MORe++: k-Means Based Outlier Removal on High-Dimensional Data

Anna Beer(✉), Jennifer Lauterbach, and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany
{beer,seidl}@dbs.ifi.lmu.de, j.lauterbach@campus.lmu.de

Abstract. MORe++ is a k-**M**eans based **O**utlier **Re**moval method working on high dimensional data. It is simple, efficient and scalable. The core idea is to find local outliers by examining the points of different k-Means clusters separately. Like that, one-dimensional projections of the data become meaningful and allow to find one-dimensional outliers easily, which else would be hidden by points of other clusters. MORe++ does not need any additional input parameters than the number of clusters k used for k-Means, and delivers an intuitively accessible degree of outlierness. In extensive experiments it performed well compared to k-Means-- and ORC.

Keywords: Outlier detection · High-dimensional · Histogram-based · K-means

1 Introduction

As outlier detection in general delivers valuable results for fraud detection, medical problems, or finding errors in data, most techniques do not regard the plethora of attributes which is gathered for each data point in modern applications. An outlier, which is often defined as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [14], is more difficult to find in high-dimensional data than in low-dimensional, since the mechanisms generating data are difficult to identify in high-dimensional data due to the curse of dimensionality. Thus, most classic outlier detection algorithms are not applicable to high-dimensional data. Density based algorithms for example, are not meaningful in high-dimensional data, which usually is per se sparse. Also angular based outlier factors like, e.g., ABOD [18], are not interpretable anymore for high-dimensional data. Moreover, most of those algorithms do not scale with the number of dimensions.

Thus we introduce MORe++ (k-*M*eans-based *O*utlier *Re*moval using k-Means++), a fast method to score outliers in high-dimensional data. We achieve scalability w.r.t. the number of dimensions and retain explainability of the scores by regarding each dimension separately. In contrast to other methods we can even find clusters or outliers overlapping in some dimensions, since we do not

G. Amato et al. (Eds.): SISAP 2019, LNCS 11807, pp. 188–202, 2019.
https://doi.org/10.1007/978-3-030-32047-8_17

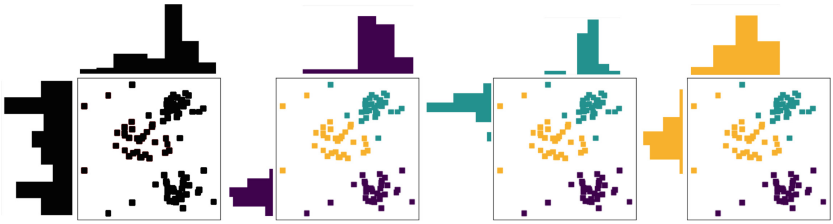

Fig. 1. Histogram of complete dataset in black, in contrast to histograms of points belonging to the same centroid according to k-means in green, yellow and purple. (Color figure online)

regard all points at once, but only those which are in one cluster according to k-Means. Using histograms, we accelerate our algorithm even further in regards to the number of points. Figure 1 shows how overlapping in a dimension prevents finding outliers if regarding the whole dataset at once, which is why we regard only a part of the datapoints at once.

Summarizing, our main contributions are:

1. We introduce a meaningful score for outliers in high-dimensional data
2. Our method is fast and scales linearly with the number of dimensions and points
3. It is based on k-Means and compatible to a lot of k-Means extensions
4. It is easy to implement
5. It is easily parallelizable and suitable for high-dimensional data, since it does not rely on distance measures operating on the full-dimensional space.

The remainder is structured as follows: in Sect. 2 we give an overview over other k-Means extensions and outlier detection methods using k-Means or a histogram-based approach. We also investigate diverse approaches of histogram segmentation. The complete algorithm is explained in detail in Sect. 3. In Sect. 4 we examine our algorithm regarding a plethora of aspects in overall 40 synthetic as well as real data experiments. Section 5 concludes this paper giving a short summary and prospect to promising future work.

2 Related Work

We first give the foundations looking at k-Means Clustering and existing recent extensions in Sect. 2.1. As there are already several methods combining k-Means and outlier removal, we give an overview over those in Sect. 2.2 and discuss the advantages of MORe++ in contrast to them. In Sect. 2.3 we look at histograms and outlier detection algorithms using them. We note that regarding only the projections of the complete dataset at once cannot lead to an effective outlier detection.

2.1 K-Means Clustering and Extensions

k-Means [19,20] is one of the most famous clustering algorithms and is still frequently used for diverse tasks. Given the number of clusters k, centers are randomly initialized in the original algorithm. All points are assigned to their closest center and the cluster centers are recomputed. Those two steps are repeated until no point changes its cluster membership any more and the algorithm converges against a local minimum of the mean distance from points to their cluster centers.

