24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Non-linear Cluster Enhancement: Forcing Clusters into a compact shape

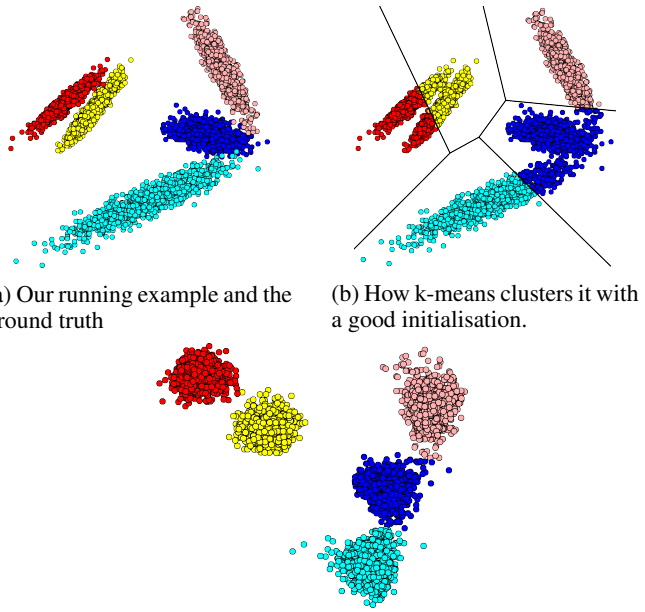
Benjamin Schelling[1] and **Lukas Miklautz**[1] and **Claudia Plant**[1,2]

Abstract. K-means is one of the most widely used clustering algorithms there is and applied for a wide range of settings, but how do we know that a data set is suited for it? K-means's assumptions about the data are relatively strict: clusters should be Gaussian distributed with uniform variance in all directions. These assumptions are rarely satisfied in a data set. While clusters that do not deviate from these assumptions too far, can be cut out with sufficient precision, the farther the data is from these assumptions, the more likely k-means is to fail. Instead of testing whether the assumptions are met and k-means can be applied, we make it so. Our goal is to improve the suitability of data sets for k-means and widen the range of possible data sets it can be applied to. Our algorithm changes the position of data points so that the clusters become more compact and, thus, fit better into the requirements of k-means. Based on cluster-wise PCA and local Z-transformation we estimate the form of the correct clusters and move the data points so that the correct clusters become more compact with each iteration and – in the end – have uniform variance, as well as increase the distance between clusters. We explain the theory behind our approach and validate it with extensive experiments on various real world data sets.

1 INTRODUCTION

K-means [18] is one if not the most widely used clustering algorithm there is. It is being studied intensively in scientific circles, with dozens of articles about every aspect of it. It is used from speech recognition [5, 17] to autonomous driving [28] and is frequently an important building block for a more extensive system as k-means is simple, fast and often gives good results. It is used in combination with classical Clustering methods, e.g. as an initialisation for EM [7], as well as for new neural networks-based methods, e.g. DEC [29].

In principle, k-means requires clusters that have a Gaussian bubble shape with equal variance in all directions (we shorten this to "uniform variance"), but it can handle many forms of clusters as long as they are mostly convex, non-overlapping and somewhat well separated. K-means partitions the data into Voronoi cells (see [9] for a short introduction; an example can be seen in Fig. 1b). The closer the clusters match the assumptions of Gaussian shape with uniform variance while being well separated, the better they fit into this Voronoi cell structure of k-means. Compact clusters are more suited for these cells, while non-convex clusters might not fit at all. Thus, data sets are often preprocessed by normalizing them. Normalizing in the [0,1]-range or Z-transformation prevents clusters from being extremely stretched in one direction and very contracted in another. The clusters are therefore in a more compact shape, which makes them easier identifiable as they fit better into the Voronoi cells. Normalization is often recommended for k-means [21] and other clustering approaches [14].



(a) Our running example and the ground truth

(b) How k-means clusters it with a good initialisation.

(c) How we manage to change the data set. Clusters are now almost perfectly Gaussian.

Figure 1: The running example, how k-means clusters it and how it looks after PCE (our method).

