Knowledge and Information Systems
https://doi.org/10.1007/s10115-019-01388-5

REGULAR PAPER

Dataset-Transformation: improving clustering by enhancing the structure with DipScaling and DipTransformation

Benjamin Schelling[1] · Claudia Plant[1,2]

Received: 4 January 2019 / Revised: 12 July 2019 / Accepted: 22 July 2019

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 techniques—DipScaling and DipTransformation—which enhance the data set by rescaling and transforming its features and thus emphasizing and accentuating its structure. If the structure is sufficiently clear, clustering algorithms will perform far better. We refer to such techniques as "Dataset-Transformations" and try to provide a mathematical framework for them. To show that our algorithms work 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 approaches.

Keywords Dip-test · Dataset-Transformation · Data mining · Clustering · k-means

1 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.

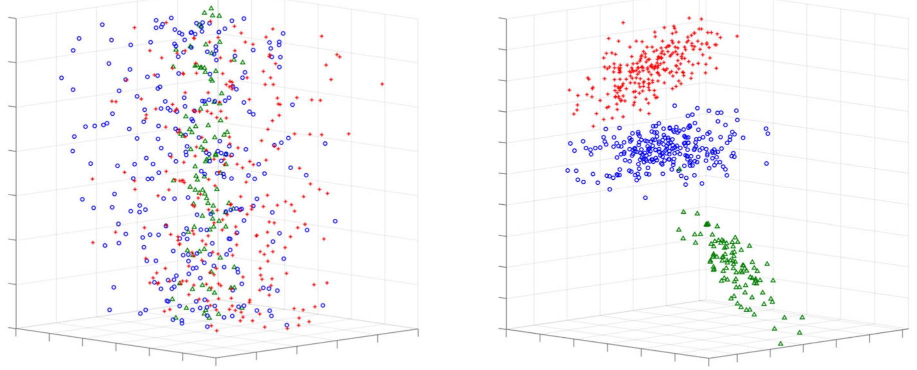
Confronted with a data set one cannot quite cluster, one would usually proceed by creating a new clustering method which is capable of dealing with the new and problematic type of data set, but this is not the approach we chose here. Instead, we wish to create methods which transform the data set—by enhancing the already existing structure—such that established clustering methods can deal with it. We present here DipScaling and DipTransformation,[1]

[1] Source code and data sets: https://dm.cs.univie.ac.at/research/downloads/.

✉ Benjamin Schelling
benjamin.schelling@univie.ac.at

[1] Faculty of Computer Science, Research Group Data Mining, University of Vienna, Vienna, Austria

[2] ds:UniVie, Vienna, Austria


(a) Our running example through this paper shown as a 3D scatter-plot.

(b) Our running example after DipTransformation in a 3D scatter-plot. It is now far easier to cluster.

Fig. 1 The running example before and after the DipTransformation

which are capable of accentuating structure and bringing the data set into a more clusterable form.

Consider the data shown in Fig. 1a as a 3D scatter plot of our 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 uniformly 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 yield surprisingly poor results on this data set. K-means scores merely 0.01 in NMI (Normalized Mutual Information),[2] DBSCAN [9], Spectral Clustering [16] and SingleLink [18] also perform disappointingly. The best choice appears to be EM [7] with an NMI score of 0.43.

Since the data set consists of a superfluous third dimension, we try dimensionality reduction techniques in the hope of adapting the data set into a more clusterable form. The combination of clustering and dimensionality reduction is well established and might yield better 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 with k-means. The best choice seems to be t-SNE [20] which scores approximately 0.78, but has highly varying results. (All these techniques were used in combination with k-means with correct k. The average of 100 runs is given.) The clusters are purely in the first two dimensions—so techniques like PROCLUS [1] and CLIQUE [2] which search for clusters in axis-parallel subspaces could be successful, but our experiments show otherwise (0.21 and 0.71 in NMI, correct k for PROCLUS).

DipScaling makes it possible to compensate for the unfortunate scaling of the features. We briefly stated that the problem lies partly therein that uniform/unimodal features (i.e. essentially structure-free features) receive the same degree of attention as such features that deviate from it. Uniform/unimodal features have no visible cluster structure. Clusters are not

[2] Normalized Mutual Information (NMI) [21] is one of the most often used evaluation measures for clustering. NMI is scaled between 0.0 and 1.0, with 0.0 the worst possible score and 1.0 the best.

distinguishable there and, thus, these features are difficult to cluster. Multimodal features, on the other hand, have clearly separated clusters. They are very important for clustering as the clusters can be found far easier. For k-means, which is our main focus here, this implicates that features with more structure should be larger scaled compared to features with barely any structure. If a feature is scaled to a very small range, the data points are almost the same in regard to this feature and, thus, this feature will have a very small impact. A very large scaling of a feature, on the other hand, means a high impact; the k-means clustering will be heavily influenced by those structure-rich, multimodal features in computing the centres of the clusters and the way the clusters are determined.

This requires a measure that evaluates the amount of structure of a feature and, therefore, its scaling. We find this in the Dip-test [12] explained in Sect. 2.1. The Dip-test gives an appropriate measurement of the structure a feature has and, thus, the scaling it "deserves". In Sect. 2.2 it is explained in more detail how this measurement is used.

DipScaling rescales the axes-parallel features. Restricting oneself to only the axes, however, is basically the same as assuming independence of the axes-parallel features, which is not necessarily the case. *DipTransformation* expands on DipScaling and generalizes it, so that this assumption is no longer needed. It is capable of handling data sets where the clusters are not necessarily axes parallel, such as our running example. DipTransformation searches for multimodal features that are non-axes parallel and then, when found, apply the strategy of DipScaling. DipTransformation is capable of 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 Fig. 1b). K-means now reaches an NMI of 0.97.

1.1 Contributions

This work presents a parameter-free method as well as an almost parameter-free method—DipScaling and DipTransformation—which are capable of improving the structure of a data set and thus allowing for better clustering. Our main focus lies with k-means, but this also holds for other methods as we will show for the standard clustering approaches. DipScaling and DipTransformation do not assume a specific distribution for the clusters or data. They simply enhance structure and thereby improve clustering. They are both deterministic and require no distance calculations. We extensively tested on real-world data sets for a wide range of algorithms.

1.2 Related work

The most common approach when a data set cannot be clustered well by any clustering 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, such as rescaling it into the [0,1]-interval. In addition, there is the Z-transformation (sometimes referred to as Z-normalization), which rescales the axes-parallel features to a mean of 0 and a variance of 1. Z-transformation 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 of dealing with such noise information. Of particular interest are FOSSCLU [10] and SubKMeans [15] 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 [22] and DCN [23] are being developed, with the aim of 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, spectral clustering techniques are similar to DipScaling and DipTransformation. They try to transform the data set into a more clusterable form. One of the most well known is the fundamental technique by Ng, Jordan and Weiss [16]. We also use the popular Self-Tuning Spectral Clustering [24] and FUSE [25] as state-of-the-art comparison methods.

DipScaling and DipTransformation use 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 [5] with the goal of estimating the number of clusters for k-means. After that, there is SkinnyDip [14], a technique to cluster in the presence of noise, using the Dip-test. Very recently has [19] also been published, a generalization of the Dip-test to higher dimensions. It is used as a tool to find out if there are multiple clusters in a data set or not to decide if clustering makes sense. Hence, it is a clustering support technique like ours. The Dip-test is still a rather unknown tool that has not yet found full recognition.

