KMN - Removing Noise from K-Means Clustering Results

Benjamin Schelling and Claudia Plant

University of Vienna

Abstract. K-Means is one of the most important data mining techniques for scientists who want to analyze their data. But K-Means has the disadvantage that it is unable to handle noise data points. This paper proposes a technique that can be applied to the k-Means Clustering result to exclude noise data points. We refer to it as KMN (short for K-Means with Noise). This technique is compatible with the different strategies to initialize k-Means and determine the number of clusters. Moreover, it is completely parameter-free. The technique has been tested on artificial and real data sets to demonstrate its performance in comparison with other noise-excluding techniques for k-Means.

1 Introduction

A scientist who has not had much contact with data mining will use the simplest algorithms when he decides to use data mining techniques. The simplest and best known is probably k-Means [12]. More refined techniques, that offer the possibility of achieving better results are likely to be applied at a later stage, once the scientist has become familiar with the automatic labelling of the data and has learned to appreciate the additional information that such techniques might yield. K-Means is something like a gateway to data mining techniques. It has many advantages that predestine it for this purpose: Its simplicity, the comparatively good results and its runtime. But k-Means also has some disadvantages that are not to be neglected: The initialization that determines which (local) optimum the algorithm converges to, the need to set the k parameter and its inability to handle noise. This can be seen in Figure 1. K-Means will add each data point to a cluster, since the possibility, that a data point is a noise point is simply not supported in the algorithm. The first two problems, initialization and setting k, were examined in detail and various (sometimes very capable) strategies have been proposed, the last problem however received less recognition. This may be partly due to the fact that publicly available data sets rarely contain noise, which is why k-Means-based techniques that find and characterize noise did not seem so important at first. Recently, however, there have been more and more techniques that take noise into account when clustering.

1.1 Related Work

Clustering in the presence of noise is a classic field of research in data mining. Many techniques like DBSCAN [6] have been suggested and studied extensively.

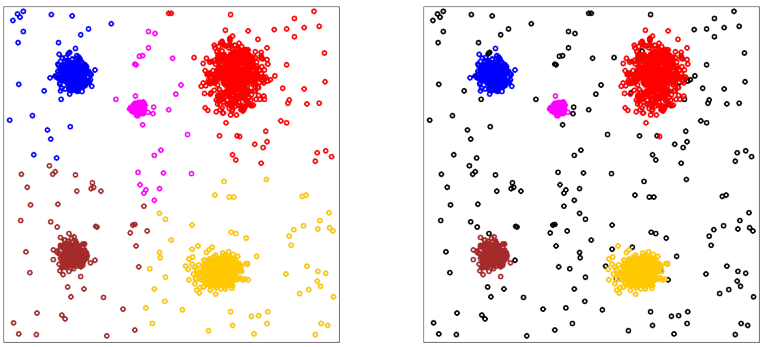


Fig. 1. A typical clustering result of k-Means, if the value of k is chosen correctly, and how the clustering actually should look like.

However, few of them are based on k-Means, hence will not be a likely first choice for a new user of data mining techniques. The focus here is exclusively on techniques which adapt k-Means to noisy data sets and thus offer the possibility to stay within the framework of k-Means while they are still dealing with noisy data sets.

The best known of these clustering-in-the-presence-of-noise techniques based on k-Means is k-Means-- from Chawla et al. [5]. The problem is that it asks for the number of outliers as a parameter for the functionality of the technique. Chawla et al. stated in their paper that "all existing outlier-detection methods use one or more parameters". This seems to be largely correct for the outlier clustering techniques that are based on k-Means. Similarly, the algorithm KMOR proposed by Gan et al. [7] asks for two additional parameters, one of which is the maximum number of outliers. The algorithm ODC [1] has the "difference" between outliers and real data points as a parameter, while Neo-k-Means [16] asks for two parameters α and β, which are also related to the assignment of data points to a cluster, i. e. the number of outliers.

The aim of this work is to find a way to remove noise points from the clusters without additional parameters. Having to set parameters will most likely prevent inexperienced users from using data mining techniques, hence, the ideal clustering technique is one completely without parameters. One could argue that the other problems of k-Means might have already detered possible users, but these problems have many solutions, many of which are implemented in publicly available software. It is easy to add the option of, say, X-Means [14] to estimate the value of k and k-Means++ [2] to achieve a good initialization. The ideal solution would be to add the option to remove noise too, without setting additional parameters.

1.2 Contribution

We present here a k-Means based technique that adapts and expands k-Means to noisy data sets.

- The technique presented in this work, KMN, can efficiently remove noise from k-Means clustering results without additional parameters.
- It does so, while being deterministic. For a k-Means clustering result, the excluded noise points are always the same.
- KMN offers a higher independence from the chosen k for k-Means. This work demonstrates in section 3 how KMN fares for wrongly chosen values of k and shows the resilience it gives in relation to these values.
- It is fully compatible with other k-Means-based techniques that aim to improve k-Means like X-Means and k-Means++.

