime algorithms. The concept of anytime algorithms has already been introduced in 1988 by Dean & Body [5, 6] which follows the approach that an algorithm can be stopped at anytime during computation in order to quickly provide its users intermediate results with a reasonable result quality. This allows the users to decide if the given results are satisfactory or if the processing of the algorithm should be resumed in order to receive a further improved result. The quality of early results should be close to the quality of a batch algorithm and the final result should be at least as good or even better than the batch algorithm. The runtime of an anytime algorithm should not be much larger than the batch algorithm's runtime [7, 8, 9]. This behavior is illustrated in Figure 2.

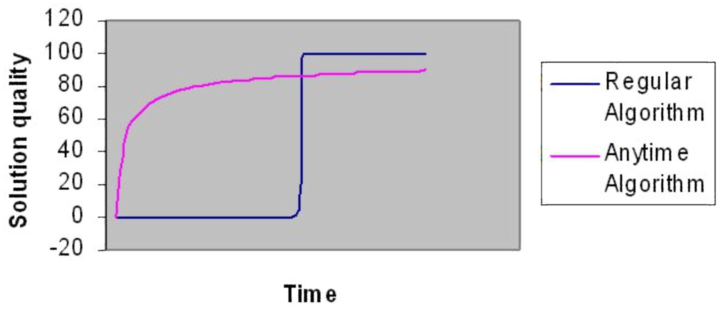

Figure 2: Comparison of the result quality in relation to the runtime for an anytime and batch algorithm.

Since this approach of approximated processing [10] has been proven to be useful for AI applications [7] it has been evolved and widely adapted to various tasks in the field of knowledge discovery. This led to developments like, e.g., AnyDBC [11, 12], which is a DBSCAN-based anytime clustering algorithm that can be used on high-dimensional data sets like time series or trajectories. Further developments are AnyHC [13], an anytime algorithm for hierarchical clustering, and different variants of A-DBSCAN [14, 15] which deliver results based on approximated density-based clusters. I-kMeans [16] works with Discrete and Haar Wavelet Transformation as an anytime *k*-Means on time series and was extended with Piecewise Constant Approximation (PAA) in [17]. These algorithms are just a few well known representants of anytime algorithms, which mainly perform clustering on data streams. Besides the tasks of clustering and anytime embedding [18], anytime outlier detection has also been addressed with algorithms like, e.g., AnyOut [19], which uses the data structure ClusTree [20] to analyze constant and changing data streams. In addition to that, there are approaches like FPOF [21] for frequent pattern outlier detection on massive data sets or the anytime novelty detection method of Sofman [22].

This selection of anytime algorithms gets extended by our proposed method AnyCORE which is capable to conduct clustering as well as outlier detection and removal within one iteration so that its users are constantly able to stop, interrupt and request intermediate results at any time.

3. AnyCORE

In this section we introduce our algorithm AnyCORE (**Any**time **C**luster **O**utlier **RE**moval) and highlight its differences to the underlying algorithm MORe++[1] in detail. Variables and methods from the pseudo code of Algorithm 1 are referenced and formatted in *italics*.