One must bear in mind while reading this that DipScaling and DipTransformation are not rivals for the mentioned techniques in the classical sense, but that they can be used as supporting techniques that ease the difficulty in the task they attempt. We will show in the experimental section (see Sect. 5) that they can benefit from DipScaling and DipTransformation.

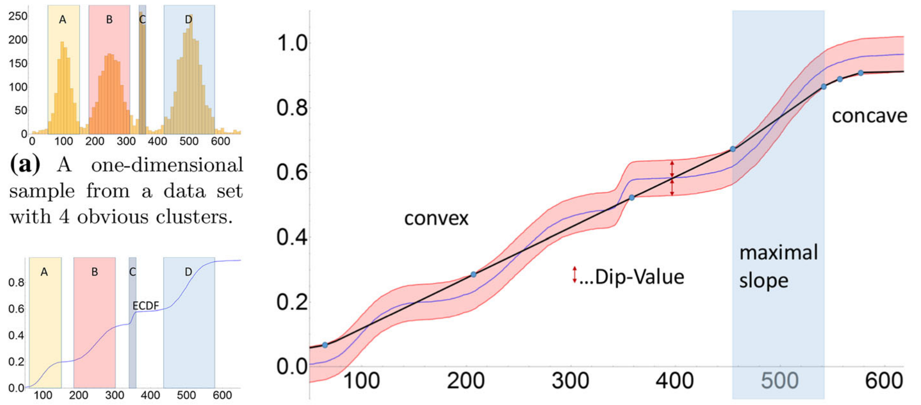
2 The algorithm

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

2.1 The Dip-test

The Dip-test was created by Hartigan & Hartigan in the 1980s as a measure of how much a one-dimensional 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. 2. 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 only for visual clarity. It, therefore, has no bin width parameter. In fact, it has no parameter at all.



(a) A one-dimensional sample from a data set with 4 obvious clusters.

(b) The ECDF of the data set from Fig. 2a.

(c) The Dip-test measures how far away a ECDF is from a unimodal distribution.

Fig. 2 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

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. 2c. 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 representing the closest possible unimodal distribution. The dip statistic (we refer to it as the dip value or “dip”) shows how far away the ECDF is from such a unimodal distribution. It can be understood as the distance of the sample to a unimodal distribution.

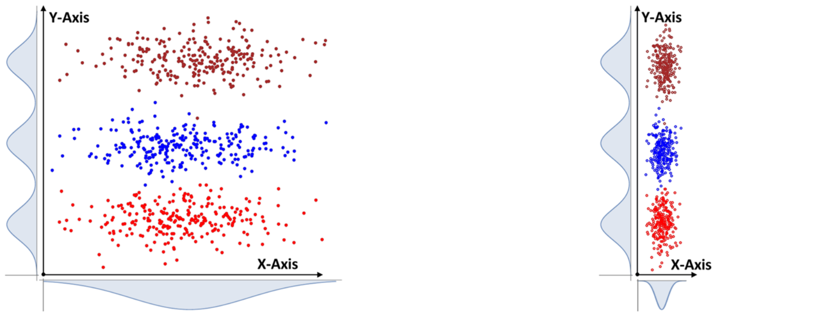
The Dip-test also returns another value, the probability of how likely a sample is unimodal, as well as the interval of the highest slope, but we only need the dip value. The dip value is always in the interval (0, 0.25]; hence, it is always positive.

The Dip-test itself 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 Hartigan and Hartigan [12].

2.2 Applying the Dip-test

The Dip-test gives a value, the dip value, which estimates how much structure can be found in a feature. We stated in the introduction that we use this estimation to rescale the features and thus improve clustering. For us, the Dip-test measures the influence the feature should have. The more influence it is supposed to have, the larger it will be scaled (in relation to the other features) and the greater is its importance in determining the result of *k*-means.

Let us explain the approach with Fig. 3. The figure shows a very simple data set consisting of three Gaussian distributed clusters. Even though the data set is very simple and should be easy to cluster for *k*-means (non-overlapping, Gaussian clusters), *k*-means performs rather poorly. The problem is obvious: the scaling of the clusters is very unfavourable. Thus, we rescale the features to make the structure of the data set more accessible. We



(a) The data set and the projections onto its axes to show the structure found there. **(b)** The axes are now scaled according to the found structure.

Fig. 3 A simple data set before (**a**) and after (**b**) applying DipScaling

restrict ourselves to rescaling the axes-parallel features, i.e. we stay in the framework of DipScaling.

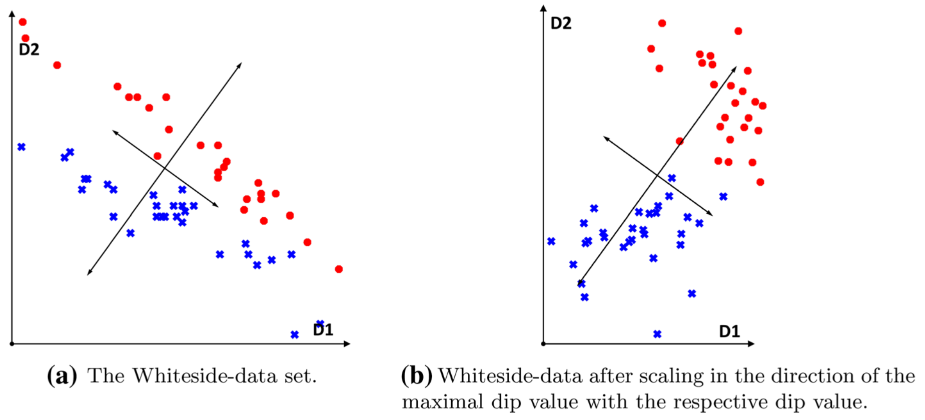
The first step is to analyse the features. For that, we project the data onto the axes. These projections can be seen in Fig. 3. The Y-axis has a clear cluster structure, i.e. three different clusters can be identified, while the X-axis is completely without structure, basically the clusters are completely undistinguishable. Measuring the amount of structure gives us a value of 0.009 for the X-axis and 0.063 for Y-axis. We rescale the X-axis in the interval [0,0.009] and the Y-axis in [0,0.063] and get the data set represented in Fig. 3b. The feature with the structure is now scaled comparatively large and has a high impact in clustering, just as we wanted.

This change makes 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 Normalized Mutual Information (NMI) [21] 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 initializations 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 (rather 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.

2.3 DipScaling

We have seen in Fig. 3 the effect that rescaling the features of a data set can have. The Dip-test is used to obtain an estimation of the amount of structure contained in a feature and this estimation—the dip value—is used for rescaling the feature. We refer to this procedure as DipScaling. For this, it is necessary that the original features can be handled separately, i.e. they are independent of each other. If they were in some way dependent, i.e. correlated, on each other the clusters would not be parallel to the axes. This is of course a very significant assumption, which is not always justified and which is not true for every data set. DipTransformation does not share this restriction. DipScaling is the foundation on which DipTransformation builds upon; it can be used on its own, as a very fast, first reshap-



(a) The Whiteside-data set. **(b)** Whiteside-data after scaling in the direction of the maximal dip value with the respective dip value.

Fig. 4 The direction of the feature with the maximal dip value, as well as its orthonormal direction. These directions are scaled with their dip values to show how much structure is found in these directions

ing of the data set, very much like Normalization and Z-Transformation. We will see in Sect. 5 that it can improve clustering, but most of the time DipTransformation is the better choice.

2.4 DipTransformation

If features are not independent, clusters are not stretched parallel to the axes. Let us look at the Whiteside data set [11] (depicted in Fig. 4a) as such a case. The Whiteside data set is a real-world data set.

