DipTransformation: Enhancing the Structure of a Dataset and thereby improving Clustering

Benjamin Schelling
Faculty of Computer Science
University of Vienna
Vienna, Austria

Claudia Plant
Faculty of Computer Science and
ds:UniVie
University of Vienna
Vienna, Austria

***Abstract*—A data set might have a well-defined structure, but this does not necessarily lead to good clustering results. If the structure is hidden in an unfavourable scaling, clustering will usually fail. The aim of this work is to present a technique which enhances the data set by re-scaling and transforming its features and thus emphasizing and accentuating its structure. If the structure is sufficiently clear, clustering algorithms will perform far better. To show that our algorithm works well, we have conducted extensive experiments on several real-world data sets, where we improve clustering not only for k-means, which is our main focus, but also for other standard clustering algorithms.**

***Index Terms*—Clustering, Dip-Test, Dataset-Transformation**

I. Introduction

The clustering of a data set is strongly dependent on the structure it contains. If there is hardly any structure or if the structure is well hidden, clustering will most likely fail because the boundaries between the clusters are hard to determine. A strong and clearly defined structure usually leads to significantly better clustering results. Accentuating the structure would therefore be useful for clustering, but to the best of our knowledge there are currently no methods that are capable of doing so. The most one can try is normalizing the data set in the hope that this defines the structure more clearly.

We present here DipTransformation[1], which is capable of accentuating structure and turning the data set into a more clusterable form.

Consider the data shown in Fig. 1 as a 3D scatterplot as running example. It is actually not a complicated data set, consisting of three stretched Gaussian distributed clusters, with different rotations and a third dimension of uniform distributed noise, which has about the same range as the clusters. The problem here is twofold: 1) The third dimension, which does not contain any structure, is given the same weight as the dimensions that contain the entire cluster structure. 2) The clusters, while not overlapping and with clear borders, are most unfavourably scaled.

The standard clustering algorithms are surprisingly bad on this data set. K-means scores here merely 0.01 in NMI[2], DBSCAN [8], Spectral Clustering [17] and SingleLink [19] also perform disappointingly. The best choice would be EM [6] with scores 0.43 in NMI.

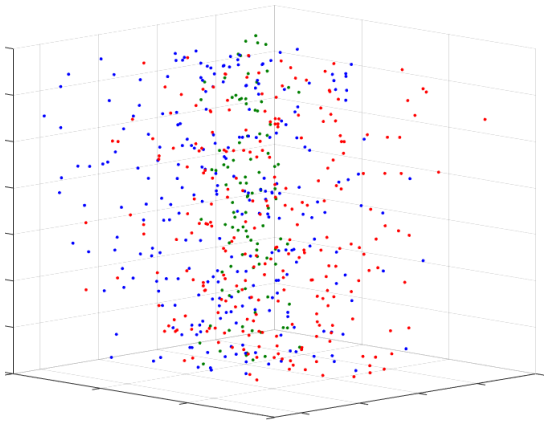


Fig. 1. Our running Example through this paper shown as a 3D Scatterplot.

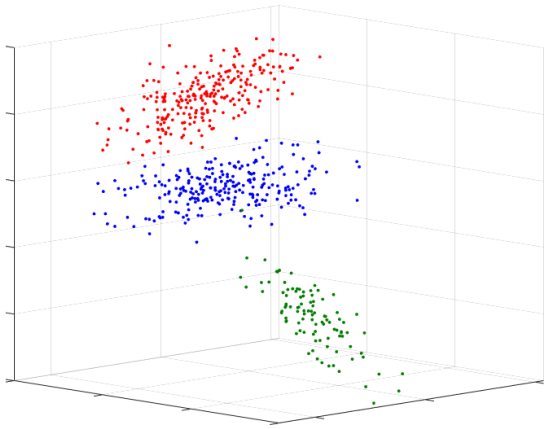


Fig. 2. Our running example after the DipTransformation in a 3D Scatterplot. It is now far easier to cluster.

Since the data set consists of a superfluous third dimension, we try dimensionality reducing techniques in the hope of adapting the dataset into a more clusterable form. The combination of clustering and dimensionality reduction is well established and might yield results here (see [13] for more details on this). However, neither PCA (0.03 in NMI) nor ICA (0.01 in NMI) lead to a data set that can be clustered

[1]Source code is found here: https://dm.cs.univie.ac.at/research/downloads/

[2]Measured in Normalized Mutual Information (NMI) [21]. NMI is scaled between 0.0 and 1.0, with 0.0 the worst possible score and 1.0 the best.

with k-means. The best choice would be t-SNE which scores ≈0.78, but has highly varying results. (All these techniques in combination with k-means with correct k.) The clusters are purely in the first two dimensions - so techniques like PROCLUS [1] and CLIQUE [2] which search for a clusters in axis-parallel subspaces could be successful, but they are not (0.21 and 0.71 in NMI, correct k for PROCLUS).

DipTransformation makes it possible to compensate for the unfortunate scaling of the features. We stated that the problem lies therein, that uniform/unimodal features (i.e. essentially structure-free features) receive the same degree of attention as such features that deviate from it. The basic assumption is that multimodal features are more interesting in regards to clustering since they contain more cluster structure. For k-means, this implicates that features with more structure should be larger scaled compared to features with barely any structure. A larger scaling would lead to a higher impact in k-means clustering due to the greater effect they have in computing the centres of the clusters and thus the way the clusters are determined. This requires a measure that evaluates the "interestingness" of a feature and therefore its scaling. We find this in the Dip-test [10] explained in Section II-A. The Dip-Test gives a appropriate measurement of the structure a feature has and thus the scaling it "deserves".

DipTransformation is capable of re-scaling and transforming our running example into a form that is almost perfectly clusterable with k-means. The clusters are better separated from each other and the structure of the data is more pronounced (see Figure 2). K-means now reaches an NMI of 0.97.

A. Contributions

This work presents an almost parameter-free method - the DipTransformation - that is able to improve the structure of a data set and thus allows k-means to cluster data sets better. The algorithm does not assume a special distribution for the clusters or data. It simply enhances structure and thereby improves clustering. Thus, it is not only a preparatory technique for k-means, it can also be used to improve clustering for various clustering techniques. DipTransformation is deterministic and requires no distance calculations. We extensively tested on real world data sets for a wide range of algorithms.

B. Related Work

The most common approach when a data set cannot be clustered well by any cluster algorithm is to create a new algorithm that can handle that data set. The reverse approach of adapting the data set to the algorithm is the much more unorthodox approach. It is usually only done in the simplest way, i.e. by normalizing a data set. In addition, there is the Z-transformation (sometimes referred to as Z-normalization), which is also relatively conventional, but is already applied far less often. Apart from these two methods, however, we are not aware of any approaches that attempt to adapt a data set with the aim of enhancing structure for improved clustering. Of course there are techniques that try to improve clustering, for instance k-means++ [3], which provides an initialization strategy for k-means that is often very successful, but transforming a data set is unusual. One might consider SynC [4] as a transformation technique, because it collapses clusters into single points using the principle of Synchronization.