There exist several improvements of k-Means. For example, k-Means++ [2] optimizes the initial cluster centers by regarding the shortest distance to already chosen cluster centers. We will use this extension for our algorithm MORe++, as it usually improves the quality of clustering and reduces the variance of results. Another improvement is k-Median [5], which uses the median instead of the mean when calculating the new cluster centers to minimize the negative impact of outliers. Nevertheless, this comes at the cost of an increased runtime. On the other hand, kmeans|| [3] reduces runtime by parallelizing k-Means++. Instead of sampling single points for the initialization like k-Means++, $O(k)$ points are sampled $O(\log n)$ times. Also high-dimensional data can be clustered better with diverse variants of k-Means developed for subspace clustering, like NR-kmeans [22] or Sub-kmeans [21]. Where Sub-kmeans finds a "clustered" space containing all structural information and a noise space, NR-kMeans looks for an optimal arbitrarily oriented subspace for each partition. Those improvements, of which we apply only k-Means++ in this paper, are compatible to MORe++ and will be regarded in future work.

2.2 Outlier Detection and K-Means

There are several algorithms combining k-Means and outlier removal, of which we introduce the most common ones in the following. In Sect. 4 we will compare MORe++ to the first both introduced, k-means-- [7] and ORC [13].

k-means-- [7] combines outlier removal and k-Means by alternately removing outliers and performing k-Means iterations. In every step l points which are farthest away from their center are removed from the dataset for the next calculation of cluster centers, where l is given by the user. Even for $k = 1$, its running time is $O(n^{d^3})$, which is infeasible for high-dimensional data.

ORC (Outlier Removal Clustering) [13] assigns an outlyingness factor o_i to every point after a complete pass of k-Means. Points with o_i higher than a user given threshold T are removed from the set of points, and k-Means is performed again. o_i is, similar to k-means--, based on the distance to the nearest cluster center, and normalized by division by the highest distance between a point and its center. The algorithm is quite sensitive to the choice of T, and our experiments will show that ORC cannot handle high dimensional data well.

NEO-K-Means [28] considers outliers and overlapping clusters, for which it requires two parameters α and β. Using those parameters, it strives for a "trade-off between a clustering quality measure, overlap among the clusters, and

non-exhaustiveness (the number of outliers not assigned to any group)" [28] in every k-Means step. Reaching this trade-off requires iterations until convergence, there seems to be neither an upper bound for the running time, nor do the authors perform a complexity analysis. The main criterion is again the distance between points and their closest center, which becomes more and more useless with increasing number of dimensions.

KMOR [10] uses an additional cluster for all outliers, needing two parameters to control the number of outliers. A point is considered an outlier if the distance to *all* cluster centers is at least $\gamma \times d_{avg}$, which forbids finding local outliers.

Other methods combining k-Means and outlier detection are, e.g., ODC (Outlier Detection and Clustering) [1], where outliers are points having a distance to their cluster centers larger than p times the average distance. CBOD [17] and [16] are two-stage algorithms, where [16] additionally creates a minimum spanning tree on which they work on. [29] is a three stage algorithm first finding local outliers, then global outliers, and lastly combining clusters with similar densities and overlapping clusters.

All mentioned algorithms have in common that they rely on distance measures in the full-dimensional space, usually the Euclidean distance. As with increasing number of dimensions, all distances become similar due to the curse of dimensionality [4], the results get distorted for high-dimensional data. In contrast, MORe++ does not need any distance measure working on high-dimensional space. Additionally, due to the separate consideration of each dimension, it is already faster than these methods plus it is easy parallelizable.

2.3 Outlier Detection with Histograms

As using histograms is an established way to simplify data, several construction possibilities regarding the bin-width and bin-quantity exist: One of the most common possibilities, Sturges' rule [15], tends to oversmooth histograms and does not work well with large datasets and not normally distributed data. Other common rules are Scott's rule [26] and Freedman and Diaconis's rule [9], which are both better for larger samples. MORe++ uses Scott's rule which suggests h as the number of bins: $h = \frac{3.45\hat{\sigma}}{n^{1/3}}$, where $\hat{\sigma}$ is the sample standard deviation.

There are methods using histograms to find outliers: HBOS [12] constructs a histogram for each dimension and calculates an anomaly score for each data instance using the inverse estimated densities and supposing feature independence. [11] finds sparse regions in the dataset using histograms and a nearest neighbour approach. Based on those regions, local outlier candidates are identified, which can be removed from the set of outliers in a later optional reconsideration phase.