Revisiting the MORe++ approach, which we described in detail in Section 2, we observe that the removal of outliers holds further potential for optimization. This potential gets revealed by our method utilizing the beneficial effects of removed outliers in the clustering phase, which is without outliers more likely to calculate correct cluster labels, as, e.g., already applied in *k*-Means−− introduced in Section 2. This gets achieved by alternating between clustering and outlier removal instead of running them consecutively. In addition to that an outlier score threshold *ost* is required. The threshold is needed, because AnyCORE temporarily removes outliers from the data set to get a more precise result. Since the convergence criteria of the percentage of cluster jumpers[2] is independent of the convergence and availability of cluster centers, AnyCORE is able to run with arbitrary clustering methods and arbitrary outlier scoring methods, which allows to chose those methods which are most suitable for the provided data set. Hence the last input parameters of AnyCORE are the selected clustering algorithm $algo$ with its related parameter set p and the selected outlier scoring method *outlier_scoring*. The optional parameter *init* can be used to initialize centers in case of $algo$ is centroid-based but also to e.g. provide an initial ϵ-range, which can be adapted according to 1d-gap-sizes in case of $algo$ is density-based. For our experiments in Section 4 we evaluated our method with the clustering algorithms k-Means++ (p=k, *init*='k-means++') as well as with the histogram-based outlier-scoring method of MORe++ which considers the ratio of outlier dimensions of a sample as outlier score (parameter *outlier_scoring*). AnyCORE can be interrupted by users at any time to get an intermediate result. If users are not satisfied with the outcome, they can either stop the algorithm and start again with other input parameters or continue the current calculation to get a refined result. For the calculation of intermediate results, the outlierScore or the $numberOfOutlierDims$ is needed as it changes the outcome of the *outlierScore*. Thus the result of AnyCORE can change whenever $numberOfOutlierDims$ is updated after each calculation of the 1*dOutliers*. Only if users stop AnyCORE during the first iteration, an intermediate *outlierScore* with the newest value of $numberOfOutlierDims$ gets returned without previously removed outliers. As in each iteration outliers and the $numberOfOutlierDims$ are newly calculated and as up from the second iteration an refined initialization by outlier removal is applied, one faces potentially also partly incorrect cluster memberships. So it can be possible for interim results to get intermediately worse than previous interim results. The pseudo-code of the illustrated Algorithm 1 describes the sequences of AnyCORE. Lines 9-27 represent the part of MORe++ but with the above described AnyCORE-specific adaptions highlighted by the prefix symbol #.

4. Experiments

In this section we describe our experimental setup as well as the results of our experiments demonstrating the competitiveness of our method AnyCORE.

Datasets. The experiments for benchmarking the AnyCORE approach have been conducted on synthetic as well as on real world data sets. Since massive real world data sets are scarcely available for outlier detection, all used ODDS[3] data sets described in the Table 1 had been evaluated despite of their short runtime. Additionally to those data sets, we also experimented with two constructed synthetic data sets, SynthA and SynthB, in order to investigate how Any-CORE performs on large data sets. Both synthetic data sets had been constructed using cluster centers drawn from a uniform distribution function and generating gaussian-distributed clusters around them (parameters: centers=5, center_box=(0, 40), cluster_std=1.2, random_state=3). In

[2]samples which changed their cluster membership between the previous and current iteration of AnyCORE
[3]http://odds.cs.stonybrook.edu/

Algorithm 1: Pseudo-Code of AnyCORE

```
Data: Data $X$, #clustering algorithm $algo$, clustering algorithm parameters $p$ ⟵{number
      of clusters $k$}, outlier score threshold $ost$, #scoring method $outlier\_scoring$,
      clustering initialization $init$ ⟵{'k-means++'}
Result: Clustering labels, $outlierScore$ for data points
$t\_start \leftarrow now$;
$outliers \leftarrow$ empty list;
$clusterMembershipsOld \leftarrow$ empty list;
while $not\ converged\ and\ not\ stopped$ do
    if $outliers <> empty\ list$ then
        $clusterMemberships, init \longleftarrow \#algo(X \setminus outliers, p, init)$ ;
    end
    $clusterMemberships, init \longleftarrow \#algo(X, p, init)$ ;
    foreach $x \in X$ do
        $numberOfOutlierDims[x] \leftarrow 0$ ;
    end
    foreach $c \in clusters$ do
        foreach $d \in range(dimensions)$ do
            $histogram \leftarrow buildHistogram(c, d)$;
            $1dOutliers \leftarrow calculate1dOutliers(histogram)$;
            foreach $1dOutlier \in 1dOutliers$ do
                $numberOfOutlierDims[1dOutlier] + +$;
            end
        end
    end
    $outliers \leftarrow$ empty list;
    foreach $x \in X$ do
        $outlierScore[x] \leftarrow \#outlier\_scoring(numberOfOutlierDims[x])$ ;
        if $outlierScore[x] >= ost$ then
            $outliers.add(x)$;
        end
    end
    $timestamp \leftarrow (now - t\_start)$ ;
    publish $timestamp, outlierScore$ ;
    while $interrupted$ do
        sleep
    end
    $converged = (perc\_changed(clusterMemberships, clusterMembershipsOld) == 0)$ ;
    $clusterMembershipsOld \leftarrow clusterMemberships$ ;
end
```

