Short Paper

Rock – Let the points roam to their clusters themselves

Anna Beer
LMU Munich
Munich, Germany
beer@dbs.ifi.lmu.de

Daniyal Kazempour
LMU Munich
Munich, Germany
kazempour@dbs.ifi.lmu.de

Thomas Seidl
LMU Munich
Munich, Germany
seidl@dbs.ifi.lmu.de

ABSTRACT

In this work we present Rock, a method where the points **Ro**am to their **c**lusters using **k**-NN. Rock is a draft for an algorithm which is capable of detecting non-convex clusters of arbitrary dimension while delivering representatives for each cluster similar to, e.g., Mean Shift or k-Means. Applying Rock, points roam to the mean of their k-NN while k increments in every step. Like that, rather outlying points and noise move to their nearest cluster while the clusters themselves contract first to their skeletons and further to a representative point each. Our empirical results on synthetic and real data demonstrate that Rock is able to detect clusters on datasets where either mode seeking or density-based approaches do not succeed.

1 INTRODUCTION

Mode seeking clustering algorithms aim for detecting the modes of a probability density function (pdf) from a finite set of sample points. This type of clustering algorithm serves the purpose of detecting local maxima of density of a given distribution in the data set. In contrast density-based methods have the capability of detecting density-connected clusters of arbitrary shape. Rock positions itself between mode-seeking methods and density-based approaches. It is capable of detecting clusters of non-convex shapes *and* arbitrary densities at the same time. Plus it provides representatives for the detected clusters. Such an approach contributes additional expressiveness to the density-connected clusters. Further Rock provides the possibility to observe the directions of the data points roaming to the representatives. As an example, Figure 1 shows the path every point travels to its cluster representatives over time when Rock is applied. As Rock is not relying on any predefined density kernel it is able to detect non-convex clusters, which is a major advantage regarding other clustering methods.

The remainder of this paper is organized as follows: We provide a brief overview on the mode seeking method Mean Shift and other k-NN-based mode seeking algorithms. We then proceed explaining the Rock algorithm. Following to the elaboration on the algorithm we evaluate the performance of Rock against k-Means, Mean Shift and DBSCAN. Here the linking-role of Rock becomes clear, as our algorithm detects non-convex shaped areas where e.g., k-Means and Mean Shift fail and yet provides a representative of each detected cluster similar to the centroids in k-Means or modes in Mean Shift. We conclude this paper by providing an overview of the core features of our method and further elaborating on potential future work.

The main contributions of Rock are as follows:

- It finds non-convex shaped clusters of any density, which neither density based clustering methods nor mode or centroid based clustering methods are capable of
- The number of clusters does not have to be given by the user
- It provides a representative for every cluster

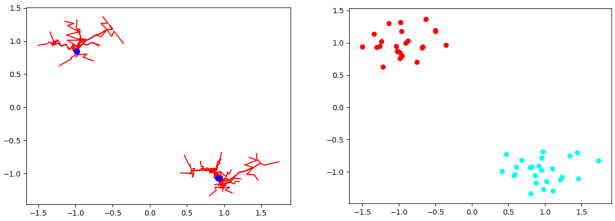


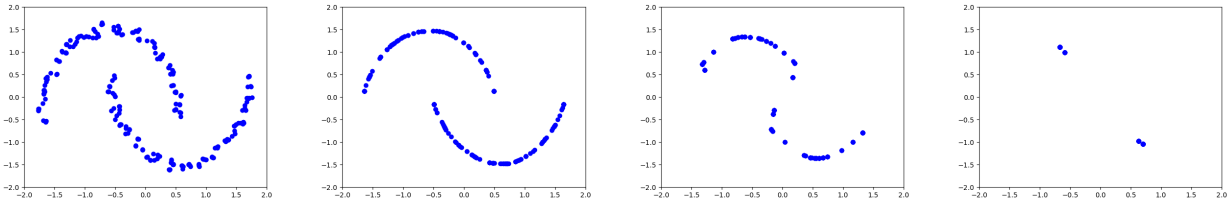
Figure 1: All points of a dataset with two clusters: overlapping final positions and the trajectories of all points on the left, initial positions and clustering on the right

2 RELATED WORK

The most popular related work is Mean Shift, which has been initially proposed in [5]. Its core idea is to regard the points in an area around each point and iteratively shift the center of this area to the mean of the points lying in the previously regarded area. The Mean Shift method became popular in the field of clustering by the work of [2]. The core concept of Mean Shift is to determine the maxima (referred to as modes) of a given density function. In contrast to Rock, Mean Shift requires a kernel function K by which the weights of the nearby points are determined in order to re-estimate the mean. Further the used kernel requires a size of the area to be regarded around each point, which is also referred to as bandwidth. The choice of the bandwidth is not trivial, additionally there can be the need of an adaptive bandwidth size.