2 The algorithm

The main intention of this work is to present an approach to exclude noise from the clustering result of k-Means, to which we refer to as KMN. The algorithm starts with a k-Means clustering result and tries to locate the areas where noise is prevalent. K-Means follows a centre-oriented approach, in which data points are assigned to the nearest centre. The centre is then updated as the mean of the assigned data points. This means that the closer a datapoint is to a centre, the more likely it is to be assigned correctly (provided that k-Means is a fitting technique for the dataset and k is correctly selected). Or in other words: the closer a data point is to the centre, the more likely is it correctly assigned. The further away, the more likely is it that it should be assigned to another centre or regarded as noise.

The search for noise should therefore begin at the locations furthest from the cluster centres. The clusters in k-Means are Voronoi cells and between two neighbouring Voronoi cells there is a hyperplane that separates them. At the intersections of these hyperplanes one will find the point farthest from the centres, as shown in Figure 2. Let us call this point m.

In an d-dimensional data space $d+1$ voronoi cells determine such an intersection point. This point m is basically defined as the point in the data space, where $||v_j - m|| = r$, $j \in \{1, 2, \ldots, d+1\}$, holds; v_j are the centres of the voronoi cells, i.e. the centres that k-Means finds, r is the distance of m to the centres of the voronoi cells. If one of the voronoi cell centres had a distance to m unequal r, m would be assigned to the voronoi cell with the smallest distance.

When these intersections are found, the assumption is valid that the area of these intersections should be considered to contain noise data points. Retaining the spirit of k-Means, the algorithm then simply opens a new Voronoi cell at the intersection, but one in which all data points within its boundaries are regarded as noise (see Figure 2.2). It may not always be a good decision to open such a noise voronoi cell, because it is possible that such a noise voronoi cell contains data points and not (only) noise points. It is necessary to have a criterion to

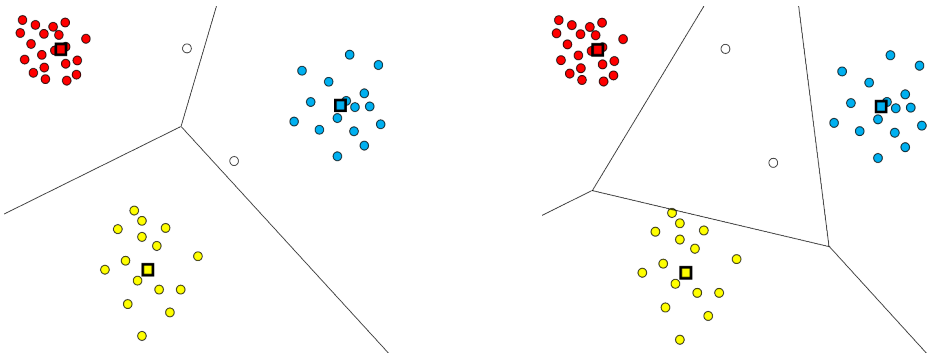


Fig. 2. Three simple clusters and two outliers. The squares are the centres of the clusters as found by k-Means which determine the voronoi cell walls. This shows, if k-Means is a suitable technique, that the data points near the centres are correctly assigned, while the more distant data points are more likely to be wrongly assigned. When a new Voronoi cell is opened at the intersection of Voronoi cells, the noise data points are separated from the clusters.

decide whether or not to open such a noise voronoi cell. KMN uses the principle of Minimum Description Length (MDL).

MDL is a well established principle in the Data Mining community and is used in various technique like X-Means [14]. The basic assumption behind MDL is that the coding cost for a clustering result depends on how good the clustering is. The coding cost is basically an estimate of how much memory is needed to encode the clustering. If the clustering is good, only little memory is needed, but if the clustering is far off, i. e. the model used to encode the data is not fitting, the encoding requires a lot of memory.

If the criterion is applied and the Voronoi cells are either retained or discarded, the next iteration can begin. We see in Figure 2.2 that there are new Voronoi cell intersections and the same steps can be taken as before. Before this happens, the centres of the clusters are updated. The process is iterated until no more intersection improve the clustering, according to the criterion.

This is, very briefly, the procedure of the algorithm. Let us now elaborate on that.

2.1 Finding the Voronoi Intersections

There are several ways to find the intersections of the Voronoi cells. The most obvious way would be to find m using geometric calculations. In this approach one would to work with the equation $||v_j - m|| = r$ directly. This equation is the formula for a sphere with radius r and middlepoint m. The $d+1$ points v_j are given and therefore uniquely determine the centre m. To calculate m, this would entail inverting a $d+1 \times d+1$-matrix, which would mean computations in the order of $O((d+1)^3)$. While this might be acceptable, the difficulty lies with the

choice of the v_j. If there are k centres of k-Means in a d-dimensional data space, without knowing the adjacencies of the centres, one would have to compute all $\binom{k}{d+1}$ possible combinations to find the correct combinations of voronoi cells. This is comparatively expensive and should be avoided.