order to be able to properly benchmark the outlier detection and removal part of the compared algorithms, these data sets had been additionally extended by outliers, which also follow an uniform distribution function (parameters: low=0, high=40, seed=5).

Dataset	n	dim	outlier	k	type
Glass	214	9	9 (4.20%)	5	real world
Lympho	148	18	6 (4.10%)	2	real world
Mnist	7603	100	700 (9.20%)	2	real world
Musk	3062	166	97 (3.20%)	3	real world
Optdigits	5216	64	150 (3.00%)	9	real world
Shuttle	49097	9	3511 (7.00%)	1	real world
Smtp	95156	3	30 (0.03%)	1	real world
SynthA	100000	5	5000 (5.00%)	5	synthetic
SynthB	1000	3000	50 (5.00%)	5	synthetic

Table 1: ODDS[3] real world data sets and synthetic data sets used within the experiments.

Compared methods and experimental setup. We compare the performance of the regular batch algorithm MORe++ (cf. Section 2) with its adapted anytime version AnyCORE as introduced in Section 3. For our experiments we initialized the input parameters of the compared algorithms with the empirically determined values described in Table 2. The *outlier_scoring* method is implemented by the relative outlier dimension count ratio and the outlier scoring threshold *ost*, which decides if a sample is considered as an actual outlier in the full-dimensional space. Moreover, for the arbitrarily selectable clustering method *cluster_algo* of AnyCORE we decided to conduct the experiments with *k*-Means++ to compute clusters for the outlier detection phase but also to emphasize the beneficial effects generated by the anytime related adaptions like iterative outlier removal and anytime response. Furthermore, we used the Area Under the Receiver Operating Characteristic Curve (ROC AUC)[4] score as evaluation metric for our experiments. Based on rates, ROC inherently adjusts for the imbalance of class sizes typical of outlier detection tasks.[23]
All experiments ran 100 times for each MORe++ due to *k*-Means++'s non-determinism. So it is possible that for some timestamps only a few results were measured, depending on how much time AnyCORE has needed to finish its computation. We took the average of these results for MORe++ and compared them to the average result of AnyCORE with respect to the respective timestamp. Whenever the intermediate result might change, we saved the current timestamp and the current *outlierCountDim* and calculated the outlier score for each timestamp after the computation is finished. Due to saving timestamps and scores for anytime response, AnyCORE needs partly more time than MORe++ (cf. Figure 3a, 3c and 3g).

[4]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html (18.07.2021)

	outlier_scoring	*cluster_algo*
	ost	*k*-Means
Glass	0.1	5
Lympho	0.4	2
Mnist	0.2	2
Musk	0.3	3
Optdigits	0.2	9
Shuttle	0.3	1
Smtp	0.4	1
SynthA	0.2	5
SynthB	0.2	5

Table 2: Empirically determined hyperparameters.