Subspace clustering techniques such as the aforementioned PROCLUS and CLIQUE can be considered related work, since they intend to reduce dimensionality, i.e. adapt the data set by removing "unnecessary" information. The DipTransformation does not remove any information, but - as the analysis of the Running Example will show - it is very capable in dealing with such noise information. Of particular interest are FOSSCLU [9] and SubKMeans [16] which intend to reduce dimensionality with the goal of finding a subspace compatible with k-means.

We are also aware of progress in the field of Deep Learning, where techniques such as DEC [24] and DCN [22] are being developed, aimed at finding good subspaces using neural networks.

Spectral clustering takes a data set and transforms it into a distance matrix, computes its eigenvectors and applies (mostly) k-means to the data set. It is not necessary to use k-means, other partitioning algorithms can also very well be used. In this regard are spectral clustering techniques similar to the DipTransformation. They take the data set and try to transform it into a more clusterable form. One of the most well known is the fundamental technique by Ng, Jordan and Weiss [17]. We also use the popular Self-Tuning Spectral Clustering [25] as well as FUSE [23] as comparison methods due to them being state-of-the-art techniques.

DipTransformation uses the Dip-test for measuring structure and therefore one can consider all data mining-techniques that use the Dip-test as related. It was first used in data mining by DipMeans [11] with the goal of estimating the number of clusters for k-means. After that, we only found SkinnyDip [15] using the Dip-test. We conclude, that it is still a rather unknown tool, that has not yet found full recognition.

One must bear in mind while reading this, that DipTransformation is not a rival for all the mentioned techniques in the classical sense, but that it can be used as a supporting technique that eases the difficulty in the task they attempt. We will show in the experimental section (see Section IV) that they can all benefit from DipTransformation.

II. The Algorithm

A. The Dip-Test

To understand how the algorithm works, we must first go into detail about the Dip-test.

The Dip-Test was created by Hartigan & Hartigan in the 1980s as a measure of how much a sample deviates from unimodality. Unimodality is defined here as a distribution that is convex until it reaches its maximum and concave thereafter.

The test starts with sorting the sample and then creating the Empirical Cumulative Distribution Function (ECDF). This can be seen in Fig. 3. The histogram shows 4 clusters (A, B, C and D), which can be clearly identified in the ECDF to its right. The Dip-Test only requires the ECDF; the histogram is

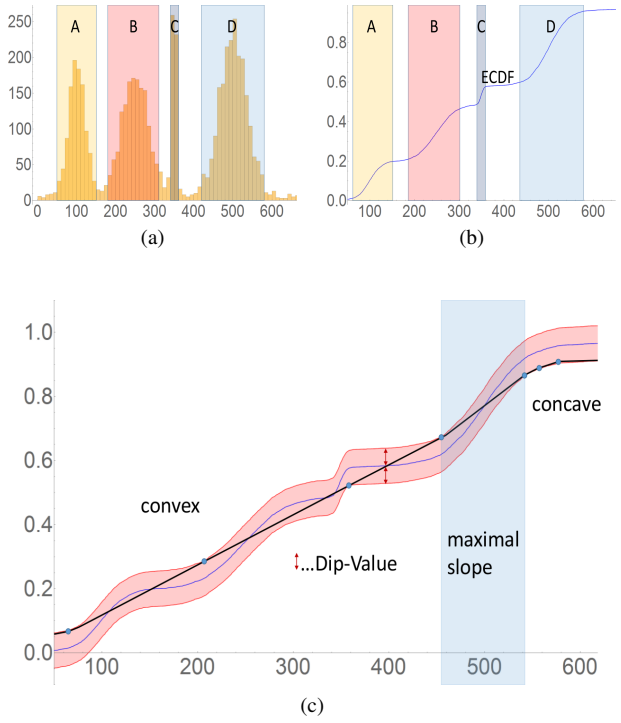


Fig. 3. A histogram and the resulting ECDF (empirical cumulative distribution function). The dip-test uses the ECDF to find out how much it deviates from a unimodal distribution, i.e. how big the offset is to fit a unimodal distribution. The offset is the dip-value.

only for visual clarity. It therefore has no bin-width parameter. In fact, it has no parameter at all.

The Dip-test measures the extent to which the ECDF deviates from unimodality. It computes how much the ECDF has to be offset, so that it can fit a unimodal distribution. This can be seen in Fig. 3(c). The ECDF has been shifted vertically by a certain value (the dip value), and ECDF+dip and ECDF-dip is plotted there. This offset is large enough so that a line can be drawn in between ECDF+dip and ECDF-dip, which is first convex and then concave. This line is representative of the closest possible unimodal distribution. The dip-value (or "dip") shows how much the ECDF is off from such a unimodal distribution.

The Dip-Test also gives a second value, a probability of how likely a sample is unimodal, as well as the interval of the highest slope, but we only need the offset/dip-value. The dip-value is always in the interval $(0, 0.25]$, hence it is always positive.

The Dip-Test has a runtime of $\mathcal{O}(n)$, but since its input must be sorted to create the ECDF, the effective runtime for this part of the technique is $\mathcal{O}(n \log n)$. Further details about the Dip-Test can be found in [10].

B. Applying the Dip-Test

By means of the Dip-Test, we obtain a value, the dip statistic, which provides a measure of the structure of a dimension and thus, as explained in the introduction, a measure of the "relevance" of the dimension. The more relevant a dimension is, the larger will it be scaled (in relation to the other dimensions) and the greater its influence on the clustering result from k-means.

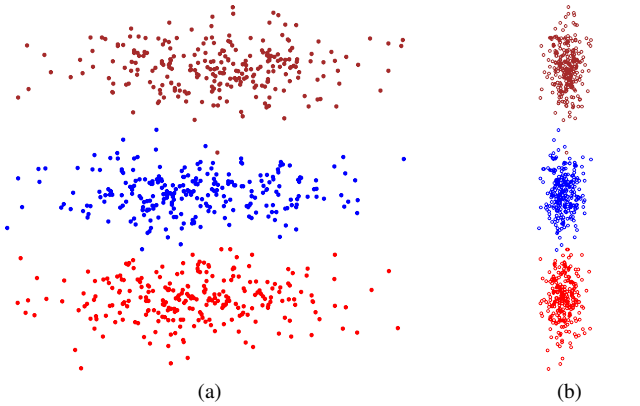


Fig. 4. A simple data set before (a) and after (b) scaling with the Dip-Values.