Another algorithm for detecting modes is developed in [3]. In contrast to the Mean Shift approach they rely on k-Nearest Neighbors (k-NN). While Mean Shift relies on a kernel density estimate (KDE) the k-NN approach relies on a k-NN density estimate. The algorithm in [3] defines for each input point x_i a pointer to the point within the k-Nearest Neighbors with the highest k-NN density. The process is repeated until a pointer points to itself, which represents then the local mode of x_i. However, in contrast to Rock, this method also relies on a density definition, where the density of a point is defined as inversely proportional to the distance of the k-th nearest neighbor. In a more recent work, [8] proposed a k-NN clustering method relying on consensus clustering, in which results of several clustering trials are combined in order to get a better partition.

Regarding methods which generate a backbone of a cluster, in a more recent work named ABACUS [1] the authors propose a method which is specialized in identifying skeletons of clusters using k-NN. However ABACUS and Rock differ in two major aspects. First, Rock does not primarily aim for the skeletons of a cluster but for a representative for each detected cluster (body to representative). In contrast ABACUS is targeted to find first a



(a) t=1 (b) t=4 (c) t=7 (d) t=14

Figure 2: Actual positions of the points at the given time t. First "skeletons" of the clusters emerge, which then tighten to divisible clusters.

backbone of a cluster and then expand from the backbone until a border of a cluster is reached (backbone to body). Secondly, ABACUS uses a static k for the k-NN approach. In our work in progress, we provide a method where k changes over each iteration step.

In this paper we compare Rock with Mean Shift and k-Means [7]. K-Means is a partitioning clustering algorithm where the user provides the number of clusters k which shall be detected. The cluster model of k-Means provides as a result the cluster centroids μ and the assignment of the datapoints to their centroids. Finally we evaluate the performance of Rock compared to DBSCAN [4], which is a density based clustering method. We compare to density based clustering since it is in contrast to other types of clustering algorithms capable of detecting non-convex shaped clusters. DBSCAN requires an ε-range and a minimum number of points (*minPts*) which shall be located within this ε-range. Through these two parameters the user can control how dense the detected clusters have to be.

3 THE ALGORITHM

Rock implements the best of both worlds, partitioning and density-based clustering. It is able to detect non-convex clusters and clusters of highly different densities. Since we use the k-Nearest Neighbors as main indication for cluster membership the algorithm works for arbitrary numbers of dimensions. Given a query point p and a number of neighbors k, the k-Nearest Neighbors at timepoint t (we will explain that later) are defined as a set $kNN_t(p,k) \subseteq DB$ with exactly k objects, such that for all of these objects $\{o_1, o_2, ..., o_k\}$ the distance to the query point p is less or equal to the distance of any other data point o' to p in our data base DB:

$$\forall o \in kNN_t(p,k), \forall o' \in DB \setminus kNN_t(p,k) : d_t(p,o) \leq d_t(p,o')$$

t refers to the timepoint at which the data base DB is looked at, since each point will have a different position at different timepoints in our algorithm. Similarly we use $d_t(p,o)$ to describe the distance between points p and o for the status of the data base at timepoint t. For detecting the k-Nearest Neighbors, our implementation of Rock uses a Balltree [9] as index structure according to the sklearn reference.

The idea of Rock is, that every point moves gradually to the representative point ζ of its cluster by "wandering" into the center of its k-Nearest Neighbors in every step. So Rock does not need any other information than where the k-NN of each point are located at time step t, which makes it easily parallelizable. Equation 1 gives us the position ζ of a point p at a time t using $k(t)$ as the number of regarded neighbors: we get the new position $\zeta(p,k,t)$ of p by calculating the mean position of all objects $o \in kNN_{t-1}(p,k(t))$, i.e. of all k-NN of this point p *at the previous time step.*

$$\begin{aligned} \zeta(p,k,t) &= \sum_{o \in kNN_{t-1}(p,k(t))} \frac{\zeta(o,k,t-1)}{k(t)} \\ \zeta(p,k,0) &= initialPosition(p) \end{aligned} \tag{1}$$