Result discussion. In this section we discuss the results of our experiments. We compared the runtime and the quality of the outlier detection over time with the batch algorithm MORe++ and our adapted anytime version AnyCORE. As further described in the experimental setup the outlier detection quality is represented by the ROC AUC evaluation metric, which gets illustrated in Figure 3 and Figure 4. Results better than randomness are indicated by ROC AUC Scores above 0.5.
The experiments showed that the result quality strongly depends on the data set. Observing the result plots in Figure 3 and Figure 4 one recognizes that AnyCORE gets better the more time it takes for computation. For the shuttle dataset in Figure 3d, our approach is even significantly faster than MORe++. Lympho in Figure 3b, MNIST in Figure 3g and Musk in Figure 3a have adequate results. The intermediate results are similar to the outcome of MORe++.
The plots of Glass in Figure 3f and Optdigits in Figure 3e are especially interesting, since in those AnyCORE has partly much better intermediate results than MORe++. But these two data sets also support the assumption from section 3 of potentially facing performance fluctuations based on the alternating outlier removals. Thus more experiments and evaluations about how the outliers behave and change during the iterations of AnyCORE are required. The experiments on the synthetic data sets in Figure 3h, 4a and 4b just show that saving *timestamp* and *outlierCountDim* takes especially much time for many dimensions. AnyCORE is comparably slow due to saving *timestamp* and *outlierScore*, but it provides intermediate results. Thus, for massive complex data sets AnyCORE is a reasonable choice, if users are looking for an anytime algorithm. Its comparison to MORe++ is more difficult, as MORe++ is often finished before even the first intermediate results got provided, but AnyCORE is likely to deliver more refined results with increasing runtime. Thus, before making a final decision, there should be evaluations conducted for each individual data set in order to investigate how and when AnyCORE provides better results.