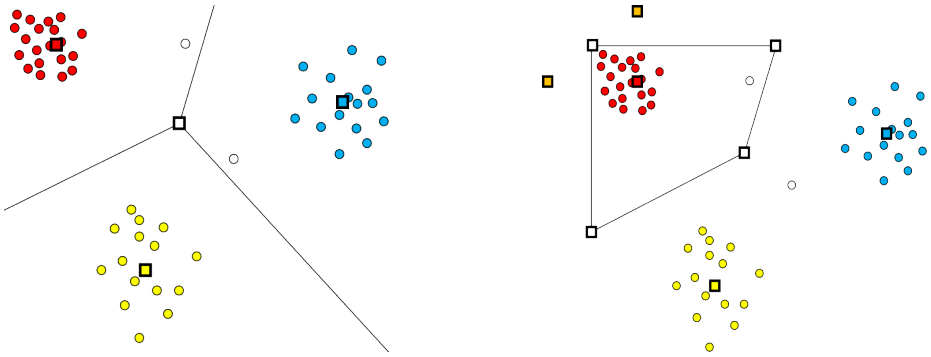


Fig. 3. Convex Hull and intersection. The three clusters determine one voronoi cell intersection, which is represented by the white square. The centres of the other clusters as well as the projections of its own centre determine a bounded polytope, i.e. the voronoi cell.

This technique uses the Avis-Fukuda algorithm [3], which is specialized in finding Voronoi intersections. More precisely, Avis-Fukuda takes the centre of each Voronoi cell, i. e. the centres that k-Means finds, as input and calculates the intersection points of these Voronoi cells. The algorithm is based on a linear optimization approach. It works according to the following principle: it starts with a centre of a Voronoi cell and the other centres that bound it. This forms a convex hull around the centre, a polyhedron, with the corners of this hull being the intersections we are looking for. It calculates the nearest vertex with a linear optimization approach and then follows the beams of the convex hull along to find the other vertices. Avis-Fukuda follows the optimal path and finds all nodes, i. e. all Voronoi intersections. The algorithm uses "Bland's rule" to ensure that the path is the optimal one and that all vertices are found. In this way, all corner points of a convex hull, i. e. all Voronoi intersections, are found.

A very simple example of this is illustrated in Figure 3. The point of intersection between the three Voronoi cells is easy to find, but the Voronoi cell is unbounded. Therefore, we project the centre for all Voronoi cells beyond the data set boundaries, so that all clusters have a limited convex hull defined by the centres found by k-Means and the added projections. The Avis Fukuda algorithm can then be used to find the vertices (represented in Figure 3.2 as white squares). The algorithm continues by finding such an vertex and then moving on to the others. In a two-dimensional environment like this, it is rather straightforward. After the first step, there is always only one direction in which the algorithm can

advance. It follows the rays of the convex hull until it reconnects with the first vertex. The Avis-Fukuda algorithm has then found all nodes, i. e. all Voronoi intersections for the cluster.

The Avis-Fukuda algorithm is estimated to have a runtime of $\mathcal{O}(k \cdot d \cdot v)$, where v is the number of vertices. Since we apply the algorithm for each of the k voronoi cells, we have a runtime of $k \cdot \mathcal{O}(k \cdot d \cdot v)$.

2.2 The MDL-Criterion

The algorithm has found the possible intersections of the Voronoi cells. However, the question now arises as to whether the algorithm should open new Voronoi noise cells or not. MDL assumes that lower coding costs imply a better clustering. Thus, if the new Voronoi noise cell reduces the total coding costs, the algorithm keeps the new Voronoi noise cell. The coding costs consist of two parts: The coding costs for the model $L(M)$ and the coding costs for the data $L(D|M)$. The total coding cost is therefore $L(M, D) = L(M) + L(D|M)$. K is the number of clusters C_i; N is the number of data points x; p_i is the number of parameters. Hence, coding cost is given by the following equation:

$$\begin{aligned} L(M,D) =& L(M) && + L(D|M) \\ =& \sum_{i=1}^{K} \sum_{j=1}^{|C_i|} log_2(\frac{N}{|C_i|}) + \sum_{i=1}^{K} \frac{p_i}{2} log_2(|C_i|) && - \sum_{i=1}^{K} \sum_{x \in C_i} log_2(pdf(x)) \end{aligned}$$

The model coding costs $L(M)$ is not difficult to calculate. Basically, one only needs to know the cluster sizes. The data encoding cost $L(D|M)$ is the more cumbersome part, since one needs to know the distribution of the data points. For algorithms like DBSCAN [6] that are not based on a probability distribution, this can be difficult, but k-Means is based on the assumption of a Gaussian distributed cluster, where the variance is the same in all directions. Therefore we assume the data points to be Gaussian distributed in a cluster. Since the variance is the same in all directions, the distance of a data point from the centre is sufficient to determine its probability. We need two parameters to calculate the normal distribution of a cluster, the expected value and the variance/standard deviation of the cluster. The expected value is easy to calculate, it is simply the centre of the cluster. The variance is a slightly more difficult one.