Let us consider this approach with Fig. 4. The dip values of the individual dimensions are 0.009 for the projection into the horizontal dimension and 0.063 for the projection in the vertical one. We rescale the horizontal axis in the interval [0,0.009] and the vertical one in [0,0.063] and get the dataset represented in Fig. 4b.

The changed distances make this data set now easily clusterable by k-means. The dimension containing the structure is now much more pronounced and accordingly more influential for k-means. The improvement of the clustering result is best described using the NMI value, which increases from an average value of 0.55 for the unscaled data set to 0.98 for the rescaled data set (100 random initialisations each). The only error and the reason why an NMI-value of 1.0 is not reached is due to some edge-data points that have been falsely assigned, but could not reasonably be expected to be correctly clustered.

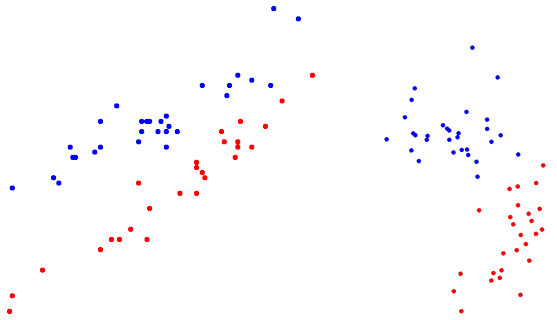
This (somewhat trivial) example shows how important it is to enhance the structure of a data set. The horizontal axis in which the data set has barely, if any, structure is reduced to a very small range and the vertical axis, where the clusters and structure are located, is now the relevant dimension that determines the result of k-means.

C. Lopsided Cluster

We do not generally assume axis-parallel stretched cluster. Let us therefore look at the Whiteside data set depicted in Fig. 5a. The Whiteside data set is a real-world data set.

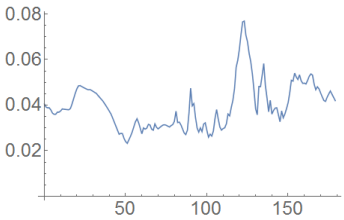
The clusters are obviously not axis-parallel. We have the same situation here as in the simple data set shown in Fig. 4, as in that k-means fares rather badly. To be precise the NMI-value is 0.006. Scaling the axes with the Dip-Test values is ineffective; it improves clustering only minimally.

We see in Fig. 5c that the dip value changes greatly depending on the angle at which the dip value is measured. If we rotate the Whiteside data set by the angle at which we find the maximal dip value, the clusters are almost axis-parallel. Now scaling the axes with their dip value leads to



(a) The Whiteside-data set.

(b) Applying the maximal Dip.



(c) The Dip-value of the Whiteside data set when rotated.

Fig. 5. The Whiteside-data set and the change of the Dip-value of an axis when the data set is rotated (0 to 180 degrees) as well as the Whiteside-data set, when it is rescaled at the maximal dip.

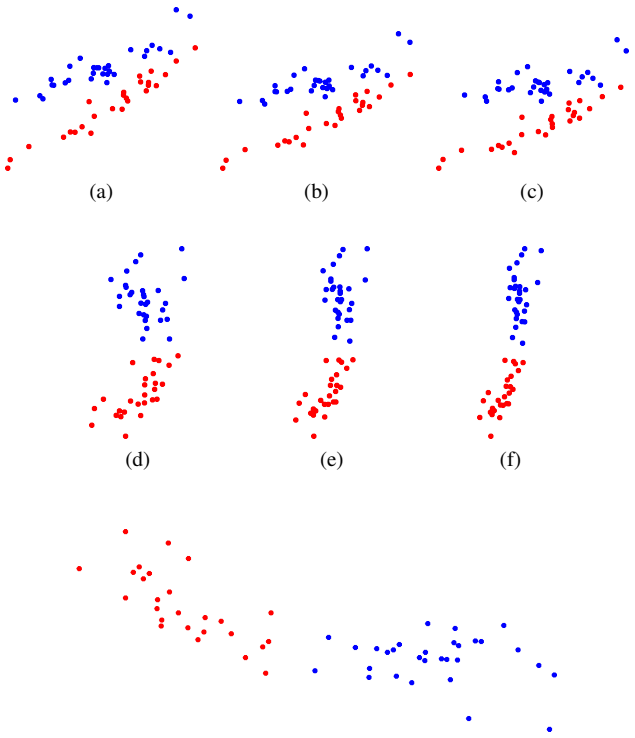


Fig. 6. Steps in the DipTransformation of the Whiteside data set and the final data set. There seem to be big leaps in the transformation; this is due to the iterations, when the maximal new dip value is smaller compared to the $MaxDip$, and thus no scaling takes place. Until a new $MaxDip$ is found the data set is rotated quite a lot and the changes seem extensive.

the transformed data set shown in Fig. 5b. This data set can be clustered considerably better by k-means. In fact, we get an average NMI of 0.92. This is a massive improvement over the previous NMI of 0.006, but it is possible to improve even further, as we will see.

Determining the angle of the maximal Dip-value is not straightforward. Of course, one could use the brute-force approach of simply testing as many angles as possible, but this will prove impractical at the latest when the data set is of higher dimensionality. A too simple search algorithm will also not lead to a satisfactory result, since the data, as we have seen in Fig. 5c, has more than a few local optima. Hence, we do not try to find only a single maximal dip-angle, but scale the data set along along several high dip angles. Basically, the algorithm does not restrict itself to only re-scaling the data set along the maximal dip-value, but rescales the data set along multiple such high dip values. This converts the data set into a more clusterable form. The algorithm scales the data set in various instances, so that the structure of the clusters become more clearly defined.

We start with two dip-values, D_1 and D_2, the two dip values along the axes. We calculate the ratio between those values $r = Max(\frac{D_1}{D_2}, \frac{D_2}{D_1})$. If r is high, then there is a good chance that we have hit a good dip-value. Then we rotate the data set in clockwise direction by an angle of $\frac{1}{r} * c$, with c as the rotation speed parameter. This ensures that we rotate the data set only by a small angle if chances are good that we are close to a high dip value and by a bigger angle if the dip values are similar, i.e. chances are that no high dip value is close. The rotation speed parameter c can be freely selected. The larger c is selected, the shorter the runtime, but a smaller c of course makes it more likely to find high dip-values. (Throughout the paper, the rotation speed parameter c is set to 5. Its effects are further explored in Section II-E.) The algorithm remembers the maximal overall dip value (we refer to it as $MaxDip$) and every time it finds a new maximal overall dip value, the axes are scaled with their respective dip values. The total degrees the data set has been rotated is remembered and after 360° the algorithm stops. For the Whiteside-data set with $c = 5$ leads this to the transformation displayed in Fig.6.