Looking at higher dimensional datasets and subspace clustering, objects may belong to different clusters in different subspaces, thus they could be outliers in some subspaces, but not in others. OutRank [27] addresses this issue by introducing a "degree of outlierness", the outlier rank. With that, points which are only in a subset of attributes anomalies, can also be detected as outliers [23]. In contrast, many outlier detection algorithms, like for example, LOCI (local

correlation integral) [24], look for outliers in the full dimensional space, or even micro-clusters. LOF [6], the local outlier factor, is another approach returning the degree of outlierness. It regards the isolation of a point with respect to the surrounding neighborhood, and was even extended for high-dimensional data [18,30].

3 MORe++

In the following we describe and analyze the outlier detection algorithm MORe++ in detail: Sect. 3.1 gives an overview, Sect. 3.2 explains how we find one-dimensional outliers in histograms, and Sect. 3.3 gives a complexity analysis.

3.1 Outline of MORe++

Based on the idea already shown in Fig. 1, MORe++ regards the points of every k-Means cluster separately. Like that, one-dimensional projections already enable the detection of outliers. Section 3.2 explains how to get one-dimensional outliers based on the according histogram. The higher the number of dimensions in which a point is considered an outlier, the higher is its outlier score. Thus, MORe++ finds a degree of outlierness. Algorithm 1 describes our approach in detail: on the basis of the clustering returned by k-Means++, we build a histogram for every dimension for every cluster. The method *calculate1dOutliers* returns one-dimensional outliers for each dimension and each cluster given the according histogram, as explained in Sect. 3.2. The outlier score is the relation between the number of dimensions in which the point is considered a (one-dimensional) outlier and the total number of dimensions. Users can now either use this degree of outlierness, or give a threshold *ost* (outlier score threshold), so that points with an outlier score higher than *ost* are outliers in the full dimensional space. This approach delivers several advantages:

1. Our distance measure does not get skewed with increasing number of dimensions, as we regard every dimension separately
2. We can easily parallelize the calculation of outliers as we regard all clusters and also all dimensions independently. Thus, MORe++ is suitable for many points as well as for high- dimensional data.
3. Users do not have to know the number of outliers beforehand
4. A degree of outlierness gives more information than a hard classification
5. The threshold users *can* give is quite intuitive, as it is simply the minimal percentage of dimensions in which a point should be a one-dimensional outlier. As experiments will show, MORe++ is quite robust w.r.t. *ost* (we use the same value $ost = 0.2$ for 35 out of 40 experiments in total), thus a hard classification using a fixed *ost* is also a promising idea for future work
6. MORe++ is very fast with only $O(nd)$, where, e.g., k-means-- is exponential.

Algorithm 1. Pseudo-Code of MORe++

```
Data: Data X, number of clusters k
Result: Clustering labels, outlierScore for data points
foreach x ∈ X do
    numberOfOutlierDims[x] ← 0 ;
end
foreach c ∈ clusters do
    foreach d ∈ range(dimensions) do
        Build histogram;
        1dOutliers ← calculate1dOutliers(histogram);
        foreach 1dOutlier ∈ 1dOutliers do
            numberOfOutlierDims[1dOutlier] + +;
        end
    end
end
foreach x ∈ X do
    outlierScore ← numberOfOutlierDims[x]/dimensions ;
end
```

3.2 Detecting One-Dimensional Outliers in Histograms

To detect one-dimensional outliers in histograms, it is important that we only look at points assigned to one cluster by k-Means. Else, points of other clusters would cover outliers in the one-dimensional projections, as Fig. 1 already showed: see for example the outlier in the bottom middle, which is later in the first, purple cluster. Using histograms of the complete data, it is covered in both dimensions by the purple resp. the yellow cluster. Looking at the histogram of only the points assigned to the first (purple) cluster for dimension 1 (horizontal), it can be detected quite easily using the following approach:

If there are empty bins in the histogram, as shown in Fig. 2 on the left, then we partition the data along these empty bins. If there are no empty bins, we split the data where the height of the bins changes most from one bin to the next, relatively to the higher bin of both, as can be seen in Fig. 2 on the right. If there are several changes $s_0, ..., s_j$ which are (relatively) equally high, then we perform the split in the middle at $s_{\lfloor j/2 \rfloor}$. After the dataset is partitioned, all points which are not in the partition containing the majority of the points are marked as outliers for this dimension.