The running example shown in Fig. 1a is simple and would be well suited for k-means, as the clusters follow a Gaussian distribution and are well-separated. The spread of the clusters, though, makes it difficult for k-means to partition it well. A typical clustering result of k-means, if the initialisation goes well, is shown in Fig. 1b. Clusters are cut into multiple parts and incorrectly merged. Even if the initialization strategy succeeds, i.e. returns one data point from each cluster as the start initialization for the centers, k-means performs poorly. The clusters do not fit into the Voronoi cell structure of k-means. Merely normalizing the data is not enough to make it suited for k-means. Besides the normalization of data, the most commonly used approaches that can be considered transformations are PCA [10] and ICA [6]. However, both are linear and as can be seen in Fig. 1a the clusters cannot be separated linearly (see Table 1 that they have almost no effect on the running example). Non-linear transformations are rare. Most, like DEC and IDEC [12], are based in Neural Networks. These could in principle separate the clusters, but often

[1] Faculty of Computer Science, University of Vienna, Vienna, Austria
[2] ds:Univie, University of Vienna, Vienna, Austria

proceed rather rough and tear the data set apart (see Section 4), making the clusters unlikely to fit into the Voronoi cell structure.

The method that we present takes the (preliminary) clusters found by k-means (i.e. the Voronoi cells) and shifts the data points depending on the shape and spread of them. It forces very elongated clusters into a more compact form with uniform variance while separating the clusters better from each other. It does so while still keeping the basic shape and form of the data set intact (see Section 4). A successful application for the running example is shown in Fig. 1c. The clusters fit now seamlessly into the Voronoi cell structure of k-means, which can perfectly cluster the data. Thus, we extend the spectrum of data sets suited for k-means. We intend to demonstrate that this increase in suitability is substantial enough, that the combination of our transformation with k-means can now outperform a wide range of various clustering algorithms. With this, we intend to forestall the retort "Why do you not simply use another algorithm besides k-means?" which also ignores how wide-spread k-means has become in various fields less familiar with various clustering approaches.

1.1 Contributions

The method presented in this paper, PCE (Principal Cluster Enhancement) is a **non-linear** transformation, making the clusters better suited for k-means.

- The focus is on k-means, as we explicitly try to make the clusters more suited to it, but many methods might benefit from more compact clusters. We include experiments for different standard clustering methods and demonstrate that they are also compatible with our method.
- We demonstrate that PCE keeps the basic shape intact, transforms the data non-linear while making clusters more compact.
- PCE and the reasons for its decisions are easy to understand. Some methods, e.g. Neural Network-based ones, are often hard to understand and resemble black boxes in this regard.
- The procedure is deterministic, excluding the initialization of k-means. With a deterministic initialisation for k-means it would be completely deterministic.
- Our method is light in runtime-requirement. The major part is executing PCA, a technique heavily researched and optimized.

1.2 Related Work

PCE is an algorithm that tries to lessen the assumptions of k-means and broaden the range of data sets suited for it. In regards to being a support-method for k-means, it falls into a long line of algorithms, from which X-means [22], to estimate the number of clusters, or k-means++ [1], to find a good initialisation, are most likely the best known. K-means has, over time, become a sort of framework which is freely adapted to suit the needs of various approaches. Examples are e.g. FOSSCLU [11] or SubKMeans [20], that look for subspaces compatible with k-means.

Closely related to us in terms of the pursued goal is DipTransformation [23] that has the explicit goal of changing the structure of the data set to improve clustering. It also transforms the data, but it is restricted to linear transformations like PCA and ICA. A basic version of DipTransformation is DipScaling [24], which scales the axes according to the information given by the Dip-Test [15]. It falls into the category of very simple normalization-methods like Z-transformation (also referred to as Z-normalization) and the normalization into the $[0, 1]$-range, which are the two fundamental transformation-methods which are often used for pre-processing. Transformation methods per se are rare. Apart from those mentioned, one of the closer members of the clustering community is SynC [2], which collapses multiple data points onto a single data points. One could further include Spectral Clustering-methods, that create a distance matrix from the data, which is after some steps clustered with partitioning methods like k-means. These could be considered as a pre-processing step for k-means. Kernel-based approaches [25] do not change the data set, but interpret distances differently, which is a similar concept. Feature Weighting-method like EWKM [16] also do not change the data points but put a different relevance on each feature for clustering. Closer are Neural Network-based methods like DEC, IDEC and DCN [30] that combine k-means and Deep Learning, which actually change the data set.