We increase k linearly over time and stop if the algorithm converged or a given maximal time *tmax* is reached. Of course we do not only store the actual position at a time of each point p but also its initial position $initialPosition(p)$. With that we are in the end able to put all points which have almost the same actual position into one cluster. This is when the distance is smaller than ε, which is described in equation 2. Therefore we regard the 1-NN-distances of all points, which is the distance to the first nearest neighbor each, and use half of the average 1-NN distances of all points as ε:

$$\varepsilon = \frac{\sum_{p \in DB} \left(\sum_{o \in kNN(p,1)} dist(p,o) \right)}{2 \cdot |DB|} \tag{2}$$

We choose this ε since it delivers an intuitively good measure for how close two points have to be to be "mergeable" in regards to their initial distribution.
In summary the algorithm looks like follows:

Algorithm 1 Rock(pointset, tmax)

```
changed= true
t=0
while changed and t < tmax do
    k=k(t)
    changed=false
    for p in pointset do
        oldPosition= p.getOldPosition()
        newPosition= meanOfKNN(pointset, p, k, t)
        p.setActualPosition(newPosition)
        if distance(oldPosition, newPosition)<ε then
            changed=true
    t++
```

Note that *tmax* is not mandatory a hyperparameter and can be set beforehand to the same value for various experiments as we will show in section 4. If we took a static k, most of the points would only shrink together with their exact k-Nearest Neighbors. Thus, we increase k over time, to receive stable clusters. The minimal value with which we will also start should be $k = 3$, since observing less nearest neighbors (including the point itself)

would not lead to anything. The highest useful value for k is $\frac{n}{2}$ for a dataset of n points since if more neighbors would be regarded, it would lead to only one big cluster. We want to reach that highest value at the maximal number of iterations $tmax$, thus we obtain for linear increase:

$$k(t) = \frac{0.5n - 3}{tmax} \cdot t + 3$$

In order that k actually increases in every step the slope of $k(t)$ should be at least 1. Therefore $tmax$ should be chosen such that $tmax < 0.5n - 3$. Choosing $tmax$ significantly too low would mean to early regard a high number k of nearest neighbors, which potentially belong to other clusters, and therefore merging several clusters into one. Vice versa, a significantly too high $tmax$ could result in clusters splitting up.

Figure 2 shows the positions of all points at expressive points in time t while clustering the two moons dataset, which is further described in section 4.1. Regarding that, we already note an additional advantage: The skeleton which emerges already in early steps for each cluster could be used for anytime results or elimination of outliers, which we plan to examine further in future work.

3.1 Complexity

Having described our algorithm, we now elaborate on the runtime complexity of Rock and compare it on a theoretical basis to k-means and MeanShift. Focusing first on the used index structure, according to [9] the runtime complexity of a Ball tree is $O(d \log(N))$ where d denotes the dimensionality and N the number of data points. The runtime complexity of MeanShift is $O(iN^2)$ where i denotes the number of iterations. k-Means has a runtime complexity of $O(Nkid)$ where k refers to the number of clusters which shall be found. For Rock, using the Balltree structure, we get a runtime complexity of $O(kNd \log(N))$ where k denotes here the number of k-NN. Conclusively it can be stated that our method performs with regards to its time complexity between the performance of k-means and MeanShift. Since this is a work in progress, it may be viable to consider query-strategies which rely on different structures, such as e.g., Locality-Sensitive-Hashing (LSH) as used in [10].

4 EVALUATION

We tested Rock on several datasets and compared our results to the ones obtained by DBSCAN, k-Means and Mean Shift, since Rock positions itself between them: it finds non-convex shaped clusters as DBSCAN, but as well finds clusters of different densities with according representatives as k-Means and Mean Shift. As quality measures we use the Normalized Mutual Information (NMI) and the Adjusted Rand Index (ARI) in regards to the ground truth. All experiments were executed with $tmax = 15$, which is an empirically determined good value for the normalized data we use (i.e. removing the mean and scaling to unit variance). For all comparative methods the best parameters have been determined by iterating over a parameter range [1] and choosing those parameters which yield the best NMI resp. the best ARI for each of the methods. This shows again the simplicity of choosing the one hyperparameter Rock requires in comparison to the several parameters of our competitors, as Rock is not too sensitive to $tmax$, as we will show in future work due to the lack of space.