Let x be a random point in the Gaussian cluster. We assume the cluster to be centred at 0. Every axis of the cluster is $N(0, \alpha)$ distributed, i.e. normal distributed with variance α, and we want to find α as it is the variance we are looking for. We calculate the distance of x to the centre $dist(x, 0) = \sqrt{\sum_{i=1}^{d} (\alpha X_i)^2}$, with X_i being a $N(0, 1)$ distributed random variable, so αX_i is $N(0, \alpha)$ distributed. We have:

$$dist(x, 0) = \sqrt{\sum_{i=1}^{d} (\alpha X_i)^2} = \sqrt{\alpha^2 \sum_{i=1}^{d} (X_i)^2} = \alpha \sqrt{\sum_{i=1}^{d} X_i^2}$$

The term $Y = \sqrt{\sum_{i=1}^{d} X_i^2}$ is known in the literature (e.g. [9]) as being Chi-distributed and with that we have its probability distribution, which we label as $pdf(x)$. The difference is that we have $\alpha\sqrt{\sum_{i=1}^{d} X_i^2}$, but probability theory tells us, if $\sqrt{\sum_{i=1}^{d} X_i^2} \sim pdf(x)$, then $\alpha\sqrt{\sum_{i=1}^{d} X_i^2} \sim \frac{1}{|\alpha|} pdf(\frac{x}{|\alpha|})$. We now know the form of the probability distribution.

The formula for the variance is $Var[Y] = E[Y^2] - E[Y]^2$. We can rewrite $E[Y^2]$ to $E[Y^2] = E[(\sqrt{\sum_{i=1}^{d}(\alpha X_i)^2})^2] = \alpha^2 E[\sum_{i=1}^{d} X_i^2]$. The literature tells us $Y^2 = \sum_{i=1}^{d} X_i^2$ is Chi-squared distributed and has a mean of d. Hence:

$$Var[Y] = \alpha^2 d - E[Y]^2$$
$$\alpha = \sqrt{\frac{Var[Y] + E[Y]^2}{d}}$$

The variance and mean values are the variance and mean of the distances of all data points to the centre, therefore we can compute them directly and get α.

$$pdf(x) = \frac{x^{d-1} e^{-\frac{x^2}{2\alpha^2}}}{2^{\frac{d}{2}-1} \alpha^d \Gamma(\frac{d}{2})}$$

is therefore the probability density we were looking for. Γ is the standard Gamma-function.

One could also estimate the variance differently, e. g. like X-Means [14], but this approach seems to be more compatible with the heuristics of our approach. Both have the same mathematical validity, but the use of the chi distribution for the *pdf* seems to take better account of the distortion of Gaussian spheres by outliers in the dataset. With the probability density function found, the coding costs of the data can be calculated and the MDL criterion is set up to test the Voronoi intersections.

The question that remains is which PDF should be used to model the noise. The obvious notion is to assume uniformly distributed noise, but this has the disadvantage of being massively distorted by outliers. Assume that the data set is completely within the $[0.1]^d$-cube. A single data point (2,2, ..., 2) would change the volume of the data set from 1 to 2^d and thus also change the probability of a noise data point by a factor of 2^d. To make it more resistant to such extreme outliers, the algorithm assumes that the noise is also Gaussian distributed. The parameters for the noise distribution are calculated as before, whereby the mean value is the average value of all data points and the variance of the noise is the variance of all data points.

Let us go through the steps of the algorithm so far with Figure 4. We have the old Voronoi noise cells in white that are given by the centres of k-Means and the projections of the cluster centres. They now determine the Voronoi cell intersections that are shown in gray. The MDL criterion is used to check whether the new Voronoi noise cells are to be opened at these intersections. Three of the

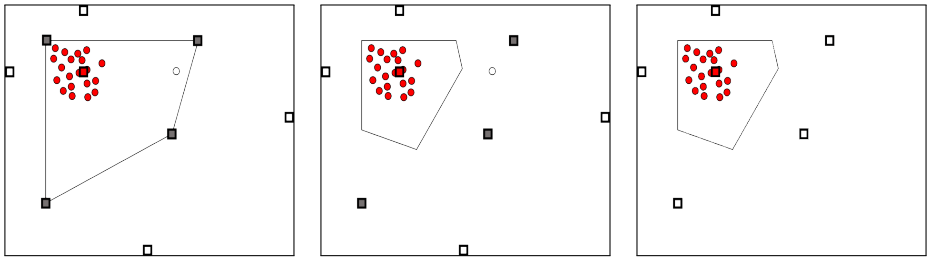


Fig. 4. The convex hull of a cluster in the course of one iteration. First, all potentially neighbouring Voronoi cells are determined (here only represented by their centres) and the intersection points of the cells are calculated; then those intersection points that open "good" noise cells, according to the MDL principle, are retained. Finally, those noise cells that are no longer adjacent are removed.