The clusters are obviously not axes parallel. We have the same situation here as in the simple data set shown in Fig. 3, as in that k-means fares rather badly. To be precise the NMI value is 0.006. DipScaling is ineffective here; it improves clustering only minimally. The situation requires an approach that can find features, which are not axes parallel, but are interesting in regard to their dip value. In the data set shown in Fig. 3, this is not a relevant aspect, as the feature with a high dip value is axes parallel, but here this is not the case. The axes-parallel features of the Whiteside data are very uninteresting, i.e. they have a low dip value. But, if we project the data onto a straight line at an angle of roughly 123°, we find a feature with a very high dip value. Following our earlier arguments, we are very interested in these features, as they most likely contain the structure that determines the quality of a clustering.

In Fig. 5 we see that the dip value changes notably depending on the angle at which the dip value is measured. While the dip value changes continuously, it is full of local minima and maxima. Let us assume that we found the angle of the maximal dip value and scaled the Whiteside data set with this dip value in the direction of the angle. This leads to the transformed data set shown in Fig. 4b. A data set that can be clustered considerably better by k-means, due to the clearly more accessible structure of the data. In fact, we get an average NMI of 0.92. This is a massive improvement to 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 for the angle with the highest dip value will also not lead to a satisfactory

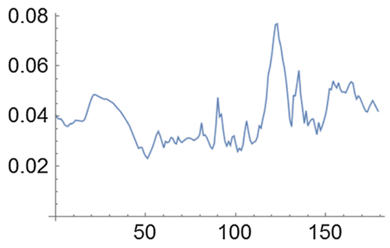


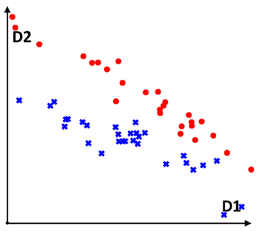
Fig. 5 The brute force approach to finding the maximal dip value is to measure the dip value at as many angles as possible. This graph shows the dip values of the Whiteside data set and how they change depending on the angle at which they are measured. Basically, the data set is rotated by, say, $10°$, the data set projected onto a axis, and the dip value of the now one-dimensional sample computed

result, since the data, as we have seen in Fig. 5, has more than a few local optima. Our solution to this predicament is to not only search for a single angle with maximal dip angle, but to find multiple angles with a high dip value and scale the data set along them. Basically, DipTransformation does not restrict itself to a single rescaling like DipScaling, but finds multiple interesting features. The algorithm scales the data set in various instances, so that the structure of the clusters becomes more clearly defined and more clusterable.

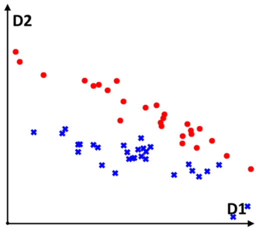
A search strategy to find interesting angles, i.e. with high dip values, is needed, but instead of changing the angle and analysing the projection, we rotate the data set and analyse the features that are now the axes. This is basically the same approach, but simplifies some computations. The procedure starts by computing the dip values of the axes, or more precisely the projection of the data set onto the axes. We then have some information about the "landscape" of the dip values depending on the angle. For the Whiteside data set the landscape is shown in Fig. 5. The difficulty is that the absolute maximal dip value depends on the specific data set. For Whiteside it is around 0.08, but other data sets may have a very different maximal dip value. Hence, we cannot transfer any expectation from one data set to another. Nevertheless, we have the information from the axes and this gives us a starting point. For example, for the Whiteside data set we know the two dip values, D_1 and D_2, along the axes. The ratio between those values $r = Max(\frac{D_1}{D_2}, \frac{D_2}{D_1})$ tells us something of about the "quality" of these features. If r is high, then there is a good chance that we have hit a good dip value. After all, high r means that either D_1 or D_2 is quite larger than the other, i.e. it is a feature with a high dip value, which we are looking for. We continue by rotating the data set in clockwise direction by an angle of $\frac{1}{r} * c$, with c as a rotation speed parameter. This ensures that we rotate the data set only by a small angle if one of the axes has a high dip value. Since the dip value changes continuously, it is quite possible that the axis is close to an angle with even higher dip value. On the other hand, if r is small, then both axes have similar dip value; thus, they are not interesting for our search of high dip values and the direct neighbourhood is probably not very interesting as well. Rotating by $\frac{1}{r} * c$ will lead to a larger leap and the uninteresting area is skipped.

The rotation speed parameter c can be freely selected. The larger c is selected, the shorter is 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 Sect. 2.6.)

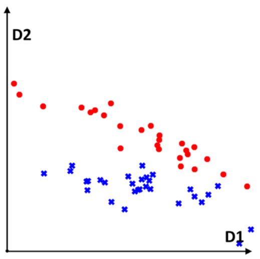
The algorithm remembers the maximal found dip value (we refer to it as $MaxDip$) and every time it finds a new maximal dip value, the axes are scaled with their respective dip values. The total degrees the data set has been rotated by is stored and after $360°$ the algorithm stops. For the Whiteside data set with $c = 5$, this leads to the transformation displayed in Fig. 6.



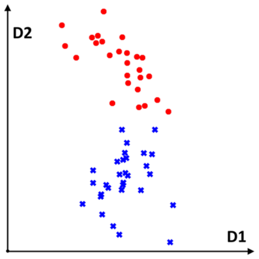
(a) The Whiteside data after the first rotation.



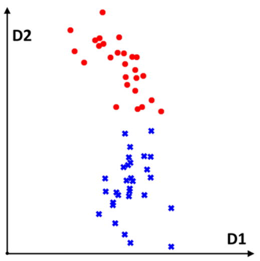
(b) Small changes become visible.



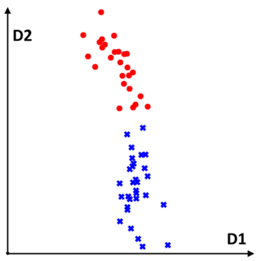
(c) The horizontal axis is now very close to a very large dip value.



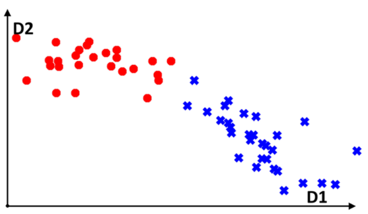
(d) The maximal dip value is now almost found. Rescaling the data is very influential.



(e) The maximal dip value comes even closer. The data set is only rotated by a very small angle.



(f) The maximal dip value is found.



(g) The final transformation.

Fig. 6 Steps in the DipTransformation of the Whiteside data set (**a**)–(**f**) and the final data set (**g**). There seem to be big leaps in the transformation; this is due to those cases, when the dip values of the axes become very different and scaling the data set seems to lead to enormous changes

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 maximal dip value is found. In the following iterations the maximal dip value of the current axes is 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 in each step of Fig. 6. The step from Fig. 6c to d seems to be enormous, but all that happens here is that we come very close to the overall maximal dip value, when rotating. The dip values of the axes are now very different and the scaling leads to a seemingly very much changed data set, but if closely examined, one sees that that all that happens is that the data set is stretched. The dip value of the axis with the clusters is rather large (the cluster structure is located here), while the other axis has a small dip value. Scaling the axes with the dip value causes what we wanted: A data set where the cluster structure is far more obvious. We are now rather close to the data set transformation from Fig. 4b.

The algorithm is not yet finished. It remembers the new $MaxDip$. The data set continues to rotate, but because we are very close to a high value the algorithm selects only a small rotation angle. This is advantageous because the next dip value of an axis is even higher. Scaling with these dip values leads to Fig. 6e, with an even more pronounced structure. The next iteration(s) change only very little. The data set does not change drastically, but becomes more compact and the clusters are defined more clearly with each iteration. Figure 6g shows the final state of the data set after the DipTransformation.