3.3 Complexity Analysis

For n points of dimensionality d the complexity of k-Means itself is $O(nkdi)$ with i the number of iterations until convergence and k the number of clusters. We build a histogram for every cluster and every dimension, which sums up to $O(kdb)$ for histograms with $b < n$ bins. Using Scott's rule for the construction of histograms (see Sect. 2.3), $b \in O(n^{-\frac{1}{3}})$. One-dimensional outliers are calculated

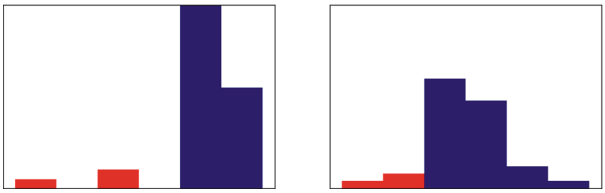

Fig. 2. Outliers (marked red) are detected using either empty bins or the highest relative difference between two adjacent bins. (Color figure online)

in $O(b+n) \subseteq O(n)$ and the calculation of all outlier scores can be done in $O(nd)$. Thus, MORe++ lies with $O(nd+kdn^{-\frac{1}{3}}) \subseteq O(nd)$ in a smaller complexity class than k-Means itself and is only linear in the number of dimensions as well as in the number of points. Furthermore, it is easily parallelizable.

In contrast, the running time of k-means– is with $O(n^{d3})$ much larger. ORC, which delivers clearly worse results than MORe++, needs to run j iterations of k-Means alternately with determining the outlyingness factor, which is in $O(nd)$, resulting in a total runtime of $O(j * (nkdi + nd)) \subset O(j * nkdi)$.

4 Experiments

We performed several experiments regarding the quality of outlier detection compared to ORC and k-Means-- (see Sect. 2), based on the ROC AUC (Area Under the Receiver Operating Characteristic Curve) value [8][1] and F1-measure; due to the lack of space and a high similarity of the results we only show the former here. All synthetic datasets were constructed using cluster centers drawn from a uniform distribution function and generating Gaussian distributed clusters around them. Outliers were added following a uniform distribution function.

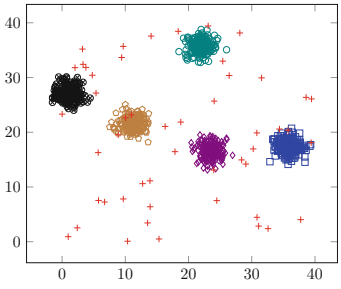

Fig. 3. 2d-projection of the base case experiment

In Sect. 4.1 we examine the influence of the following aspects onto the results of MORe++: size of dataset n, number of dimensions dim, percentage of outliers out, variance of clusters var, number of clusters k, and percentage of additional noise dimensions dim_n. For that, we created a base-case shown in Fig. 3 from which we kept all parameters but one at a time to investigate MORe++'s behaviour regarding that one aspect. In Sect. 4.2 we regard the behaviour of the algorithms on some real world datasets. They show, that even though MORe++ and k-Means-- deliver similar results

[1] Reminder: ROC AUC ranges from 0 to 1, where a perfect outlier prediction is 1. It regards the true positive rate vs. false positive rate. If ROC AUC is 0.5, the model has no class separation capacity.

in most of the previous test series, MORe++ clearly outperforms k-Means-- for real world data.

4.1 Influence of Various Aspects

To evaluate the behaviour of MORe++ regarding several aspects, we created a base case experiment as explained above with the following parameters: size of dataset $n = 1000$, number of dimensions $dim = 5$, percentage of outliers $out = 5\%$, variance of clusters $var = 1.2$, number of clusters $k = 5$, and percentage of additional noise dimensions $dim_n = 0$. Figure 3 shows two arbitrary dimensions of the base-case, where the ground truth outliers are marked by red crosses and clusters by colored shapes other than crosses. In each test series we changed exactly one parameter of that base case and compared the results to those of ORC and k-means--, where the base case is always marked with box brackets in the x-axis. To improve comparability, we also kept the (initially randomly chosen) cluster centers of the generated clusters the same, where possible.

As k-means-- is non-deterministic, we took the average of 100 executions. MORe++ and ORC are deterministic due to the use of cluster centers as in k-Means++ [2]. For each test series and each algorithm, we chose the parameter resulting in the best ROC AUC value after testing values from 0 to 1 in steps of 0.1 for the parameter ost in MORe++ and T in ORC, which resulted for both parameters and most of the test series in a value of 0.2, otherwise the best parameter settings are given in the according experiments. For k-Means-- the parameter l was taken from the ground truth (i.e. $l = 50$ for all experiments but the ones were the percentage of outliers or the size of the dataset was changed).