four are accepted because they either do not change or reduce the coding cost. The fourth option would, however, massively increase the coding cost as it would move many data points from the cluster to a noise cell. Therefore, only three of these intersections are kept and the new convex hull of the cluster is described by the Voronoi cells, which centres can be seen in Figure 4.2. One can see that not all these centres are necessary any more, since two of the old centres are no longer adjacent to the Voronoi cell. When eliminated, the situation would be as shown in Figure 4.3.

If one were to keep these unnecessary noise cells, the next iteration would take slightly longer and the next one somewhat longer. Depending on the number of iterations the runtime would increase massively, so the question arises how to eliminate the unnecessary centres.

2.3 Finding the Voronoi Adjacencies

The Avis-Fukuda algorithm offers the possibility to calculate the Voronoi cell adjacencies, but is not focused on this by default. Mendez et al [13] created an algorithm based on linear optimization, which specializes in this.

The algorithm translates the Voronoi adjacency problem into the linear optimization version of it and then continues the dual problem as it is more practical to solve. It begins with the equivalent of two randomly selected centres of Voronoi cells and tests whether the point in the middle is located in one of the two Voronoi cells. If this is the case, the Voronoi cells are adjacent to each other. If this is not the case, the dual problem forms the basis for the linear optimization problem, which could contain information about the neighbourhood of several other Voronoi cells. Therefore, the algorithm is faster than the Avis Fukuda algorithm, which calculates the adjacency of all pairs of Voronoi cells (see [13] for details). Mendez et al. estimate the runtime of their algorithm as $\mathcal{O}(k^2 \cdot f(k,d))$, where f is unknown, but not worse than polynomial.

By using the Mendez algorithm, our technology can now eliminate the unnecessary centres for our problem. The next iteration would now start with the centres shown in Figure 4.3 and repeat the same steps as before. The algorithm updates the centres of the clusters in the same way as k-Means, resulting in a small change of the Voronoi cell structure. If the algorithm rejects all new possible Voronoi noise cells of an iteration, then it is converged and the cluster is marked as "fully cut out".

Algorithm 1 KMN

Require: k-Means clustering result D

procedure KMN(D)
  Initialize: Compute initial voronoi cell adjacencies ▷ $\mathcal{O}(k^2 \cdot f(k,d))$
  while Coding cost decreases **do** ▷ l times
    Find v **voronoi cell intersections** ▷ $k \cdot \mathcal{O}(k \cdot d \cdot v)$
    Check intersections v **with MDL** ▷ $\mathcal{O}(v \cdot n)$
    Compute adjacencies ▷ $\mathcal{O}(k^2 \cdot f(k,d))$
    Update cluster centres ▷ $\mathcal{O}(k \cdot n)$
    Compute coding cost ▷ $\mathcal{O}(n)$
  end while
  return Cluster $C_1, \ldots, C_k$ and Noise N
end procedure

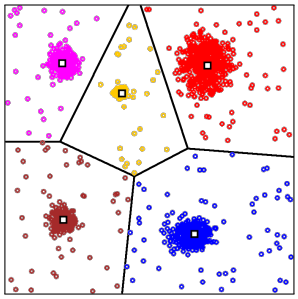
2.4 Pseudo Code

Following the pseudocode shown in Algorithm 1 we can assume that the runtime of this approach is in the order of $\mathcal{O}(l \cdot k^2 \cdot d \cdot v^2 \cdot n \cdot f(k,d))$. This shows us that this approach is relatively stable in terms of the data size n, but is somewhat more influenced by the dimensionality d and the number of clusters k. This is logical because the algorithm has to calculate the Voronoi cell intersections and adjacencies. This is independent of the size of the data set itself and could also be seen as a constant; the algorithm itself would then have a purely linear dependency on n, like k-Means itself has.

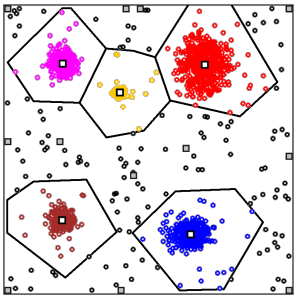
2.5 Performance on Running Example

The theory behind this technique has been presented, now let us see how it performs on our running example. In Figure 5a we have the result of k-Means on a simple dataset, consisting of 5 clusters and 10% noise. The clusters are Gaussian distributed, as k-Means assumes, and differ quite strongly in density and size. The data set is well suited for k-Means and all clusters are well separated by it. The main error lies in the wrongly assigned noise points.