At the beginning of the transformation, the changes are comparatively small. This is because the dip values of the axes are not very different and therefore scaling the axes only leads to limited changes. $MaxDip$ is updated, whenever a new Maximum is found. The following iterations is the maximal dip value of the axes not greater than $MaxDip$ and hence no scaling takes place. However, after several rotations, the maximal dip value of the axes is greater than $MaxDip$ and the data set is scaled again. This happens between Fig. 6(c) and Fig. 6(d). Because the data set is rotated over several iterations, it has been rotated by a rather large angle and the following scaling makes the data set look quite different. The algorithm remembers the new $MaxDip$. The data set continues to rotate, but due to the way the algorithm selects the rotation angle, it is only rotated by a small angle, which is advantageous because finds high dip values. In the following iterations, the data set

is again scaled several times. In the second row of Fig. 6, it can be seen that the data set does not change drastically, but becomes more compact and the clusters are defined more clearly with each iteration. Fig. 6(g) shows the final state of the data set. Between Fig. 6(f) and Fig. 6(g) is again a stretch where the data set is rotated but no new $MaxDip$ is found.

This transformation of the Whiteside-data is very easy to cluster for k-means. We get an average NMI-value of 1.0 (in 100 iterations), which means that the data set is perfectly clustered. This result is not specific to a value of $c = 5$, but can also be reached by e.g. $c = 9, 8, 7, 6, ...$ However, the transformed data set may look slightly different for a different value of c.

This transformation is easier to cluster, when comparing to the original Whiteside-data set shown in Fig. 5, but also when comparing to the transformation along the maximal Dip-value angle shown in Fig. 5b. One could have expected these transformations to be more similar, if not identical, but that is not the case. The transformation here is not along an orthogonal basis.

Scaling along axes leads to a basis transformation that stretches the basis vectors, but leaves orthogonality intact. Applying the transformation method sketched above leads also to a change in basis vectors, which no longer implies that two previously normal (i.e. perpendicular) vectors are normal to each other afterwards.

Theorem 1: The DipTransformation DT is a linear operator. More precisely, it is a basis-transformation.

Proof. Every rotation in $\mathbb{R}^n$ can be expressed as a matrix R. Scaling a data set in $\mathbb{R}^n$ simply means applying a diagonal matrix S with the scaling-parameters in the main diagonal. Hence, applying the DipTransformation on a data set is equivalent with applying the Rotation- and Diagonal-matrices R_1, S_1, R_2, S_2, R_3, S_3,.... Thus, the DipTransformation DT is the product of matrices, which is again a matrix. A matrix is a linear operator, hence the DipTransformation is a linear Operator.
A rotation is an orthogonal matrix with determinant 1, a scaling matrix has the determinant $c_1 \cdots c_n$, with c_i the entries in the diagonal. Since the Dip-Test values c_i can never be zero, the determinant of the scaling matrix is non-zero. The determinant has the property $Det(A \cdot B) = Det(A) \cdot Det(B)$, hence the determinant of the DipTransformation is $Det(DT) = Det(R_1) \cdot Det(S_1) \cdots Det(R_l) \cdot Det(S_l) = 1 \cdot (c_{11} \cdots c_{n1}) \cdots 1 \cdot (c_{1l} \cdots c_{nl}) \neq 0$. Thus is DT a matrix with non-zero determinant, i.e. a basis-transformation. ■

The focus of DipTransformation is on k-means, but we see from Fig. 6 that other techniques might also benefit from this approach. We will explore in Section IV how other techniques are influenced by this (and other) transformed data set(s).

D. More than 2 Dimensions

The algorithm for a 2-dimensional data set was explained in detail, because it forms the basis for data sets with more than two dimensions. There are several ways to adapt this approach; the one that seems to work best is now explained:

The main difference is that there are more than two directions, the data set can be rotated in. It would seem logical to rotate the data set in all directions at once following the angle-computation as before, but there is a problem involved with that: Rotations are not commutative. That means, it makes a difference in which order the rotations are executed. Finding only one non-axes parallel angle in which the data set is rotated, is anything but straightforward, since all we have are the dip values of the axes, that we can use to compute axes-parallel rotation angles. Nevertheless, the algorithm simply executes one rotation after another. However, since every rotation changes the data set (slightly), it is better to recalculate the dip values. This could be omitted to save runtime, but the recalculated dip values are more precise and this in turn improves the transformation, especially with larger rotation speed parameters c. One might expect that changing the order in which the rotations are executed might improve the transformation, but this is not the case, according to the experiments we conducted. We also tried only executing the rotation with the highest/lowest dip value, but this even seems to impair the transformation. Through all rotations the algorithms remembers the maximal dip value found as $MaxDip$, just as before. Whenever the rotated data set has a dip value larger as $MaxDip$, the data set is scaled and $MaxDip$ is updated. The rotation angles are calculated in the same way as for a 2-dimensional data set. Furthermore, the executed rotations are rotations in the plane given by two axes-vectors.

One has to keep in mind that in higher dimensional data set, the algorithm has a larger area to search for high dip values. It is only a "half-circle" or 180° that needs to be traversed for a 2-dimensional data set. (In the interest of precision, however, the algorithm looks over the full 360°.) For a $d-$dimensional data set, it would be half of a $d-$dimensional sphere. To compensate for this, the algorithm assumes that d·360° have to be traversed. This range ascertains that all maxima can (theoretically) be found. Furthermore, it is not necessary to find the maximal dip values exactly; being close enough is sufficient to assure a good transformation.

E. The Rotationspeed Parameter c

The rotation speed parameter c has been explained in Section II-C and we now want to analyse its effect on the DipTransformation. Fig. 7 shows how NMI (for k-means) changes with different values of c and the effect is not very pronounced. The data sets depicted were chosen, because their values do not overlap, but the effect is rather similar for all data sets examined in Section IV.

There is a slight tendency for the clustering results to lose quality at higher values of c (best visible for the Whiteside data set), but it is not very pronounced. For an unknown data

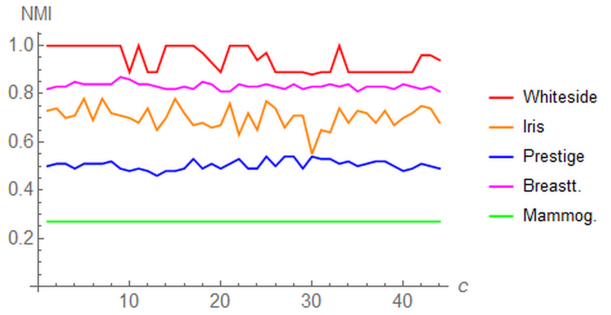


Fig. 7. The NMI-value for k-means with correct values for k vs the rotation speed parameter c for five real world-data sets.