Experiments Regarding Number of Points. To examine MORe++'s behaviour with respect to the size of the dataset, we tested the base case with different values for the number of points $n = 500, 1000, 2500, 5000, 10000, 100000$. The results can be seen in Fig. 4 and show that MORe is better or comparable to ORC and k-Means-- in most cases. For 10000 points and only 5 dimensions, relatively many outliers are overlapping with the cluster itself, which explains the slight decline of results for a very high ratio between dimensions and points.

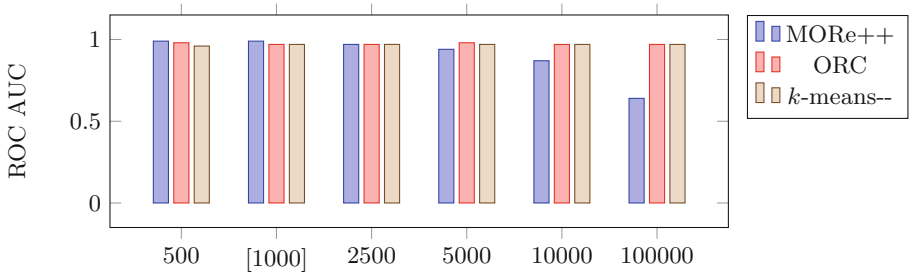

Fig. 4. ROC AUC Score for increasing number of points n

As MORe++ was developed for high-dimensional data, and especially for a high dimension to data ratio, this slight decline is predictable as well as manageable.

Experiments Regarding Number of Dimensions. We increased the number of dimensions up to 3000, which is a ratio of dimensions to data points of 3. For a lot of use cases, like data mining of textual data or image processing, the number of dimensions usually exceeds the number of data points. That often constitutes a problem for outlier detection algorithms as ORC: for a higher dimensionality than 30, the results of ORC[2], like it can be seen in Fig. 5 worsen a lot and from 1000 dimensions on, a constant ROC AUC of 0.6 is reached, which is only slightly better than guessing (which would be a ROC AUC of 0.5). k-Means-- on the other hand performs effectively as good as MORe++, and for both of them there is no decrease of quality subject to the dimensionality; they both perfom almost perfectly in high-dimensional space. For $dim \geq 5$, the ROC AUC for MORe++ is always at least 0.99, for k-Means-- the same holds for $dim \geq 30$. Note, that all clusters in this test series are in the full dimensional space, thus very similar results of MORe++ and k-Means-- were expectable.

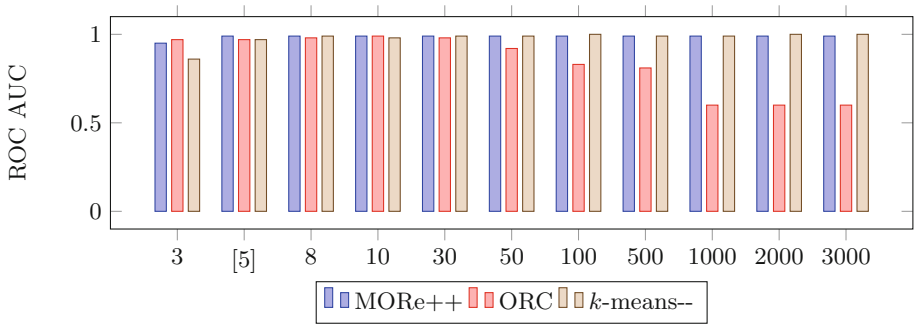

Fig. 5. ROC AUC Score for increasing number of dimensions dim

Experiments Regarding Percentage of Outliers. As Fig. 6 shows, MORe++ becomes less accurate for increasing number of outliers as well as ORC. A high percentage of outliers is difficult to handle well using our approach of finding one-dimensional outliers in histograms, as high amounts of outliers smoothen the one-dimensional histograms. But those high percentages of outliers constitute rather noise than some interesting, outlying points, which MORe++ aims to find, thus, this is more a question of where "outlierness" ends and "noise" starts.

[2] Best values for T: 0.8 for $dim = \{50, 100\}$, 0.9 for $dim = \{100, 500\}$, else $T = 0.2$.