[1] We tested ε with steps of 0.1 in a range of $[0.1, 2.0]$, $minPts$ with steps of 1 in a range of $[2, 10]$ and the bandwidth with steps of 0.1 in a range of $[0.1, 2.0]$

4.1 Two Moons

The dataset as shown in Figure 3 is a classical and well known dataset consisting of two moon-shaped clusters. For Mean Shift a bandwidth of 0.6 yielded the best NMI and a bandwidth of 1.2 yielded the best ARI. For DBSCAN $\varepsilon = 0.3$ and $minPts = 2$ yielded an NMI and ARI of 1 and for k-Means we chose $k = 2$. We tested the dataset with 250 datapoints and 5% noise, obtaining the optimal result with Rock as well as with DBSCAN. The non-density based methods k-Means and Mean Shift assigned a part of each moon to the false cluster as shown in Figure 3. As that, k-Means reaches an NMI of 0.36 and an ARI of 0.46. Mean Shift performed slightly better with an NMI of 0.53 (bandwidth 0.6) and an ARI of 0.50 (bandwidth 1.2). It may seem unintuitively first

Figure 3: The clustering of two moons as received by k-Means, Mean Shift and DBSCAN (in this order)

why Rock finds such shaped clusters even though regarding "only" the k-NN, but it makes sense considering that we slowly increase k. Figure 2 shows the development of the points' positions: First, the noise gets eliminated and the skeleton of the two clusters emerges. Then the ends of the clusters shrink up more and more into the middle of the clusters each. When the algorithm finishes at $tmax = 15$, there are only the two representatives of the clusters left as Figure 2d implies already.

4.2 Mouse

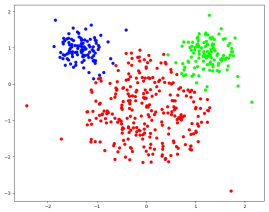


Figure 4: Result Rock returns for the mouse dataset

	NMI	ARI
k-Means	0.58	0.53
Mean Shift	0.67	0.66
DBSCAN	0.71	0.68
Rock	0.81	0.86

Table 1: Performances of the algorithms on the mouse dataset

To show how Rock deals with colliding clusters, we chose the mouse dataset. It consists of three round clusters of different densities which blend into each other at two points, where the "ears" of the mouse touch the "head". We conducted the experiment with Mean Shift using a bandwidth of 0.9, with DBSCAN using an ε-range of 0.2 and $minPts = 4$, and k-Means with $k = 3$. While Rock is able to distinguish the three clusters without a problem, the other algorithms cannot, as Figures 4.2 and 5 show: DBSCAN does not find the correct dividing line between head and right ear, k-Means puts a lot of the points from the head to the ears, and Mean Shift performs only slightly better than k-Means. The exact values of the resulting NMI and ARI can be seen in table 1.

4.3 Iris

Iris [6] may be the most famous dataset and one of the most challenging ones, since two of the three clusters are very difficult

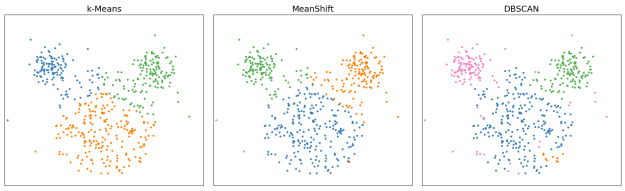


Figure 5: Clustering of the mouse dataset as obtained by the comparative methods k-Means, Mean Shift and DBSCAN (in this order).

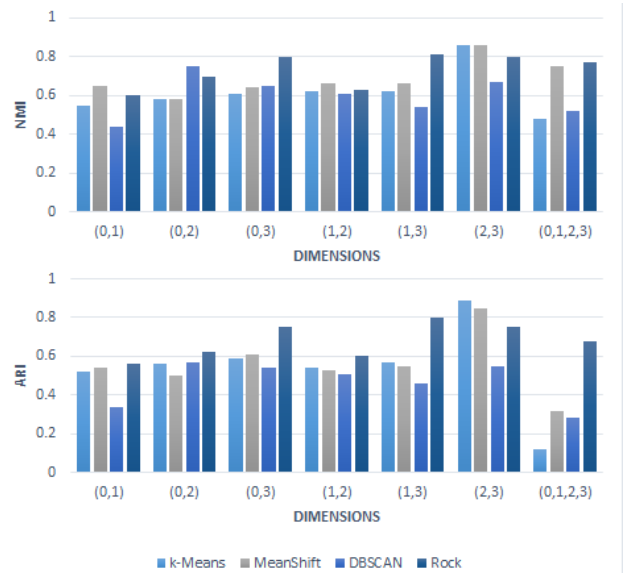


Figure 6: The NMIs (top) and ARIs (bottom) reached by the comparative methods and Rock, using different combinations of dimensions of the Iris dataset respectively.