All in all, we can say that the proposed search strategy works very well, and we are very good at finding interesting angles with high dip values. Thus, the resulting 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, for example, $c = 9, 8, 7, 6, \ldots$. However, the transformed data set may look slightly different for a different value of c.

This transformation is easier to cluster in comparison to the original Whiteside data set shown in Fig. 4a, but also when comparing it to the transformation along the maximal dip value angle shown in Fig. 4b. 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, as the one there. Scaling along axes leads to a basis transformation that stretches the basis vectors, but leaves orthogonality intact. Applying the transformation method sketched above also leads to a change in basis vectors, which no longer implies that two previously normal (i.e. perpendicular) vectors are normal to each other afterwards.

The foci of DipScaling and DipTransformation are on k-means as we have stated, but we see from Fig. 6 that other techniques might also benefit from this approach. We will explore in Sect. 5 how the performance of other techniques is influenced by this (and other) transformed data set(s).

2.5 More than two dimensions

DipTransformation for a two-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 we found to work best is explained in the following.

The main difference is that there are more than two directions, the data set can be rotated in. It seems 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

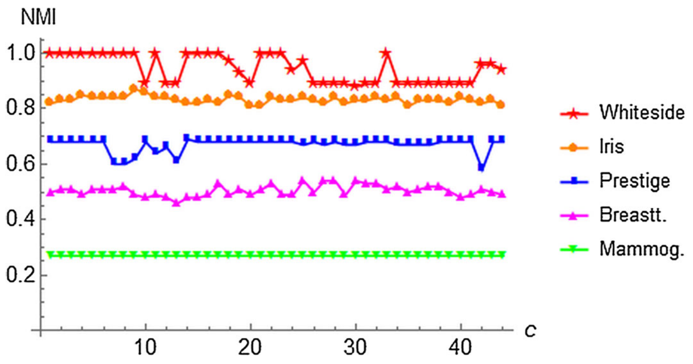


Fig. 7 The NMI value for k-means with correct values for k vs the rotation speed parameter c for five real-world data sets

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. Due to the difficulties here, we simply avoid them by rotating only the 2D-subspace spanned by two axes vectors. This has the advantage that we can simply take the earlier strategy and apply it to the 2D-subspace. The only remaining question is which of the axes (respectively their spanned subspace) one should rotate first, but this seems to be a decision without much influence, according to the experiments we conducted. The order seems to have limited influence, as long as all directions are covered. Executing only the rotation with the highest/lowest dip value, for example, seems to impair the transformation. Thus, a strategy that alternately rotates in all direction seems to be the best choice.

Through all rotations the algorithm remembers the maximal found dip value as $MaxDip$, just as before. Whenever the rotated data set has a dip value larger than $MaxDip$, the data set is scaled and $MaxDip$ is updated. The rotation angles are calculated in the same way as for a two-dimensional data set.

One has to keep in mind that in higher-dimensional data sets, 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 two-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 $d \cdot 360°$ have to be traversed. This range ascertains that (theoretically) all maxima can be found. Furthermore, it is not necessary to find the maximal dip values exactly; being close enough is sufficient to assure a good transformation.

2.6 The rotation speed parameter *c*

The rotation speed parameter c has been introduced in Sect. 2.4, and we now want to analyse its effect on the DipTransformation. Figure 7 shows how NMI (for k-means) changes with different values of c and the effect is (mostly) 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 Sect. 5.

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. However, for an unknown

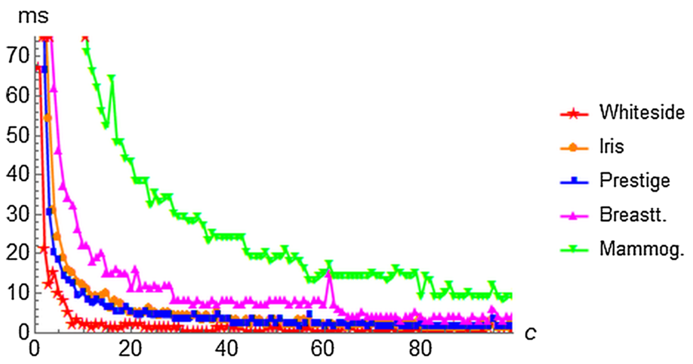


Fig. 8 The runtime of DipTransformation in milliseconds depending on the value of the rotation speed parameter c for five real-world data sets

data set a smaller value for c is 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. Figure 8 shows how the runtime is linked to the parameter c and how it decreases for larger 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 $c = 5$.

3 An attempt at a mathematical framework for Dataset-Transformations

We have already mentioned that there are basically no Dataset-Transformation-based techniques, although we have never clearly stated what we consider to be such a transformation. This could be due to the apparent obviousness of what it means; we would nevertheless like to try give a more formal definition.

The most obvious attempt is to just restrict Dataset-Transformation to continuous Transformations, i.e.,

Definition 1 Let D be a data set, consisting of n $d-$dimensional data points $x_1, \dots, x_n$. The operator T is a Dataset-Transformation, if $\forall \epsilon > 0 \quad \exists \delta > 0$, such that, $\forall i, j$ if $||x_i - x_j|| < \delta$, then $||T(x_i) - T(x_j)|| < \epsilon$.

This is basically the straightforward application of the definition of continuity to our problem, but the problems are obvious. The data set is necessarily a finite one, so the restriction of $\forall \epsilon > 0 \quad \exists \delta > 0$ cannot apply here, as one can always make δ so small that the $||x_i - x_j|| < \delta$-requirement holds for no data point and is thus trivially fulfilled. Following this line of thought, every operator on a finite set can be considered continuous and this is too broad. One can extend and vary this approach, but the main problem of D being finite remains and foils all approaches that restrict themselves to D. Eventually, the only approach left is to lift the restriction onto D and consider a potential Dataset-Transformation T as an operator on $\mathbb{R}^d$, with $D \subset \mathbb{R}^d$. This is of course a restriction as D can now be merely numerical in nature, but one has to consider that the Dip-test needs numerical values any way. It might also be possible to not restrict oneself to Euclidean spaces, as is done here, but, for example, Banach spaces instead. An advantage won by that though is not discernible. If such a generalization would become necessary at a future point in time, it can be easily obtained on this foundation, but for the moment we restrict ourselves to Euclidean spaces and accept that basically all available numerical data sets would be considered subsets of $\mathbb{R}^d$ anyway.

With the decision made that a Dataset-Transformation warps the space itself and with it the data points, we can formulate the definitions we wanted to obtain. The most obvious attempt would be a continuous transformation of the space, and hence:

Definition 2 (*Dataset-Transformation*) Let D be a data set, consisting of n $d-$dimensional data points $x_1, \ldots, x_n$. The operator $T \ : \ \mathbb{R}^d \mapsto \mathbb{R}^m$, with $m \leq d$ and $D \subset \mathbb{R}^d$ is a *Dataset-Transformation*, if it is continuous.

This by itself is in excess of what is actually needed. If, for example, all data points would contain only positive values a restriction to $\mathbb{R}^d_+$ would be enough. Nevertheless, we stick with this definition for now. This raises the question if DipScaling and DipTransformation can be considered as Dataset-Transformations.

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 being the entries in the diagonal. Since the Dip-test values c_i can never be zero, the determinant of the scaling matrix is nonzero. 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, DT is a matrix with nonzero determinant, i.e. a basis transformation.