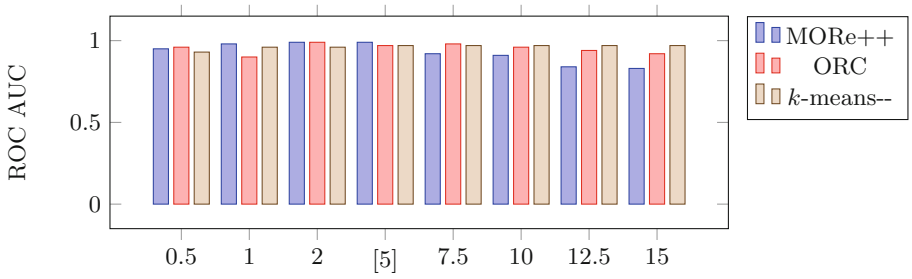

Fig. 6. ROC AUC Score for different outlier percentages

Experiments Regarding Variance of Clusters. For different variances $var = 0.7, 1.0, 1.2, 1.5, 2.0$ of the clusters MORe++ reached almost constant results, where ORC and k-Means-- get noteably worse with increasing variance, as Fig. 7 shows. That is, because with increasing variance the (full-dimensional) distances from points to their centers increase, too, thus they are more similar to the distances from outliers to cluster centers. As MORe++ does not rely on distinguishing high-dimensional distances of non-outliers and outliers to cluster centers, it is able to cope very well with diverse variances, in contrast to the comparative methods.

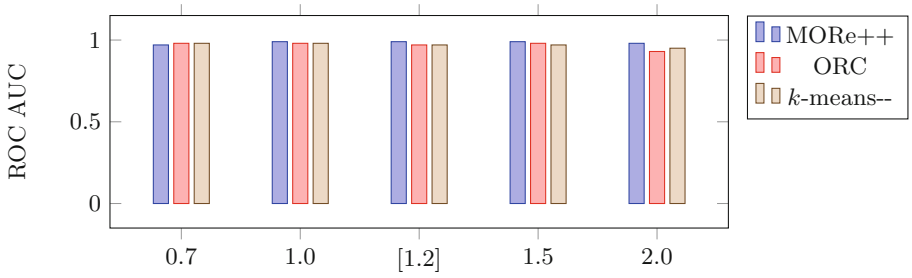

Fig. 7. ROC AUC Score for different cluster variances *var*

Experiments Regarding Number of Clusters. With increasing number of clusters, there are more overlapping clusters, thus k-Means and also all of the tested outlier detection algorithms[3] become worse, as can be seen in Fig. 8. But in contrast to k-means--, MORe++ gains an advantage, as the dataset is divided into more subsets (clusters found by k-Means) on which the outlier detection is performed separately. Thus, the outliers become more obvious in those smaller subsets, which counteracts the before mentioned negative effects. That results in a relative improvement to k-means--, although the quality of outlier detection

[3] Best values for T: 0.4 for $k = \{8, 10\}$, 0.6 for $k = \{25, 50\}$, else $T = 0.2$.

decreases noteably even for MORe++ for datasets with a higher number of clusters than $k \geq 25$.

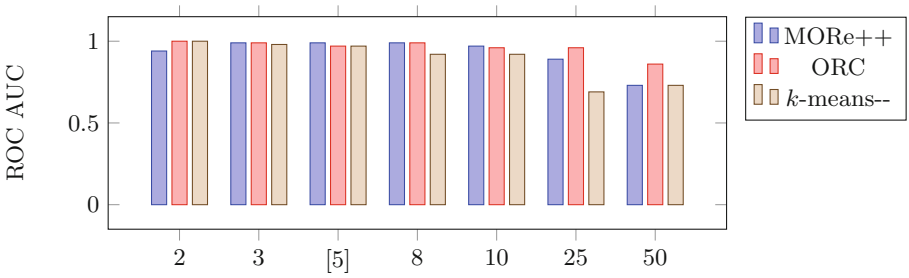

Fig. 8. ROC AUC Score for different numbers of clusters k

Experiments Regarding Noise Dimensions. Subspace clustering is based on the assumption, that with increasing number of dimensions more and more dimensions become "noise dimensions". According to that, we added noisy dimensions to our dataset, for which the results can be seen in Fig. 9[4]. With an increase of noise dimensions, distance measures in the full-dimensional space become more and more meaningless according to the curse of dimensionality, and the noise of some dimensions distorts the outlierness in other dimensions for algorithms using distances on the full-dimensional space. As MORe++ regards every dimension separately and counts the number of dimensions in which a point is an outlier, it is clearly more robust to additional noise dimension than its competitors.