to separate for they are overlapping in virtually every dimension. Nevertheless, even here Rock delivers good results: depending on which two of the four given dimensions it regards, we receive an NMI between 0.6 and 0.8 and an ARI between 0.56 and 0.8. These results are very good in comparison to the other methods, as Figure 6 shows: regarding the NMI we achieve at least the second best value, but most of the times the very best, no matter which two dimensions of the dataset are regarded. Concerning the ARI we mostly surpass the other algorithms, partially by far, as for example by using dimensions 1 and 3 (which is also reflected in the NMI). Since we claimed in Section 3 that Rock works with higher number of dimensions, we conducted an experiment on the full four-dimensional Iris dataset. Here our method yields the best results compared to its competitors, especially with regards to the computed ARI as can be seen in the rightmost case in Figure 6. The exact parameters for the comparative methods are shown in table 2 in the appendix.

5 CONCLUSION

In conclusion we developed an algorithm which is able to autonomously find clusters of various kind. Due to the successive increase of neighbors being decisive to a point, we are able to find non-convex shaped clusters. Without this successive increase Rock would be equivalent to a simple k-Means, but with the increase we only obtain the advantages of k-Means: being simple and providing kind of a cluster centroid which gives us even more information about the found clusters than DBSCAN provides. With the right use of index structures Rock is not only effective but also efficient. We showed that Rock surpasses good methods like Mean Shift, DBSCAN and k-Means in regards to non-convex shaped clusters, colliding clusters of different densities and even overlapping clusters like in the Iris dataset. Since it regards k-NN and no other distance measures it works on high dimensional datasets as well as on two dimensional ones. The rate of increase of k will be treated in future work, as a logarithmic or hyperbolic increase could lead to interesting results. Moreover we are also curious to know the impact of using reverse k-Nearest Neighbors instead of the k-NN, as well as using the median or mode of the k-NN instead of the mean. The influence of the parameter $tmax$ was also not yet analyzed thoroughly. And, last but not least, the feature of Rock to create skeletons of the clusters before reducing them further to representatives could be used to detect advanced structures, eliminate noise, or deliver anytime results.

A APPENDIX

Dimensions	Mean Shift	DBSCAN		k-Means
	bandwidth	minPts	ε	k
(0, 1)	0.9	2, 8	0.4, 0.5	3
(0, 2)	0.8, 0.6	2	0.5	3
(0, 3)	0.7, 0.8	2	0.5	3
(1, 2)	1.0	2	0.59	3
(1, 3)	0.8	2	0.7	3
(2, 3)	0.5	2, 7	0.3, 0.2	3
(0, 1, 2, 3)	0.7	3	0.4	3

Table 2: Parameters used for the respective comparative methods on Iris. Multiple values indicate that the first value is optimizing the NMI and differed from the second value, which optimizes the ARI.

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 responsibilities for its content.

REFERENCES

[1] Vineet Chaoji, Geng Li, Hilmi Yildirim, and Mohammed J. Zaki. [n. d.]. *ABACUS: Mining Arbitrary Shaped Clusters from Large Datasets based on Backbone Identification.* 295–306.
[2] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17, 8 (Aug 1995), 790–799.
[3] Robert P. W. Duin, Ana L. N. Fred, Marco Loog, and Elżbieta Pękalska. 2012. *Mode Seeking Clustering by KNN and Mean Shift Evaluated.* Springer Berlin Heidelberg, Berlin, Heidelberg, 51–59. https://doi.org/10.1007/978-3-642-34166-3_6
[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[5] K. Fukunaga and L. Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21, 1 (January 1975), 32–40.
[6] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[7] S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.* 28, 2 (1982), 129–137.
[8] Jonas Nordhaug Myhre, Karl Øyvind Mikalsen, Sigurd Løkse, and Robert Jenssen. 2015. *Consensus Clustering Using kNN Mode Seeking.* Springer International Publishing, Cham, 175–186. https://doi.org/10.1007/978-3-319-19665-7_15
[9] Stephen M. Omohundro. 1989. *Five Balltree Construction Algorithms.* Technical Report.
[10] Ilan Shimshoni, Bogdan Georgescu, and Peter Meer. 2006. Adaptive Mean Shift Based Clustering in High Dimensions. *Nearest-neighbor methods in learning and vision: theory and practice* (2006), 203–220.