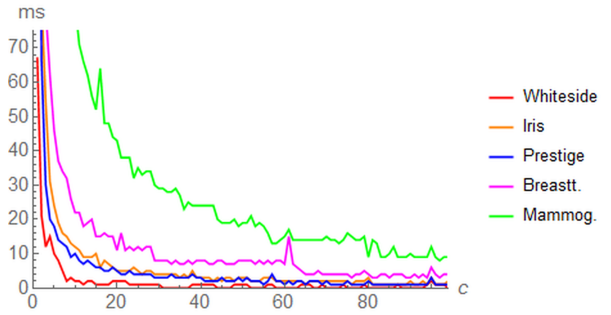


Fig. 8. The time DipTransformation takes in milliseconds depending on the value of the rotation speed parameter c for five real world-data sets.

set is a smaller value for c nevertheless more recommended. DipTransformation is dependent on finding the high dip-values and that is simply more likely for smaller rotation speed. If runtime is the primary factor (for example, for very large data sets), then a larger value for c might be more recommendable. Fig. 8 shows how the runtime is linked to the parameter c and how it decreases for smaller c. There is a (small) trade off between clustering quality and runtime, but as a general rule a high value of c seems not too detrimental. For this work, however, we stick to the fixed value $c = 5$.

III. Runtime and Pseudocode

Following the structure of the DipTransformation algorithm (outlined in Algorithm 1) we can make a runtime estimation: Scaling of a data set as well as rotating a data set has a runtime of $\mathcal{O}(n)$; Computing the Dip-Values is in the order of $\mathcal{O}(n \log n)$, since the values have to be sorted. There are two For-loops over d, where d stands for the number of dimensions. If the number of iterations in the while-loop is l, then the runtime can be estimated as:

$$\mathcal{O}(n) + \mathcal{O}(n \log n) + l \cdot d \cdot d \cdot \big(\mathcal{O}(n) + \mathcal{O}(n \log(n)) + \mathcal{O}(n)\big) \approx \mathcal{O}\big(l \cdot d^2 \cdot n \log(n)\big)$$

IV. Experiments

Persuading someone that a data set is easy to cluster, if the data set is more than two-dimensional, is difficult. The goal of the DipTransformation though is to ensure that a data set becomes easier to cluster. This work will of course

Algorithm 1 DipTransformation

Require: Data D, Rotationspeed c

procedure DipTransf(D, c)
  $Degree \leftarrow 0$
  Compute $DipValues$
  Scale(D,DipValues)
  $MaxDip \leftarrow Max(DipValues)$
  while $Degree < dim * 180°$ **do**
    for i = 1,...,dim **do**
      for j = i+1,...,dim **do**
        $a \leftarrow Max(Dip(i)/Dip(j), Dip(j)/Dip(i))$
        Turn $D(i, j)$ by angle c/a
        $Degree \leftarrow Degree + c/a$
        Compute $DipValues$
        if $Max(DipValues) > MaxDip$ **then**
          Scale(D,DipValues)
          $MaxDip \leftarrow Max(DipValues)$
        end if
      end for
    end for
  end while
  return D
end procedure

show with NMI values of experiments on real-world data set that DipTransformation is capable of doing that, but we would also like to show that with plots. Fig. 9 shows pairwise plots of the "Banknote Authentication" data set from the UCI-Repository [7]. This data set was chosen because it has a small dimensionality of four, so that all pairwise plots can be shown. Fig. 9 illustrates the difficulty involved with clustering this data set. The clusters are not clearly separated and often overlap, so it is not suited for k-means. In numerical values can this be expressed as an NMI-value 0.03 for k-means with the correct value for k. After the DipTransformation (shown in Fig. 10) the data set is much better structured and the clusters are well separated. K-means can now identify the clusters rather well. In fact, the NMI-value for k-means with the correct value for k is now 0.68.

A. Synthetic Data

The first analysed data set is the synthetic data set given in Fig. 1. This is a data set that - as we have seen - is quite difficult to cluster; k-means fares extremely bad and scores no higher than 0.01 in NMI. Other algorithms are often only marginally better. Table I shows the NMI-results. Most of them are not impressive, with 11 of the 20 algorithms scoring below 0.10. Applying the DipTransformation onto the data set leads to a massively enhanced data set with clearly stronger defined structure. The three clusters that were before stretched and scaled quite unfavourable for clustering are now well separated and compact. Clustering of this data set is far easier and the results shown in Table I demonstrate this. All of the used algorithms improve due to the DipTransformation - on average

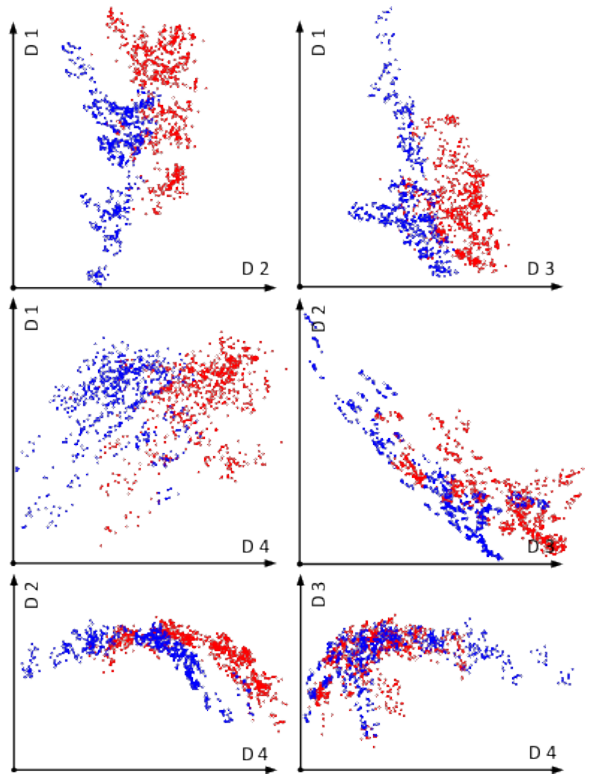


Fig. 9. Pairwise Plot of the Banknote-Authentication data set before DipTransformation. The dimensions are given as axes-label.