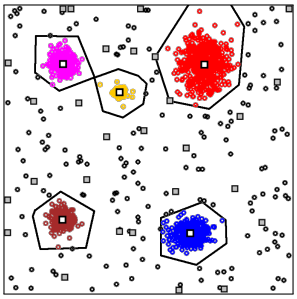
The algorithm iteratively computes the intersections, tests them with the MDL criterion, excludes noise points and updates the centres. The convex hulls



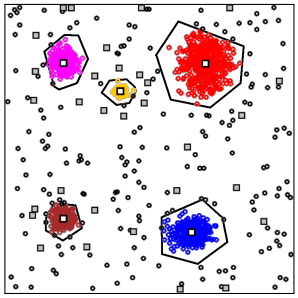
(a) k-Means with k=5. NMI is 0.82.



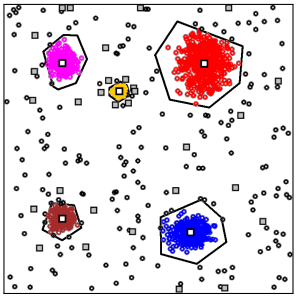
(b) The first iteration



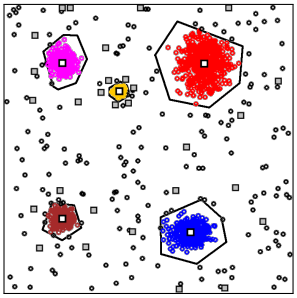
(c) Most noise is found after the second iteration.



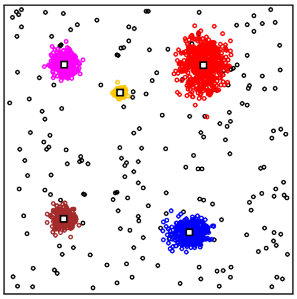
(d) The convex hulls have clearly shrunk.



(e) Some of the convex hulls do not change any more.



(f) The final result. The clusters are separated from the noise. NMI is 0.94.



(g) Ground truth.

Fig. 5. The stages of KMN from the result of k-Means to the final clustering. Cluster are displayed in different colours. Cluster-centre as white squares, centre of adjacent Voronoi cells with grey squares. The last Figure shows the ground truth. Figures best viewed in colour.

shrink until no more Voronoi noise cells are kept. For some of the clusters, this happens earlier than for others due to their spread. At the end almost all noise points are found and the clusters are cut out near ideally. The improvement can

also be measured with the help of the "normalized mutual information"-measure (NMI) [15]. The NMI value increases from 0.82 for the k-Means result to 0.94 for the result of our approach. Some of the outliers are located in the middle of a cluster and therefore cannot be recognized as such. Nevertheless, they are considered to be outliers. Therefore, a perfect NMI of 1.0 is impossible to obtain and 0.94 is almost the best result one can get.

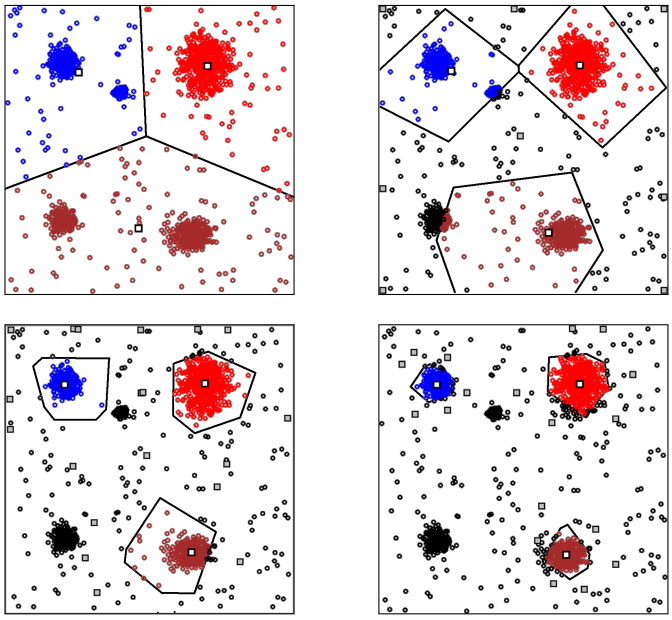


Fig. 6. The behaviour of the algorithm for a too small value of $k = 3$. Depicted is the Result of k-Means, the state after the first iteration, the second iteration and the final result.