This shows that the DipTransformation (and therefore DipScaling, for which the proof holds as well) is continuous, as every basis transformation is continuous, and hence:

Theorem 2 *The DipTransformation DT is a* ***Dataset-Transformation*** *as defined in Definition 2.*

Defining a Dataset-Transformation as a continuous change of the data set seems to be the most straightforward and meaningful approach. Let us consider for example t-SNE [20] as a technique that "transforms" a data set: It starts with the higher-dimensional data points, computes similarities and places the data points into a lower-dimensional space. While the technique intends to keep close data points close to each other, this is not guaranteed, and they can be put far away from each other. The data space is basically ripped apart and sometimes data points end up in completely new neighbourhoods. A continuous transformation like DipTransformation guarantees that this cannot happen, which is why we considered it as the "most obvious attempt". The data set is distorted and distances changed, but the basic shape of the data set is kept.

This holds especially, if the dimensionality is kept the same. The definition with $T \ : \ \mathbb{R}^d \mapsto \mathbb{R}^m$, with $m \leq d$ does not rule out dimensionality-reducing techniques like PCA. PCA can be understood as finding the vector with the highest variance in the data and projecting it onto the first axis, the vector with the second highest variance onto the second axis, and so on. If the dimensionality is kept the same, then PCA is basically a basis transformation itself and, therefore, a Dataset-Transformation according to our definition. If the dimensionality is not kept the same, then PCA is nevertheless continuous (though not a basis transformation any more) and thus a Dataset-Transformation. The same argument can be made for ICA. Normalization and Z-Transformation are obviously also covered by our definition, contrary to, for example, t-SNE, as mentioned earlier, and various methods like SynC.

We stated that the basic shape of the data set is kept, if the dimensionality is kept the same. If the dimensionality is kept, the transformation is also **bijective**; it is possible to reconstruct the original data set, if required. With a dimensionality reduction technique reconstructing the original data set is basically impossible as some information (however slight) is lost. This argument is based on DipScaling/DipTransformation being basis transformations; thus, it is not applicable to our definition of Dataset-Transformation generally. Including the requirement of bijectivity into the definition of a Dataset-Transformation is possible, as that guarantees that the effect could be undone, but continuity seems to be enough, as that means that the local neighbourhood of data point is (roughly) kept the same. This means that the structure is (roughly) the same, but—ideally—more pronounced.

Most clustering methods cannot be subsumed under our definition of a Dataset-Transformation, as they are not interested in transforming a data set. They simply want to find the clusters hidden in it, and are, therefore, not covered here. Our approach is different because we want to enhance and simplify a data set and we hope to give a first theoretic basis with this definition onto which future works can be placed. This basis is not complex, but it is a start and we have high hopes that it will prove itself as a stable foundation for an—hopefully—emerging new field in Data Mining.

4 Runtime and pseudocode

The runtime of DipScaling is easy to estimate. First, one needs to compute the dip values for every axis, which is $d \cdot \mathcal{O}(n \log(n))$, due to the necessity to order the input for the Dip-test, and the subsequent scaling of the axes with the values which is $d \cdot \mathcal{O}(n)$. Hence, as a total for the runtime for DipScaling we get

$$\mathcal{O}(d \cdot n \log(n)).$$

The DipTransformation algorithm (outlined in Algorithm 2) is slightly more complicated to estimate. 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:

$$\begin{aligned} &\mathcal{O}(n) + \mathcal{O}(n \log n) + l \cdot d \cdot d \cdot \big(\mathcal{O}(n) + \mathcal{O}(n \log(n)) + \mathcal{O}(n)\big) \\ &\quad \approx \mathcal{O}\big(l \cdot d^2 \cdot n \log(n)\big) \end{aligned}$$

Experiments on the runtime can be found in Sect. 5.5.

Algorithm 1 DipScaling

```
Require: Data $D$
1: procedure DIPSCALE($D$)
2:     for i = 1,…,dim do
3:         Compute $DipValue$ for axis $i$
4:     end for
5:     for i = 1,…,dim do
6:         Rescale values of axis $i$ to [0,DipValue($i$)]
7:     end for
8:     return $D$
9: end procedure
```

Algorithm 2 DipTransformation

```
Require: Data $D$, Rotationspeed $c$
1: procedure DIPTRANSFORMATION($D$, $c$)
2:     $Degree \leftarrow 0$
3:     Compute $DipValues$
4:     DipScale($D$,$DipValues$)
5:     $MaxDip \leftarrow Max(DipValues)$
6:     while $Degree < dim * 180°$ do
7:         for i = 1,…,dim do
8:             for j = i+1,…,dim do
9:                 $a \leftarrow Max(Dip(i)/Dip(j), Dip(j)/Dip(i))$
10:                Turn $D(i, j)$ by angle $c/a$
11:                $Degree \leftarrow Degree + c/a$
12:                Compute $DipValues$
13:                if $Max(DipValues) > MaxDip$ then
14:                    DipScale($D$,$DipValues$)
15:                    $MaxDip \leftarrow Max(DipValues)$
16:                end if
17:            end for
18:        end for
19:    end while
20:    return $D$
21: end procedure
```

5 Experiments

Convincing someone that a data set is easy to cluster, if the data set is more than two-dimensional, is difficult. The goal of DipScaling and DipTransformation though is to ensure that a data set becomes easier to cluster. This work will of course show with NMI values of experiments on real-world data sets that DipTransformation is capable of doing that, but we would also like to show that with plots. Figure 9 shows pairwise plots of the “Banknote Authentication” data set from the UCI Repository [8]. This data set was chosen because it has a small dimensionality of four, so that all pairwise plots can be shown. Figure 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 this can be expressed with an NMI value of 0.03 for k-means with the correct value for k. After the DipTransformation (shown in Fig. 9), 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. DipScaling has a far less impressive effect and improves clustering to a

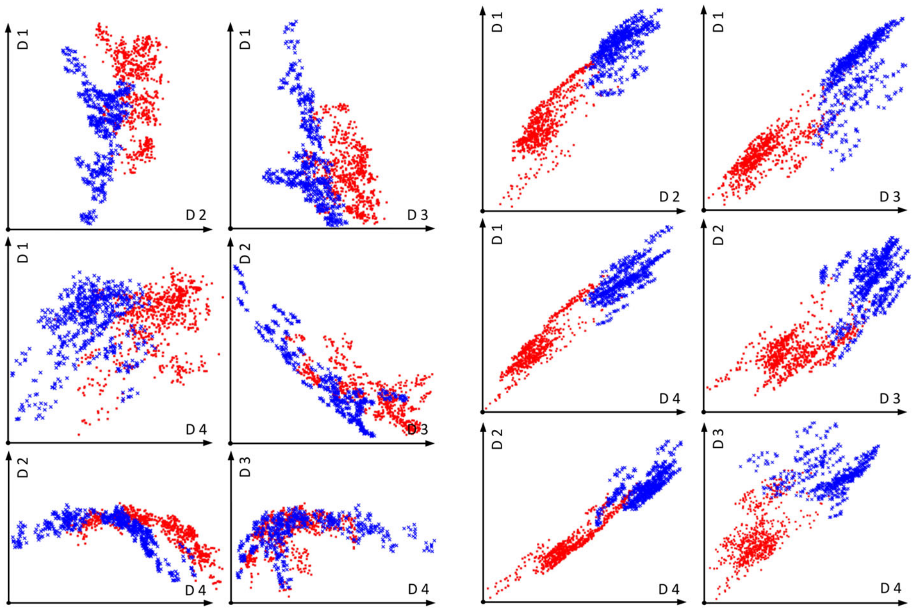


Fig. 9 Pairwise plot of the Banknote Authentication data set before and after DipTransformation. The dimensions are given as axes label. The clusters are visibly better separated from each other after the DipTransformation, an effect which cannot be obtained with DipScaling alone

NMI value of 0.18, which is not surprising as the clusters are obviously not axes parallel, i.e. independent.