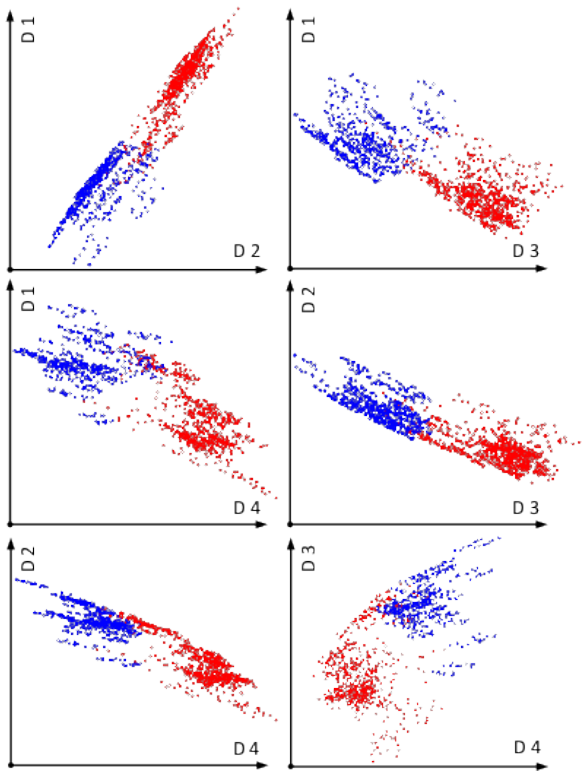


Fig. 10. Pairwise Plot of the Banknote-Authentication data set after the DipTransformation. The clusters are visibly better separated from each other.

TABLE I
CLUSTERING OF THE RUNNING EXAMPLE BEFORE AND AFTER DIPTRANSFORMATION. THE AVERAGE IMPROVEMENT IN NMI IS 0.636.

Running Example	before	after
k-means	0.01	0.97
k-means++	0.01	0.98
DipMeans	0.00	0.98
SkinnyDip	0.00	0.98
Spectral Clust.	0.36	0.97
STSC	0.00	0.99
FUSE	0.06	0.75
DBSCAN	0.23	0.95
SingleLink	0.01	0.96
EM	0.43	0.50
SubKMeans	0.08	0.63
FossClu	0.60	0.78
SynC	0.05	0.95
PCA	0.03	0.63
ICA	0.01	0.98
t-SNE	0.78	0.80
PROCLUS	0.23	0.92
CLIQUE	0.71	0.98
DEC	0.38	0.53
DCN	0.12	0.58

0.636 in NMI. After the DipTransformation are 12 of the 20 algorithms better than 0.90.

FossClu [9] and SubKMeans [16] try to find an optimal subspace for clustering while transforming the data set themselves. These transformation attempts are also more successful after DipTransformation has transformed the data set. Even other transformations profit from the DipTransformation.

B. Real-World Data sets

We have tested the DipTransformation extensively on 10 real world data sets, which differ greatly in dimensionality, number of data points and number of clusters. The ultimate goal of the DipTransformation is to enhance the structure of a data set and thus to improve clustering. To show that this goal can be achieved, we have transformed these data sets and applied the basic k-means algorithm on them. The results can be seen in Table II. The difference in clustering quality is obvious. While k-means on the original data sets usually fares somewhat lacking and other algorithms like Spectral Clustering are often the better choices, it does perform extremely well on the transformed data sets. Transforming the data set has enhanced its structure so that k-means can cluster the data sets better than the compared methods. In 9 of the 10 cases k-means clusters better (or no worse) than the other methods. Only in one case does it take second place with a deficit of 0.02.

We chose the algorithms we found to be most relevant as comparison methods here. This included the data set-transformation techniques of normalizing and Z-Transformation, the standard data mining algorithms DBSCAN [8], EM [6] and SingleLink [19], DipMeans [11] and SkinnyDip [15] as techniques based on the Dip-Test, SubKMeans [16] and FossClu [9] as the most similar Subspace-clustering-techniques, the aforementioned Spectral Clustering-methods, SynC [4], as well as PCA and t-SNE in combination

TABLE II
EXPERIMENTAL RESULTS. ALL NON-DETERMINISTIC RESULTS HAVE BEEN REPEATED 100 TIMES AND THE AVERAGE IS GIVEN. THE CORRECT VALUE FOR NUMBER OF CLUSTERS IS ALWAYS GIVEN.

Data set	Whiteside	Skinsegmen.	Banknote	Iris	Prestige	Userknow.	Mammographic	Seeds	Breast Tissue	Leaf
# of data points	56	245057	1375	150	98	258	830	210	106	340
# of dimensions	2	3	4	4	5	5	5	7	9	14
# of clusters	2	2	2	3	3	4	2	3	6	30
DipTransformation	**1.00**	**0.32**	**0.68**	**0.84**	**0.68**	**0.53**	0.27	**0.78**	**0.51**	**0.69**
k-means	0.01	0.02	0.03	0.70	0.51	0.26	0.11	0.70	0.32	0.65
Normalized	0.01	0.02	0.02	0.68	0.50	0.28	0.27	0.67	0.49	**0.69**
Z-Transformation	0.01	0.02	0.02	0.68	0.51	0.28	0.28	0.67	0.49	**0.69**
k-means++	0.01	**0.32**	0.03	0.75	0.56	0.22	0.11	0.71	0.18	0.57
DipMeans	0.00	—	0.25	0.55	0.45	0.00	0.00	0.00	0.00	0.45
SkinnyDip	**1.00**	—	0.34	0.55	0.55	0.30	0.00	0.53	0.26	0.00
Spectral Clust.	0.06	—	0.03	0.60	0.60	0.29	0.09	0.34	0.45	**0.69**
STSC	0.35	—	0.26	0.39	0.53	0.03	—	0.66	0.31	0.09
FUSE	0.09	—	0.03	0.46	0.06	0.02	0.06	0.15	0.11	0.31
DBSCAN	0.27	—	0.46	0.62	0.54	0.27	0.14	0.50	0.41	0.59
SingleLink	0.11	—	0.03	0.61	0.08	0.05	0.00	0.05	0.27	0.35
EM	**1.00**	0.23	0.01	0.58	0.28	0.44	0.01	0.63	0.37	0.25
SubKMeans	0.01	0.01	0.01	0.66	0.56	0.22	**0.29**	0.73	0.45	0.66
FossClu	—	0.27	0.44	0.75	0.48	0.50	0.08	0.50	0.32	0.34
SynC	0.12	0.13	0.14	0.58	0.52	0.13	0.24	0.48	0.29	0.27
t-SNE + k-means	0.02	—	0.64	0.31	0.02	0.06	0.11	0.16	0.08	0.35
PCA + k-means	0.01	0.01	0.01	0.64	0.56	0.21	0.26	0.74	0.49	**0.69**

TABLE III
K-MEANS++ BEFORE AND AFTER THE DIPTRANSFORMATION AS WELL AS K-MEANS AFTER THE DIPTRANSFORMATION. NUMBER OF CLUSTERS IS GIVEN.