3 Resilience in Regards to k

This technique has another advantage, which we would like to present now. For regular k-Means, setting k is one of the most important decisions and k is often difficult to estimate. Tools like X-Means [14] help the user with this decision, but are not necessarily correct. KMN now gives some leeway for this decision. We see this in our current example with a wrong k. If we have chosen a value of k that is too small, e.g. $k = 3$, k-Means can by default not separate all clusters. The clustering result will necessarily look similar to the one in Figure 6.1, where two clusters are merged into one. KMN now has the advantage that it eliminates data points from a cluster if they do not fit. We see the result in the first iteration. The

sub-clusters were (mostly) excluded and added to the noise points. The centre is updated (we see that it moves from Figure 6.2 to Figure 6.3) and moves to one of the correct clusters in the following iterations. At the end (Figure 6.4) three of the five clusters are found and separated from the noise, while two of them are simply added to the noise.

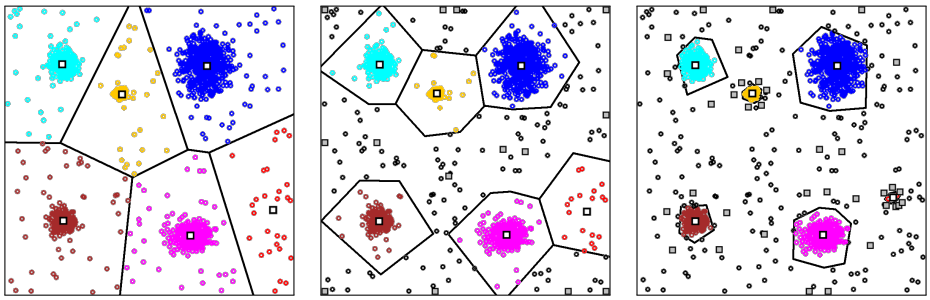


Fig. 7. K is here set to 6. Depicted is the result of k-Means, the state after the first and the second iteration, as well as the final result.

This resilience is also supported in the opposite direction. In Figure 7.1 we have k set to 6. Therefore, k-Means assigns some of the noise points to a separate cluster consisting only of noise points. We see that this "wrong" cluster loses some of its data points in the following iteration (Figure 7.2). It decreases from iteration to iteration until it is practically emptied (Figure 7.4). KMN has reduced the value of k from 6 to 5 by declaring a cluster as completely empty. We see that this is no coincidence if we choose an even bigger k of 10. The result is shown in Figure 8.1 to 8.3. K-Means has found three of the correct clusters (with some extra noise), three clusters consisting of pure noise and divided two clusters into half. When KMN is now applied to this result, the three correct clusters iteratively lose noise and converge to their correct shape. The clusters, which are divided into two halves, also lose their noise points, but remain divided into two halves. KMN does not currently have a function to merge clusters, but we hope to extend KMN to do so in the future. The three "false" clusters that consist only of noise, get most of their data points reassigned to noise. In the end, they consist of no or almost no data points. They effectively disappear.

We see that in the end, KMN may not be a technique for correctly estimating k, but it gives quite some leeway for the correct estimation. This does take some pressure from the estimation of k and techniques like X-Means are given a wider range of correct values.

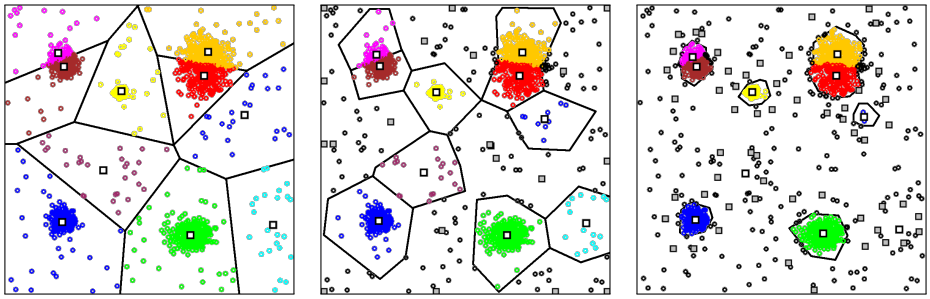


Fig. 8. K is here set to 10. Depicted is the result of k-Means, the state after the first iteration and the final result.

4 Experiments

Synthetic data We have compared the algorithms on the running example and the clustering results are shown in Table 1. The reason for using NMI as a measure for clustering lies in the opinion to see KMN as a clustering-in-the-presence-of-noise-technique. K-Means-- has been given the correct value for the outliers, for Neo-k-Means we used the internal estimator for the parameters. If a data point was assigned to more than one cluster by Neo-k-Means, it was assigned to the nearest centre. Each experiment was repeated 50 times and the average value is displayed in Table 1. We can see that KMN clearly outperforms k-Means-- and Neo-k-Means if the correct value of k is specified, but even if k is off.