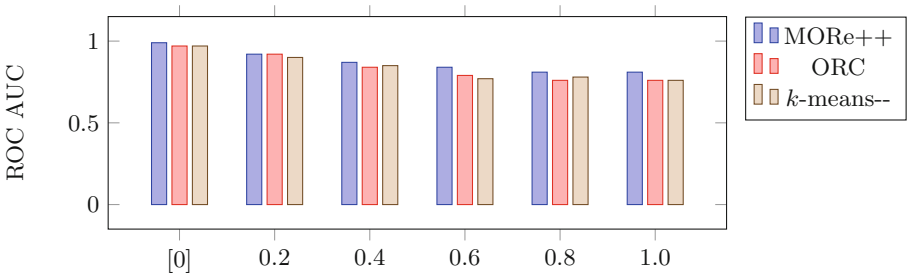

Fig. 9. ROC AUC Score for different number of noise dimensions

[4] Best ROC AUC values for ORC were achieved with $T = 0.5$ for $dim_n = 0.2$, $T = 0.7$ for $dim_n = 1.0$, and $T = 0.6$ else. For MORe++ $ost = 0.3$ delivered best results for $dim_n = \{0.8, 1.0\}$, else $ost = 0.2$.

Evaluation of Systematic Experiments. The biggest difference of results could be seen for high-dimensional data, where ORC was clearly outperformed by MORe++ and k-Means--. As many results seem to be similar or only slightly better than k-Means--, we want to emphasize the difference in runtime, where MORe++ only needs (including running the k-Means) $O(nkdi)$ and k-Means-- needs $O(n^{d^3})$. Further, MORe++ persuaded especially in the following points:

- For datasets of high dimensionality
- For datasets with clusters of higher variances
- For datasets with noise dimensions

4.2 Real World Datasets

We tested some real world datsets with different properties: sizes ranged between 148 and almost 95000, dimensionalities between 3 and 166, percentage of outliers between 0.4 and 7 and number of clusters between 1 and 5. Table 1 gives an overview over size, number (and percentage) of outliers, and number of clusters of the real world datasets which we used and which can be found in the ODDS library [25]. Table 2 gives the parameters chosen for all algorithms, where we used the parameter resulting in the best ROC AUC, resp. the ground truth value as number of outliers l for k-means--. For MORe++ and ORC we tested values between 0 and 1 in steps of 0.1. As in the previous section, we took the average of 100 executions of k-means-- due to its non-determinism. Figure 10 shows the results of the experiments: for Lympho, Shuttle, and Smtp MORe++ clearly outperforms both other algorithms. Glass is an interesting experiment, as ORC is the best performing algorithm here, followed by MORe++. For Musk, MORe++ achieved the best results, closely followed by k-means-- and ORC. So, even though MORe++ performed quite similar to k-means-- in most cases in the previous section, it seems to be more suitable for real world scenarios plus it is by far faster. ORC performed well on the Glass dataset, but cannot deal with very high number of dimensions as shown in Sect. 4.1. Thus, for outlier detection in high-dimensional datasets, MORe++ is preferable.

Table 1. Overview of real world datasets

Dataset	n	dim	outlier	k
Glass	214	9	9 (4.2%)	5
Lympho	148	18	6 (4.1%)	2
Musk	3062	166	97 (3.2%)	3
Shuttle	49097	9	3511 (7%)	1
Smtp	95156	3	2211 (0.4%)	1

Table 2. Chosen parameters for real world datasets

	MORe++ *ost*	ORC *T*	k-means– l
Glass	0.1	0.3	9
Lympho	0.4	0.7	6
Musk	0.3	0.7	97
Shuttle	0.3	0.3	3511
Smtp	0.4	0.5	2211

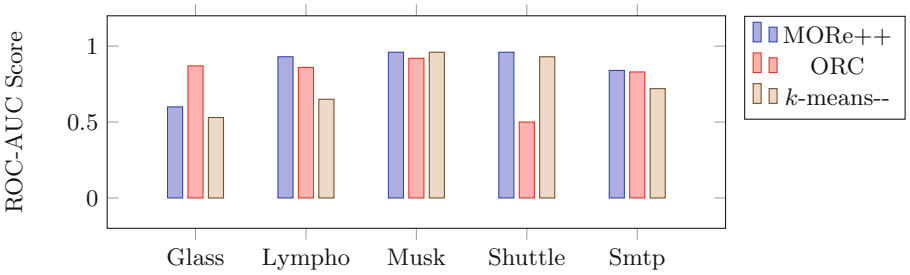

Fig. 10. ROC AUC for real world datasets

5 Conclusion

In conclusion we developed the outlier detection algorithm MORe++, which is based on histograms of one-dimensional projections of separately regarded k-Means clusters. By projecting onto single dimensions, we can circumvent some aspects of the curse of dimensionality: neither do we need a distance measure working in high-dimensional space nor is our runtime exponential in the number of dimensions. Users do not have to know the number of outliers beforehand and local outliers can easily be detected. The algorithm is easily parallelizable and easy to implement. A plethora of variations and improvements of k-Means could be used to further improve our already good results, and also using another algorithm than k-Means as foundation for the partitioning of the data could be tried.