Data set	Whiteside	Skinsegmen.	Banknote	Iris	Prestige	Userknow.	Mammographic	Seeds	Breast Tissue	Leaf
k-means before	0.01	0.02	0.03	0.70	0.51	0.26	0.11	0.70	0.32	0.65
k-means++ before	0.01	0.32	0.03	0.75	0.56	0.22	0.11	0.71	0.18	0.57
k-means after	**1.00**	0.32	0.68	0.84	**0.68**	0.53	**0.27**	**0.78**	**0.51**	0.69
k-means++ after	**1.00**	**0.44**	**0.69**	**0.86**	**0.68**	**0.64**	**0.27**	0.76	0.49	**0.70**

with k-means. For PCA and t-SNE we decided not to reduce the dimensionality, because there is no straightforward answer on how far one should reduce the dimensionality and because DipTransformation also does not reduce dimensionality.

a) Parameters and Determinism: Algorithms like DBSCAN always raise the question of how to set the parameters. To compare the DBSCAN results fairly, we decided to make the parameters dependent on the average pairwise Euclidean distance of data points. Let us call it e. We tested all combinations of distances in $\{0.05 \cdot e, 0.1 \cdot e, 0.2 \cdot e, 0.3 \cdot e, 0.4 \cdot e, 0.6 \cdot e, 0.8 \cdot e, e\}$ and MinPts in $\{1, 2, 3, 5, 10, 50\}$. Only the best NMI result is reported.

All techniques that require the number of clusters as a parameter have been given the correct number of clusters k. The only exception is SingleLink where all values in the interval $[k, 2k]$ have been tested and only the best result is reported. This decision is due to the characteristic of SingleLink to declare single data points or small subsets of a cluster as clusters.

Non-deterministic algorithms such as k-means have been iterated 100 times to reduce random effects and provide robust results.

b) Skinsegmentation: The Skinsegmentation-data set is a somewhat difficult data set simply due to its size of roughly a quarter of a million data points. For many of the provided implementations was the size too large and the execution failed. This also applies to some of the standard methods like SingleLink, DBSCAN and Spectral Clustering. These were tested on more than one implementation on different platforms, but would not run through anyway.

c) Spectral Clustering: If this paper refers to Spectral Clustering as an algorithm and not the class of algorithms, then the classical algorithm by Ng, Jordan and Weiss [17] is meant.

Besides these considerations it is most noticeable that k-means++ leads to the same increase in NMI as the DipTransformation. However, K-means++ and DipTransformation are by no means mutually exclusive and can be used together. This in fact leads to an even better performance on the Skinsegmentation data set. While they separately reach a level of 0.32 in NMI, they manage 0.44 in combined form.

C. K-means++ and DipTransformation

As mentioned, k-means++ and DipTransformation are not mutually exclusive. We tested on all the data sets used in the experiments whether k-means++ fared better before or after the DipTransformation. The results are shown in Table III. Following these, we can say that k-means++ is a bit of a

double-edged sword on the original data sets. On some of them (Skinsegmentation, Iris) k-means++ is clearly better than k-means; on some of them (Breast Tissue, Leaf) it is the other way round. After the DipTransformation, the situation is far more beneficial for k-means++. Usually, there is only a small difference between k-means and k-means++ ($\leq$ 0.02), indicating that there might be fewer local optima, compared to the original data set. The only times when k-means and k-means++ do differ (Skinsegmentation, Userknowledge) is when k-means++ performs quite a bit better than k-means alone.

DipTransformation can be used together with all types of support techniques for k-means (or other clustering algorithms). For example, X-means [18] can be used to find the number of clusters, k-means-- [5] to find outliers, k-means++ to find an initialization, SubKMeans to find a subspace and all this in combination with the DipTransformation.

D. DipTransformation and Clustering Algorithms besides K-means

DipTransformation was developed with a focus on k-means, but as we have stated throughout our work, DipTransformation only enhances the structure; it does not adapt the data set so that it only fits k-means and therefore other algorithms can also benefit from it. We have seen the transformation of the Whiteside as well as the Banknote-Authentication data set in Fig. 6 respectively Fig. 10 and both of these do seem easier to cluster for various algorithms. We have taken 5 of the data sets used in the real world data sets experiments and clustered their DipTransformations with 4 standard data mining algorithms, i.e EM, DBSCAN, SingleLink and Spectral Clustering. The results can be seen in Table IV. We chose the standard algorithms because they are well-established in the community, which makes the results all the more credible. For the sake of completeness is k-means also included here. In two cases do we see a tiny decrease in clustering quality of 0.01 in NMI. In two more cases does the quality not change at all. In the other 21 cases does the quality increase - in some cases substantially. On average, counting all cases, the quality increases by 0.223 in NMI.

Combined with Fig. 6 and 10, this should be a very convincing argument that DipTransformation can play an important role in Clustering as a support technique applied to the data set before clustering.

E. Runtime Comparisons

The runtime was estimated to have a $\mathcal{O}(n \log n)$-dependency on the number of data points n. Synthetic data sets ranging from 1.000 to 15.000 data points were created to test for this dependency and to compare with other algorithms. The runtime is plotted in Fig. 11. It is not immediately apparent, but $\mathcal{O}(n \log n)$ is a very good estimate for the runtime. We also see that DipTransformation performs quite well compared to the other algorithms tested there. DipTransformation is faster for all data sets. To be fair is in this test also the runtime of k-means included in the measured time for DipTransformation,

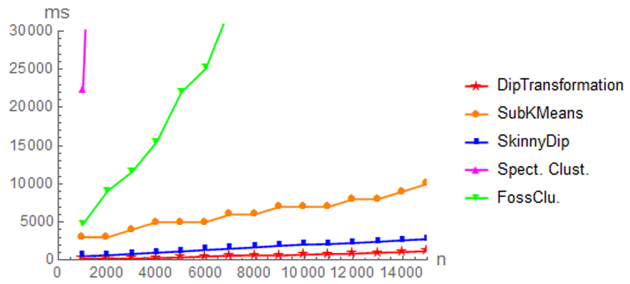


Fig. 11. Runtime relative to the data set size n.

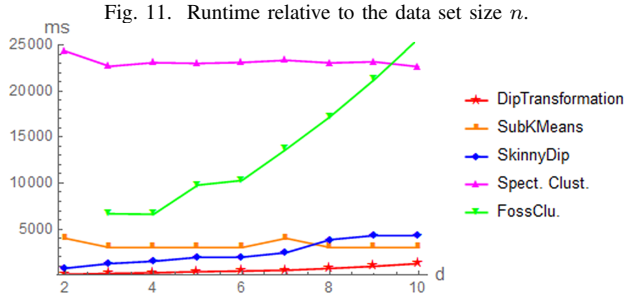


Fig. 12. Runtime relative to the dimensionality of the data set d.

since the other methods cluster data, which DipTransformation does not do by itself.