Table 1. NMI Values for our running example for varying values of k.

	k=5	k=2	k=3	k=4	k=6	k=10
KMN	**0.874**	**0.412**	**0.675**	**0.813**	**0.850**	**0.780**
k-Means--	0.824	0.410	0.595	0.749	0.816	0.712
Neo-k-Means	0.760	0.354	0.546	0.713	0.808	0.722
k-Means	0.765	0.352	0.554	0.708	0.793	0.755

Real world data The difficulty of testing outlier detecting cluster techniques lies in the lack of suitable datasets, i. e. datasets containing noise data points. Campos et al. have compiled a list of possible datasets that can be used for such techniques [4]. Most of these are UCI datasets for which some of the classes have been declared noise. Most of these are also somewhat unsuitable for k-Means-based techniques, since the cluster results of k-Means have a very low quality (i.e. very low NMI). So there are very few datasets that we can use. Due to

this, we have also included another UCI dataset showing the behaviour of the algorithm on an outlier-free datasets.

Table 2. NMI Values for real word data sets.

	Hayes-Roth	Glass	WBC
KMN	**0.10**	**0.34**	0.56
k-Means--	0.08	0.33	0.34 (l=120) 0.78(l=241) 0.45 (l=361)
Neo-k-Means	0.08	**0.34**	0.00

The first dataset in Table 2, Hayes-Roth, is without outliers. The outlier parameter of k-Means-- was therefore set to 0, which means that k-Means-- gives the same results as k-Means. Therefore, we see that KMN improves on the result of k-Means. This is because k-Means assigns some data points to the wrong cluster, which KMN finds and identifies as noise. KMN notices that they do not belong in the current cluster. Clustering noise-free datasets often yields a small improvement in clustering quality compared to k-Means. Not enough to use KMN on a data set, where noise is known not to be present, but enough to warrant mention.

The data sets Glass and WBC both contain outliers. One has to keep in mind that k-Means-- needs to know the number of outliers, which is often very difficult to estimate, while KMN is parameter-free. On the WBC data set k-Means-- fares better when given the correct number of outlier. The NMI-values become identical when the outlier-number is off by roughly 30% and when the value is off by more than that, KMN delivers the better results.

The data sets were each clustered 50 times per algorithm. The runtime of KMN for WBC proved to be quite high and hence only one iteration was performed. All cluster-algorithms were given the correct values of k on the data sets.

5 Outlook and Conclusion

We wanted to create an algorithm that would be able to remove noise data points from a k-Means clustering and could achieve this without any additional parameters. In Figure 1 we see how much a k-Means clustering result can deviate from the correct clustering, even though the data set is well suited for k-Means, simply due to the noise data points that k-Means cannot account for. Through KMN we have now developed an additional algorithm for k-Means, which can be used to remove noise data points. Due to its modular design it can be used after k-Means has run its course, which means that it is completely compatible with other extensions of k-Means, such as k-Means++. Moreover, we were also able to show that KMN makes k-Means more robust in terms of too small or big k value.

For future work we have the goal to extend KMN towards a general noise extracting algorithm that can be applied as an addition for any clustering algorithm. For this goal it is necessary to abolish the voronoi cell structure of this algorithm, since that is inherent for k-Means, but not necessarily for other algorithms. Removing the voronoi cell structure might also lead to a more dynamic approach that would give us a greater flexibility.

References

1. Ahmed, M., Naser, A., *A novel approach for outlier detection and clustering improvement*, ICIEA, 2013.
2. Arthur, D., Vassilvitskii, S., *k-means++: the advantages of careful seeding*, SODA, 2007.
3. Avis, D., Fukuda, K., *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete & Computational Geometry, 1992.
4. Campos, G., et al, *On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study*, Data Min Knowl Disc, 2016.
5. Chawla S., Gionis A., *k-means–: A unified approach to clustering and outlier detection*, ICDM, 2013.
6. Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *A density-based algorithm for discovering clusters in large spatial databases with noise*, KDD, 1996.
7. Gan G., Kwok-Po Ng M., *k-means clustering with outlier removal*, Pattern Recognition Letters 90, 2017.
8. Hawkins D., *Identification of Outliers*, Chapman and Hall, 1980.
9. Johnson, N., Kotz, S., Balakrishnan, N. *Continuous Univariate Distributions*, Houghton Mifflin, 1994.
10. Lichman, M, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, 2013.
11. Mangasarian, O., Wolberg, W., *Cancer diagnosis via linear programming*, SIAM News, 1990.
12. MacQueen, J. B., *Some methods for classification and analysis of multivariate observations*, Berkeley Symposium on Math. Stat. and Prob., 1967. Computing Voronoi Adjacencies in High Dimensional
13. Mendez J., Lorenzo J., *Computing Voronoi Adjacencies in High Dimensional Spaces by Using Linear Programming*, Mathematical Methodologies in Pattern Recognition and Machine Learning, 2013.
14. Pelleg, D., Moore A. W., *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*, ICML, 2000.
15. Vinh, N. X., Bailey, J., *Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance*, JMLR, 2011.
16. Wangh, J.J., Dhillon, I., Gleich, D., *Non-exhaustive, Overlapping k-means*, SDM, 2015.