5.1 Synthetic data

The first analysed data set is the synthetic data set given in Fig. 1a. 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 1 shows the NMI results. Most of them are not impressive, with 11 of the 20 algorithms scoring below 0.10. DipScaling leads to a somewhat better picture, but as the features are not independent, it is not a good choice. The DipTransformation on the other hands leads to a massively enhanced data set with clearly stronger defined structure. (Pairwise plot is shown in Fig. 1b.) The three clusters that were stretched and scaled quite unfavourable for clustering before are now well separated and compact. Clustering of this data set is far easier, and the results shown in Table 1 demonstrate this. All of the used algorithms improve due to the DipTransformation—on average 0.636 in NMI. After the DipTransformation 12 of the 20 algorithms are better than 0.90.

FossClu [10] and SubKMeans [15] 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.

Table 1 Clustering of the running example before and after DipTransformation

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 + k-means	0.03	0.63
ICA + k-means	0.01	0.98
t-SNE + k-means	0.78	0.80
PROCLUS	0.23	0.92
CLIQUE	0.71	0.98
DEC	0.38	0.53
DCN	0.12	0.58

The average improvement in NMI is 0.636

5.2 Real-world data sets

We have tested DipScaling and 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 Transformations 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 2. The difference in clustering quality is obvious. While k-means usually fares somewhat lacking on the original data sets 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 the structure so that k-means can cluster the data sets better than the compared methods, especially combined with DipTransformation. In 8 of the 10 cases the DipTransformation + k-means is now the best choice (sometimes together with other methods), in one case is DipScaling + k-means even the best choice (with the smallest possible lead of 0.01 over DipTransformation) and in the last case DipScaling and DipTransformation loose with a small deficit of 0.02. DipTransformation clearly improves the structure of the data sets, so that k-means is now a very good choice. DipScaling is often also a good choice and improves k-means, but it is also obvious that it does not always perform on the same level as DipTransformation. We assume that this is due to the level of independence of the features in the data sets. The Whiteside data set has features far from being independent, and hence, DipScaling is not a good choice. Other data sets like Prestige and Breast Tissue seem to fit more into the assumption of independence.

Table 2 Experimental results

Data set	Whiteside	Skinsegmen.	Banknote	Iris	Prestige	Userknow.	Mammographic	Seeds	Breast tissue	Leaf
# of data points	56	245,057	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**
DipScaling	0.01	0.16	0.18	0.81	**0.68**	0.45	0.27	0.73	**0.52**	0.68
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	0.72	0.23	0.11	0.78	0.43	0.34	0.14	0.64	0.38	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**
ICA + *k*-means	0.64	–	0.08	0.57	0.46	0.19	0.12	0.61	0.42	**0.69**

Bold values indicate the best result
All non-deterministic results have been repeated 100 times, and the average is given. Parameters as described

We chose the algorithms we found to be most relevant as comparison methods here. This included the data set transformation techniques of normalizing and k-Transformation, the standard data mining algorithms DBSCAN [9], EM [7] and SingleLink [18], DipMeans [5] and SkinnyDip [14] as techniques based on the Dip-test, SubKMeans [15] and FossClu [10] as the most similar subspace clustering techniques, the aforementioned Spectral Clustering methods, SynC [4], as well as PCA, ICA and t-SNE in combination with k-means. For PCA, ICA and t-SNE we decided not to reduce the dimensionality, because there is no completely straightforward answer on how far one should reduce the dimensionality and because Dip-Transformation also does not reduce dimensionality. For DipScaling alone Normalization and Z-Transformation would be the most comparable methods.

We used the NMI score [21] as a measure of comparison. We are aware of another frequently used, state-of-the-art metric for evaluating clustering results, called Adjusted Mutual Information (AMI). We have found that the results do not vary to the extent that the "take home message" changes, so we omit AMI results to reduce clutter. The same also holds for other variants of NMI. There are multiple ways on normalizing NMI, but the difference is usually also not very large.

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.

Mammographic mass This UCI data set contains data points with missing entries. These fragmented data points have been removed from the data set. All methods were tested on the cleaned data set.

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, the size was 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.

Spectral clustering If this paper refers to Spectral Clustering as an algorithm and not the class of algorithms, then the classical algorithm by Ng et al. [16] is meant.

Besides these considerations it is most noticeable that k-means++ leads to the same increase in NMI as the DipTransformation on the Skinsegmentation data set. 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.

5.3 *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 3. 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++ (≤ 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 (also DipScaling) can be used together with all types of support techniques for k-means (or other clustering algorithms). For example, X-means [17] can be used to find the number of clusters, k-means–[6] to find outliers, k-means++ to find an initialization, SubKMeans to find a subspace and all this in combination with the DipTransformation.

5.4 DipScaling/DipTransformation and clustering algorithms besides *K*-means

DipScaling and DipTransformation were developed with a focus on k-means, but as we have stated throughout our work, they enhance the structure; they do not adapt the data set so that it only fits k-means and therefore other algorithms can also benefit from them. We have seen the transformation of the Whiteside as well as the Banknote Authentication data set in Figs. 6 and 9, respectively. 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 transformations with DipScaling and DipTransformation with 4 standard data mining algorithms, i.e EM, DBSCAN, SingleLink and Spectral Clustering. The results can be seen in Table 4. 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 k-means is also included here. For DipTransformation we see a tiny decrease in clustering quality of 0.01 in NMI in two cases. 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. For DipScaling the situation is somewhat similar. In two cases the clustering quality decreases (both in combination with SingleLink), in 6 it stays the same and in 17 it increases. It improves on average by 0.096 in NMI.

To find out how compatible DipScaling and DipTransformation are with the other standard clustering approaches in comparison to k-means, we made Tables 4, 5 and 6. It shows by how much the standard clustering techniques improve in combination with our Dataset-Transformations on the 5 data sets. While k-means is the one that profits the most, so is, for example, Spectral Clustering not far off. SingleLink is even closer, but when looking at Tables 4 and 5 it becomes clear that SingleLink is more volatile, while Spectral Clustering profits more constantly.

This should, in combination with Figs. 6 and 9, be a very convincing argument that DipTransformation as well as DipScaling can play an important role in clustering as support techniques applied to the data set before clustering to enhance structure.