Besides the size of the data set is also the influence of the dimensionality of the data set on the runtime essential. This is shown in Fig. 12. We created 9 data sets ranging in dimensionality from 2 to 10 with 1.000 data points each. Here is DipTransformation (+k-means) again faster than all other methods, but we do see in the behaviour of the measured time, that other methods like Spectral Clustering are less affected by the dimensionality. The estimation of an $\mathcal{O}(d^2)$ dependency on dimensionality is again a very good one, so the conjecture that at some point DipTransformation will need more time than e.g. Spectral Clustering is a likely one. However, if one extrapolates from the curves, it seems as if that would happen at a rather high dimensionality.

The algorithms we found to be most similar to DipTransformation were chosen here. They were tested in Java (DipTransformation and FossClu), Scala (SubKMeans) and R (Spectral Clutering and SkinnyDip) on an Intel Xeon E5 with 16Gb RAM.

V. Limitations and Outlook

DipTransformation is a very powerful technique for improving the structure and emphasizing clusters, but there are certain limits. For example, if two clusters overlap or interlock, DipTransformation would by design not be able to separate them. DipTransformation stretches and scales the data, therefore all cluster which do not have a hyperplane in-between them cannot be separated. The same applies to clusters that contain more than one mode. The dip test is designed to work with unimodal distributions. Multimodal clusters can prevent DipTransformation from working properly.

TABLE IV
VARIOUS CLUSTERING ALGORITHMS BEFORE AND AFTER DIPTRANSFORMATION. THE CORRECT VALUE FOR k IS ALWAYS GIVEN. ON AVERAGE THE CLUSTERING IMPROVES BY 0.223 (MEASURED IN NMI).

Data set		Whiteside	Iris	Prestige	Mammographic	Breast Tissue
EM	before	**1.00**	0.58	0.28	**0.01**	0.37
	after	**1.00**	**0.90**	**0.63**	0.00	**0.45**
DBSCAN	before	0.27	**0.62**	0.54	0.14	0.41
	after	**0.72**	0.61	**0.60**	**0.15**	**0.45**
SingleLink	before	0.11	**0.61**	0.08	0.00	0.27
	after	**0.82**	**0.61**	**0.54**	**0.16**	**0.35**
Spectral Clustering	before	0.06	0.60	0.60	0.09	0.45
	after	**1.00**	**0.65**	**0.65**	**0.26**	**0.50**
k-means	before	0.01	0.70	0.51	0.11	0.32
	after	**1.00**	**0.84**	**0.68**	**0.27**	**0.51**

In terms of runtime, the main constraint we encountered was the $\mathcal{O}(d^2)$-dependency on the dimensionality of a data set. For very high dimensional data sets it might be useful to combine dip transform with a dimensionality reduction algorithm at the current state.

We do intend to implement a dimensionality-reducing feature into DipTransformation. The dip-test provides a probability estimate of the unimodality of a feature. For the running example, the dip test gave a probability of 100% for the third dimension to be unimodal. This is a correct estimate as the third dimension was constructed as uniformly distributed noise. If we assume that a unimodally distributed characteristic is essentially not of great interest, then we could eliminate this dimension and reduce the running example to a two-dimensional data set. This two-dimensional data set would then be treated as explained in this paper. At some point, however, it might happen that the dip test finds another unimodal feature to be and the data set can be further reduced.

According to this roadmap, the DipTransformation could be converted into a technique that improves the structure of a data set while reducing dimensionality. We intend to do this in the near future.

VI. CONCLUSION

In conclusion, we can say that we have achieved our goal of creating a technique that can improve the structure of a data set and thus its clustering. We have shown that this statement is true by testing it extensively on various data sets.

For k-means, which was the main focus, this is now particularly clear. On the tested data sets, k-means was usually a sub-ideal choice and other algorithms were clearly better. After the DipTransformation was k-means the best-performing algorithm on all but one data set.

We have also shown that DipTransformation is compatible with other algorithms and also improves their clustering results. DipTransformation can therefore be used as a pre-clustering step, that enhances the data set, and the clustering algorithm can be chosen as the user likes best.

REFERENCES

[1] Aggarwal, C. C., Procopiuc, C., *Fast Algorithms for Projected Clustering*, SIGMOD, 1999.

[2] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P., *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*, SIGMOD, 1999.

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

[4] Böhm, C., Plant, C., Shao, J., Yang, Q., *Clustering by synchronization*, KDD, 2010.

[5] Chawla, S., Gionis, A., *k-means--: A unified approach to clustering and outlier detection*, ICDM, 2013.

[6] Dempster, A. P., Laird, N. M., Rubin, D. B., *Maximum-Likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, 1977.

[7] Dua, D., Karra Taniskidou, E., *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, 2018.

[8] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *A density-based algorithm for discovering clusters in large spatial databases with noise*, KDD, 1996.

[9] Goebl, S., He, X., Plant, C., Böhm, C., *Finding the Optimal Subspace for Clustering*, ICDM, 2014.

[10] Hartigan, J. A., Hartigan, P. M., *The Dip Test of Unimodality*, The Annals of Statistics, 1985.

[11] Kalogeratos, A.,Likas, A., *Dip-means: an incremental clustering method for estimating the number of clusters*, NIPS, 2012.

[12] Krause, A., Liebscher, V., *Multimodal projection pursuit using the dip statistic*, Preprint-Reihe Mathematik, 2005.

[13] Kriegel, H. P., Kröger, P., Zimek, A., *Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering*, TKDD, 2009.

[14] MacQueen, J. B., *Some methods for classification and analysis of multivariate observations*, Berkeley Symposium on Math. Stat. and Prob., 1967.

[15] Maurus, S., Plant, C., *Skinny-dip: Clustering in a Sea of Noise*, KDD, 2016.

[16] Mautz, D., Ye, W., Plant, C., Böhm, C., *Towards an Optimal Subspace for K-means*, KDD, 2017.

[17] Ng, A., Jordan, M., Weiss, Y., *On spectral clustering: Analysis and an algorithm*, NIPS, 2002.

[18] Pelleg, D., Moore A. W., *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*, ICML, 2000.

[19] Sibson, R., *SLINK: an optimally efficient algorithm for the single-link cluster method*, The Computer Journal, 1973.

[20] Silva, P., Marcal, A., Almeida da Silva, R., *Evaluation of Features for Leaf Discrimination*, Springer Lecture Notes in Computer Science, 2013.

[21] Vinh, N. X., Bailey, J., *Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance*, JMLR, 2011.

[22] Yang, B., Fu, X., Sidiropoulos, N., Hong, M., *Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering*, ICML, 2017.

[23] Ye, W., Goebl, S., Plant, C., Böhm, C., *FUSE: Full Spectral Clustering*, KDD, 2016.

[24] Xie, J., Girshick, R., Farhadi, A., *Unsupervised Deep Embedding for Clustering Analysis*, ICML, 2016.

[25] Zelnik-Manor, L., Perona, P., *Self-Tuning Spectral Clustering*, NIPS, 2004.