Acknowledgement. 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 responsibilities for its content.

References

1. Ahmed, M., Mahmood, A.N.: A novel approach for outlier detection and clustering improvement. In: 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pp. 577–582. IEEE (2013)
2. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
3. Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S.: Scalable k-means++. Proc. VLDB Endow. **5**(7), 622–633 (2012)
4. Bellman, R.E.: Adaptive Control Processes: A Guided Tour, vol. 2045. Princeton University Press, Princeton (2015)
5. Bradley, P.S., Mangasarian, O.L., Street, W.N.: Clustering via concave minimization. In: Advances in Neural Information Processing Systems, pp. 368–374 (1997)
6. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM Sigmod Record, vol. 29, pp. 93–104. ACM (2000)

7. Chawla, S., Gionis, A.: k-means-: a unified approach to clustering and outlier detection. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 189–197. SIAM (2013)
8. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006)
9. Freedman, D., Diaconis, P.: On the histogram as a density estimator: L 2 theory. Probab. Theory Relat. Fields **57**(4), 453–476 (1981)
10. Gan, G., Ng, M.K.P.: K-means clustering with outlier removal. Pattern Recognit. Lett. **90**, 8–14 (2017)
11. Gebski, M., Wong, R.K.: An efficient histogram method for outlier detection. In: Kotagiri, R., Krishna, P.R., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 176–187. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71703-4_17
12. Goldstein, M., Dengel, A.: Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. In: KI-2012: Poster and Demo Track, pp. 59–63 (2012)
13. Hautamäki, V., Cherednichenko, S., Kärkkäinen, I., Kinnunen, T., Fränti, P.: Improving K-means by outlier removal. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 978–987. Springer, Heidelberg (2005). https://doi.org/10.1007/11499145_99
14. Hawkins, D.M.: Identification of Outliers, vol. 11. Springer, Dordrecht (1980). https://doi.org/10.1007/978-94-015-3994-4
15. Hyndman, R.J.: The problem with Sturges rule for constructing histograms (1995)
16. Jiang, M.F., Tseng, S.S., Su, C.M.: Two-phase clustering process for outliers detection. Pattern Recognit. Lett. **22**(6–7), 691–700 (2001)
17. Jiang, S., An, Q.: Clustering-based outlier detection method. In: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 2, pp. 429–433. IEEE (2008)
18. Kriegel, H.P., Zimek, A., et al.: Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 444–452. ACM (2008)
19. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
20. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
21. Mautz, D., Ye, W., Plant, C., Böhm, C.: Towards an optimal subspace for k-means. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 365–373. ACM (2017)
22. Mautz, D., Ye, W., Plant, C., Böhm, C.: Discovering non-redundant k-means clusterings in optimal subspaces. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1973–1982. ACM (2018)
23. Müller, E., Assent, I., Iglesias, P., Mülle, Y., Böhm, K.: Outlier ranking via subspace analysis in multiple views of the data. In: 2012 IEEE 12th International Conference on Data Mining, pp. 529–538. IEEE (2012)
24. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: Loci: fast outlier detection using the local correlation integral. In: Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405), pp. 315–326. IEEE (2003)
25. Rayana, S.: ODDS library (2016). http://odds.cs.stonybrook.edu

26. Scott, D.W.: On optimal and data-based histograms. Biometrika **66**(3), 605–610 (1979)
27. Seidl, T., Müller, E., Assent, I., Steinhausen, U.: Outlier detection and ranking based on subspace clustering. In: Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2009)
28. Whang, J.J., Dhillon, I.S., Gleich, D.F.: Non-exhaustive, overlapping k-means. In: Proceedings of the 2015 SIAM International Conference on Data Mining, pp. 936–944. SIAM (2015)
29. Zhou, Y., Yu, H., Cai, X.: A novel k-means algorithm for clustering and outlier detection. In: 2009 Second International Conference on Future Information Technology and Management Engineering, pp. 476–480. IEEE (2009)
30. Zimek, A., Schubert, E., Kriegel, H.P.: A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Min.: ASA Data Sci. J. **5**(5), 363–387 (2012)