Table 3 *k*-means++ as well as *k*-means before and after the DipTransformation

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**

Bold values indicate the best result
Parameters as described

Table 4 Various clustering algorithms before and after DipTransformation

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 bold is the better result of the data set before and after the transformation shown
Parameters as before. On average the clustering improves by 0.223 (measured in NMI)

Table 5 Various clustering algorithms before and after DipScaling

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**	**0.03**	**0.53**
DBSCAN					
Before	**0.27**	**0.62**	0.54	0.14	0.41
After	**0.27**	**0.62**	**0.61**	**0.15**	**0.44**
SingleLink					
Before	**0.11**	**0.61**	0.08	0.00	**0.27**
After	**0.11**	0.60	**0.54**	**0.23**	0.12
Spectral clustering					
Before	**0.06**	0.60	0.60	0.09	0.45
After	**0.06**	**0.65**	**0.64**	**0.28**	**0.51**
k-means					
Before	**0.01**	0.70	0.51	0.11	0.32
After	**0.01**	**0.81**	**0.68**	**0.27**	**0.52**

In bold is the better result of the data set before and after the transformation shown
Parameters as before. On average the clustering improves by 0.096 (measured in NMI)

Table 6 Average improvement for the 5 standard clustering algorithms on the 5 data sets in combination with DipTransformation and DipScaling

Method	k-means	EM	Spectral C.	SingleLink	DBSCAN
DipTransformation	0.33	0.14	0.25	0.28	0.11
DipScaling	0.13	0.15	0.07	0.11	0.02

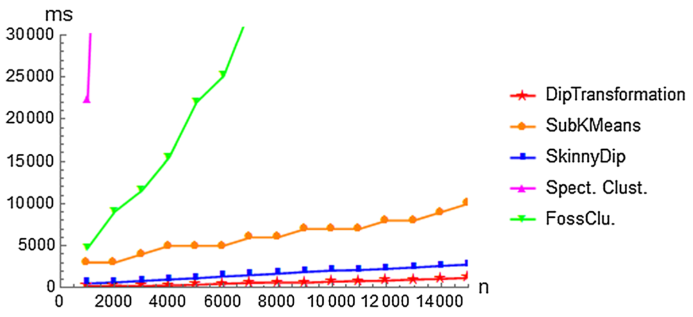


Fig. 10 Runtime relative to the data set size n. Dimensionality is 5

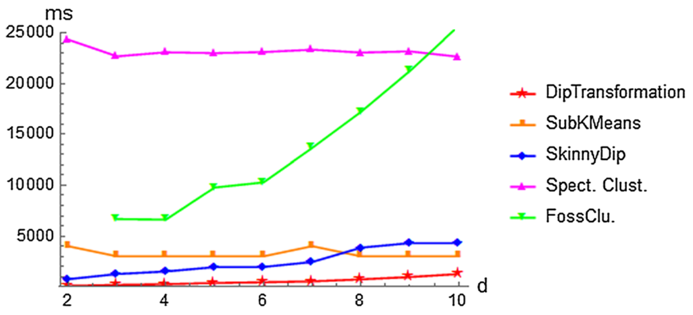


Fig. 11 Runtime relative to the dimensionality of the data set d. Data set size is $\approx$ 1.500 data points

5.5 Runtime comparisons

For the DipTransformation 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. 10. 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 tested algorithms. DipTransformation is faster for all data sets. To ensure comparability in this test is also the runtime of k-means included in the measured time for DipTransformation, since the other methods cluster data, which DipTransformation does not do by itself.

Besides the size of the data set also the influence of the dimensionality of the data set on the runtime is essential. This is shown in Fig. 11. We created 9 data sets ranging in dimensionality from 2 to 10 with 1.000 data points each. Here DipTransformation (+k-means) is 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 very good, so the conjecture that at some

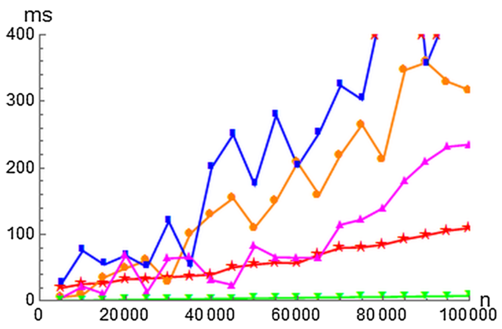


Fig. 12 Runtime relative to the data set size n. Dimensionality is 5

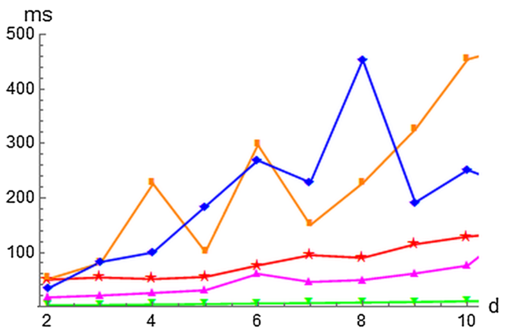
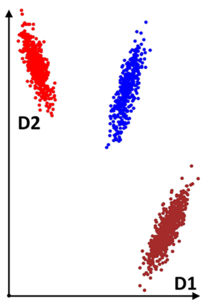


Fig. 13 Runtime relative to the dimensionality of the data set d. Data set size is $\approx$ 50.000 data points

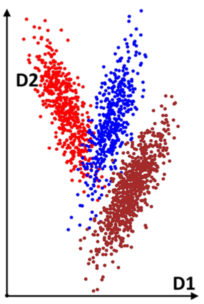
point DipTransformation will need more time than, for example, Spectral Clustering is likely. However, if one extrapolates from the curves, it seems as if that would happen at a rather high dimensionality.

We compared DipTransformation with the algorithms which we found most similar to it. DipScaling is a different, more basic type of method, and hence, we also compare it with different approaches. The closest related approaches are doubtlessly Normalization and Z-Transformation, PCA and ICA are also included as basic techniques. Contrary to before the runtime for k-means is not included (also not for the compared methods). The most obvious difference to DipTransformation is the overall runtime. DipScaling is far faster than DipTransformation, which is no surprise, considering DipScalings simplicity and that it is used as part of DipTransformation.

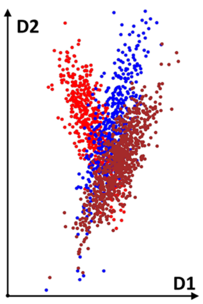
The estimation for the dependency on the number of data points and the dependency on the dimensionality made in Sect. 4 for DipScaling was $\mathcal{O}(n \log n)$ and $\mathcal{O}(d)$, respectively. These are both correct according to our runtime experiments depicted in Figs. 12 and 13. It is surprising that Z-Transformation, respectively PCA/ICA, is slightly erratic, when testing the dependencies, but this is probably due to implementation details. For all of them the standard R implementations were used, contrary to Normalization, which was implemented by us. R tends to call on C and/or Fortran code, so the assumption is close that internal methods/libraries are called, which might cause the fluctuations. Our implementation of Normalization has no (relevant) fluctuations, so this seems probable. The fluctuations are small enough, so that the trend can be seen and DipScaling is at the very least in the same range of runtime needs as the compared methods, which means that it is very fast.



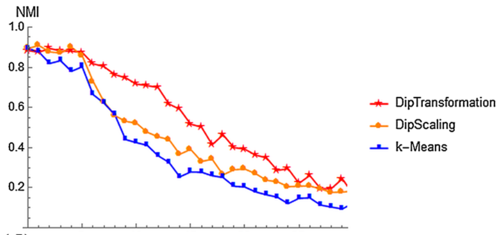
(a) Clearly separated clusters - high structure.



(b) Clusters start to overlap - structure is slowly disappearing.



(c) Overlapping clusters - no structure visible.



(d) Change in NMI depending on the amount of structure found in the original data set.

Fig. 14 Behaviour of DipTransformation and DipScaling relative to the "amount of structure" present in the data set

Algorithms are implemented in Java (DipTransformation and FossClu), Scala (SubKMeans) and R (Spectral Clustering, SkinnyDip, PCA, ICA, Normalization and Z-Transformation) and executed on an Intel Xeon E5 with 16 Gb RAM.