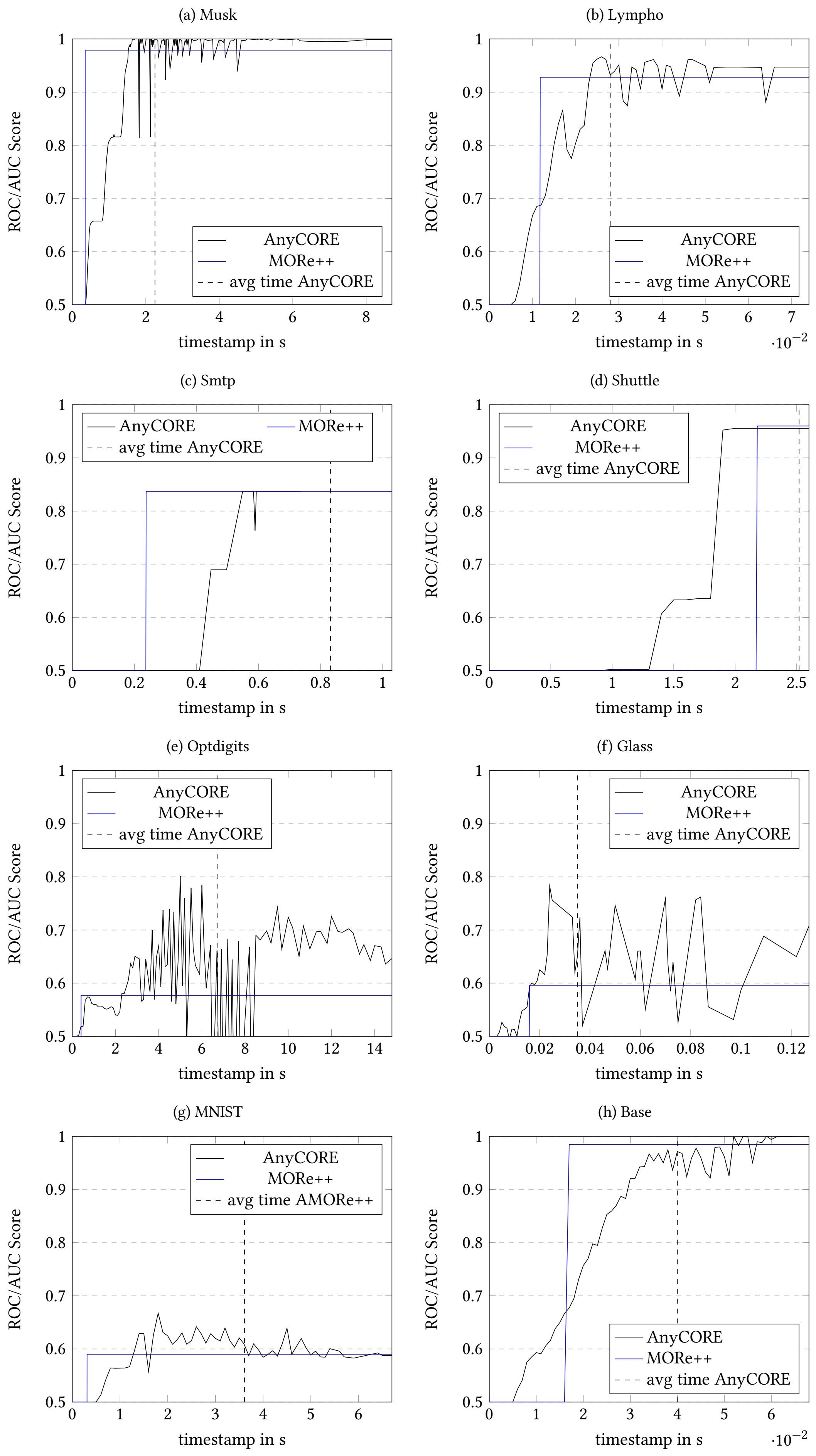

Figure 3: AnyCORE experiment results on base and real-world datasets.

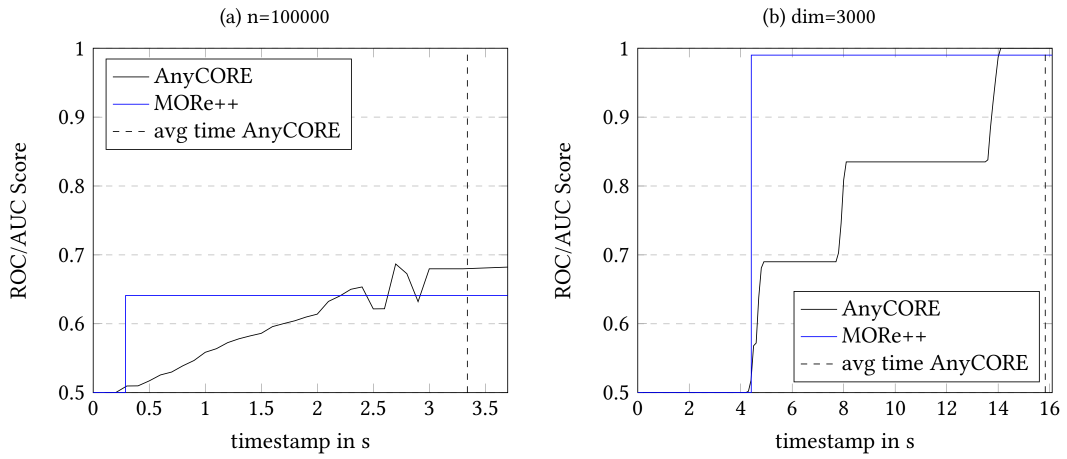

Figure 4: AnyCORE experiment results on synthetic datasets.

5. Conclusion

In summary, we introduced the Anytime Cluster Outlier REmoval algorithm AnyCORE, which utilizes MORe++'s outlier detection approach relying on histograms of 1d-projections of separately regarded clusters. Profiting from MORe++'s advantages of not depending on distance measures to work in high-dimensional spaces and of non-exponential runtimes, AnyCORE enhances this outlier removal algorithm by allowing users to interrupt and check intermediate results at any time and to stop in case of reviewed results are considered satisfactory. The experiments demonstrated, that dependent on the individual data set, our anytime version tends to outperform its batch algorithm version MORe++. Since the AnyCORE convergence criteria of the percentage of cluster jumpers[2] is independent of the underlying clustering algorithm, further evaluations with alternative clustering methods than *k*-Means++ can be done in future work. Furthermore, the partly fluctuating intermediate results could be addressed by a moving average of the initialization values for the clustering algorithm. In addition to that, the current implementation of the histogram-based *outlier_scoring* method could be replaced and evaluated with alternative approaches like Kernel Density Estimation, which does not depend on the number of bins. Moreover, the runtime of MORe++ as well as of the proposed method AnyCORE would benefit furthermore from parallelization of the outlier detection and removal steps done for each single dimension.

Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibility for its content.

References

[1] A. Beer, J. Lauterbach, T. Seidl, More++: k-means based outlier removal on high-dimensional data, in: G. Amato, C. Gennaro, V. Oria, M. Radovanović (Eds.), Similarity Search and Applications, Springer International Publishing, Cham, 2019, pp. 188–202.

[2] C. Aggarwal, Outlier analysis, in: Springer New York, 2013.

[3] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: SODA '07, 2007.

[4] S. Chawla, A. Gionis, k-means–: A unified approach to clustering and outlier detection, in: 13th SIAM International Conference on Data Mining, Austin, Texas, 2013, 2013, pp. 189–197. VK: hiit.

[5] M. Boddy, T. L. Dean, Deliberation scheduling for problem solving in time-constrained environments, Artificial Intelligence 67 (1994) 245–285.

[6] T. Dean, M. Boddy, An analysis of time-dependent planning, in: Proceedings of the Seventh AAAI National Conference on Artificial Intelligence, AAAI'88, AAAI Press, 1988, p. 49–54.

[7] S. Zilberstein, Using anytime algorithms in intelligent systems, AI Mag. 17 (1996) 73–83.

[8] S. Zilberstein, S. J. Russell, Approximate reasoning using anytime algorithms, 1995.

[9] J. Grass, S. Zilberstein, Anytime algorithm development tools, SIGART Bull. 7 (1996) 20–27.

[10] V. R. Lesser, J. Pavlin, E. Durfee, Approximate processing in real-time problem solving, AI magazine 9 (1988) 49–49.

[11] S. T. Mai, I. Assent, M. Storgaard, Anydbc: An efficient anytime density-based clustering algorithm for very large complex datasets, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, 2016, p. 1025–1034.

[12] S. T. Mai, I. Assent, J. Jacobsen, M. S. Dieu, Anytime parallel density-based clustering 32 (2018) 1121–1176.

[13] O. Arslan, D. E. Koditschek, Anytime hierarchical clustering, 2014. arXiv:1404.3439.

[14] C. Böhm, J. Feng, X. He, S. Mai, Efficient anytime density-based clustering, in: SDM, 2013.

[15] S. Mai, X. He, J. Feng, C. Plant, C. Böhm, Anytime density-based clustering of complex data, Knowledge and Information Systems 45 (2014) 319–355.

[16] J. Lin, M. Vlachos, E. J. Keogh, D. Gunopulos, Iterative incremental clustering of time series, in: EDBT, 2004.

[17] J. Lin, M. Vlachos, E. J. Keogh, D. Gunopulos, J. Liu, S. Yu, J. jin Le, A mpaa-based iterative clustering algorithm augmented by nearest neighbors search for time-series data streams, in: PAKDD, 2005.

[18] A. Tsitsulin, M. Munkhoeva, D. Mottin, P. Karras, I. Oseledets, E. Müller, Frede: Anytime graph embeddings, Proc. VLDB Endow. 14 (2021) 1102–1110.

[19] I. Assent, P. Kranen, C. Baldauf, T. Seidl, Anyout: Anytime outlier detection on streaming data, in: DASFAA, 2012.

[20] P. Kranen, I. Assent, C. Baldauf, T. Seidl, The clustree: indexing micro-clusters for anytime stream mining, Knowledge and Information Systems 29 (2010) 249–272.

[21] A. Giacometti, A. Soulet, Anytime algorithm for frequent pattern outlier detection, Inter-

national Journal of Data Science and Analytics 2 (2016) 119–130.

[22] B. Sofman, J. Andrew Bagnell, A. Stentz, Anytime online novelty detection for vehicle safeguarding, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 1247–1254. doi:10.1109/ROBOT.2010.5509357.

[23] A. Zimek, On the evaluation of unsupervised outlier detection, 2016. URL: https://imada.sdu.dk/~zimek/InvitedTalks/TUVienna-2016-05-18-outlier-evaluation.pdf, talk at TU Vienna (last access: 18.07.2021).