5.6 Structure

We have been confronted with the question of how the DipTransformation fares in regard to the amount of structure contained in the data set. The difficulty here is of course that "structure" is not a perfectly well-defined term, and hence, the "amount of structure" is difficult to measure. We decided on the following: We start with a data set, where the clusters are clearly separated (see Fig. 14a). The data set is very simple (k-means scores above 0.90 in NMI) and the common consensus would most likely be that the data set is well structured. We created more data sets in almost the same style, but the variance in the clusters is linearly increased. The clusters become wider and start to overlap (Fig. 14b). While the clusters are still somewhat well separated, they are harder to distinguish and some of the data points could belong to any cluster. K-means fares worse here (≈ 0.50 in NMI). The structure of the data set becomes less obvious, as the clusters start to overlap. Increasing the variance even further leads to basically overlapping clusters (Fig. 14c), which are now mostly inseparable. Many data points could belong to any cluster. At this point we feel justified to state that the data set contains almost no structure and the data set is not much more than a mush of data points. The performance of k-means drops below 0.10 in NMI.

We have changed the variance of the clusters in a range so that k-means performs from 0.90 (high structure, well-separated clusters) down to 0.10 (barely any structure, overlapping clusters) in NMI on the data sets and tested how DipTransformation and DipScaling influence the clustering results. If k-means performs very well, then the Dataset-Transformations have limited influence and improve clustering only slightly. As the data sets become more difficult to cluster, the influence of DipTransformation and DipScaling becomes more obvious. DipTransformation is the better choice here compared to DipScaling and can improve clustering immensely. A more difficult data set also makes it more difficult for the Dataset-Transformation. When the data set becomes not much more than a mush of data points, they also have too little information to have a strong effect; there is not enough structure left, as that it could be enhanced any more. DipScaling is in this case roughly as effective as DipTransformation.

6 Limitations and outlook

DipTransformation (and partly DipScaling) is a very powerful technique for enhancing the structure and emphasizing clusters, but there are certain limitations. For example, if two clusters overlap or interlock, DipTransformation would by design not be able to separate them. It stretches and scales the data. Therefore, all clusters 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 the DipTransformation from working properly. DipScaling has the same restrictions; furthermore, it assumes independent features, which is very often not the case.

In terms of runtime, the main constraint we encountered was the $\mathcal{O}(d^2)$-dependency of DipTransformation on the dimensionality of a data set. For very high-dimensional data sets, it might be useful to combine DipTransformation with a dimensionality reduction algorithm or stick to DipScaling. DipScaling might not improve clustering quite as much as DipTransformation, but it is very fast.

We do intend to implement a dimensionality-reducing feature into the DipTransformation. The Dip-test provides a probability estimation 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 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.

7 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 our Dataset-Transformations 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 subideal choice and other algorithms were clearly better. After applying DipTransformation/DipScaling, k-means was the best performing algorithm on all but one data set.

We have also shown that DipTransformation and DipScaling are compatible with other algorithms and also improve their clustering results. They can therefore be used as a pre-clustering step that enhances the data set, and the clustering algorithm can be chosen according to the users preferences.

Acknowledgements Open access funding provided by University of Vienna.

References

1. Aggarwal CC, Wolf JL, Yu PS, Procopiuc C, Park JS (1999) Fast algorithms for projected clustering, In: Proceedings of the 1999 ACM SIGMOD international conference on management of data, SIGMOD '99, ACM, New York, NY, pp 61–72
2. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications, In: Proceedings of the 1998 ACM SIGMOD international conference on management of data, SIGMOD '98, ACM, New York, NY, pp 94–105
3. Arthur D, Vassilvitskii S (2007) K-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp 1027–1035
4. Böhm C, Plant C, Shao J, Yang Q (2010) Clustering by synchronization, In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '10, ACM, New York, NY, USA, pp 583–592
5. Chamalis T, Likas A (2018) The projected dip-means clustering algorithm. In: Proceedings of the 10th Hellenic conference on artificial intelligence, SETN '18, pp 14:1–14:7
6. Chawla S, Gionis A (2013) k-means: a unified approach to clustering and outlier detection, In: Proceedings of the 13th SIAM international conference on data mining, May 2-4, 2013. Austin, Texas, USA, pp 189–197
7. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc Ser B 39(1):1–38
8. Dua D, Graff C (2019) UCI machine learning repository. http://archive.ics.uci.edu/ml
9. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, In: Proceedings of the second international conference on knowledge discovery and data mining, KDD'96, AAAI Press, pp 226–231
10. Goebl S, He X, Plant C, Böhm C (2014) Finding the optimal subspace for clustering. In: Proceedings of the 2014 IEEE international conference on data mining, ICDM '14, pp 130–139
11. Hand DJ, Daly F, McConway K, Lunn D, Ostrowski E (1993) A handbook of small data sets. Chapman and Hall, London
12. Hartigan JA, Hartigan PM (1985) The dip test of unimodality. Ann Stat 13(1):70–84
13. Kriegel H-P, Kröger P, Zimek A (2009) Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans Knowl Discov Data 3(1):1:1–1:58
14. Maurus S, Plant C (2016) Skinny-dip: clustering in a sea of noise. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16, pp 1055–1064
15. Mautz D, Ye W, Plant C, Böhm C (2017) Towards an optimal subspace for k-means. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '17, pp 365–373
16. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: Dietterich TG, Becker S, Ghahramani Z (eds) Advances in neural information processing systems 14. MIT Press, Cambridge, pp 849–856
17. Pelleg D, Moore A (2000) X-means: extending k-means with efficient estimation of the number of clusters. In: Proceedings of the 17th international conference on machine learning, Morgan Kaufmann, pp 727–734

18. Sibson R (1973) Slink: an optimally efficient algorithm for the single-link cluster method. Comput J 16(1):30–34
19. Siffer A, Fouque P-A, Termier A, Largouët C (2018) Are your data gathered? In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '18, pp 2210–2218
20. Van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
21. Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854
22. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: Balcan MF, Weinberger KQ (eds) Proceedings of the 33rd international conference on machine learning, vol 48 of Proceedings of Machine Learning Research, PMLR, New York, New York, USA, pp 478–487
23. Yang B, Fu X, Sidiropoulos ND, Hong M (2017) Towards k-means-friendly spaces: simultaneous deep learning and clustering. In: Proceedings of the 34th international conference on machine learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pp 3861–3870
24. Yang C, Zhang X, Jiao L, Wang G (2008) Self-tuning semi-supervised spectral clustering. In: 2008 international conference on computational intelligence and security, CIS 2008, 13–17 December 2008, Suzhou, China, volume 1, conference papers, pp 1–5
25. Ye W, Goebl S, Plant C, Böhm C (2016) Fuse: full spectral clustering. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16, ACM, New York, NY, USA, pp 1985–1994

Benjamin Schelling received a B.A. of Philosophy as well as a B.Sc. and M.Sc. of Mathematics at the University of Vienna, Austria. He is currently enrolled as a PhD student at the Faculty of Computer Science at the Data Mining research group at the University of Vienna. His main research interest is support methods for clustering, such as Dataset-Transformations.

Claudia Plant is full professor of computer science and leader of the Data Mining research group at University of Vienna, Austria. Her research focuses on new methods for exploratory data mining, e.g. clustering, anomaly detection, graph mining and matrix factorization. Many approaches relate unsupervised learning to data compression, i.e. the better the found patterns compress the data, the more information we have learned. Other methods rely on finding statistically independent patterns or multiple non-redundant solutions, on ensemble learning or on nature-inspired concepts such as synchronization. Indexing techniques and methods for parallel hardware support explore massive data. Claudia Plant has co-authored over 100 peer-reviewed publications, among them many contributions to the top-level data mining conferences KDD and ICDM and 3 Best Paper Awards. Papers on scalability aspects appeared at SIGMOD, ICDE and the results of interdisciplinary projects in leading application-related journals such as Bioinformatics, Cerebral Cortex and Water Research.