Our approach makes use of the Dip-Test [15], a statistical test for multi-modality, which has lately garnered some attention in the Data Mining-Community (see [19] for an introduction). The first method was DipMeans [4], a method from the k-means framework, that estimates the number of clusters in a data set. There is also SkinnyDip, a method to cluster in the presence of high noise, and the aforementioned DipTransformation and DipScaling. The Dip-Test has also recently been generalized from one dimension to multiple dimensions [26] and employed as a test, if clustering makes sense, i.e. if multiple clusters are present in the data set.

2 THE ALGORITHM

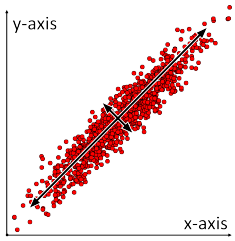
2.1 The principle: PCA and Z-transformation

We aim to change the data set so that clusters become more compact and thus easier to cluster. The basis, which we make heavy use of, is Z-transformation on one hand and PCA on the other. Z-transformation is used to obtain clusters with uniform variance. Z-transformation divides a feature by its standard deviation-value, which brings this feature to a variance of 1. It can be expressed as

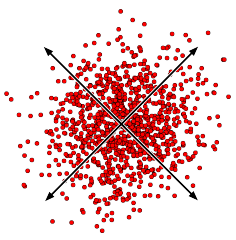
$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

with μ as the mean of the feature, x a data point and the variance σ^2, which means the standard deviation is σ.

The data set shown in Fig. 2a has the same variance in the x- and y-axis. Only applying Z-transformation will not change the shape of the cluster, i.e. it will not have uniform variance. For this we need PCA. PCA gives the directions with the highest variance. If we apply the Z-transformation along these directions we get Fig. 2b. The cluster is now of uniform variance and would be perfectly suited for k-means.



(a) A simple stretched cluster and its main components.



(b) The cluster is forced into a uniformly gaussian shape.

Figure 2: For a simple cluster, the main components of PCA and their variance is computed (shown with the length of the main components). Performing Z-Transformation along these directions leads to a shape with uniform variance.

2.2 Multiple clusters caught in a Voronoi cell

More than one cluster complicates the matter, as several clusters can be caught in a Voronoi cell. Consider the simple data set shown in Fig. 3a with two stretched clusters very close to each other, similar to two of the clusters from the running example. To the human eye, the correct clustering is easy to see, but k-means will converge to the sub-par clustering shown there with two clusters in a cell. The application of the same PCA/Z-transformation combination as before does not change the data set sufficiently to escape this state. We wish to change the data set so that the clusters are clearly separated and k-means can easily cluster it. For this, we need to adapt our approach by adjusting how we use the standard deviation.

The projections of the data points in the Voronoi cell to the main components of PCA is shown in Fig. 3b. These projections contain very different levels of cluster-information. We can see that in one of the projections we actually "caught" two clusters instead of one. Shown there is also the "strength" of the directions, i.e. the standard deviation/variance, of the main components as the length of the arrows. The cluster has roughly the same σ in both directions and thus applying the former approach does not change the data set enough for k-means to escape the local optima shown here. The goal is to push the clusters so far apart so that k-means will move out of this optima and distinguish between the clusters correctly. This can be done by "changing" σ. If σ becomes smaller, if more than one cluster is found in a direction, it will push the clusters farther away from each other and k-means can escape the local optima it is in and converge to a better state. Effectively, we change the probability landscape of k-means. We make some optima more likely while deterring k-means from others.

To test whether there is more than one cluster in such a direction, we use the Dip-test. The Dip-test shows us how likely such a projection is uni-modal, i.e. if more than one cluster has been caught. The probability found by the Dip-Test, p_d, is used to adapt the standard deviation σ. The formula we use is given by Eq. (2).

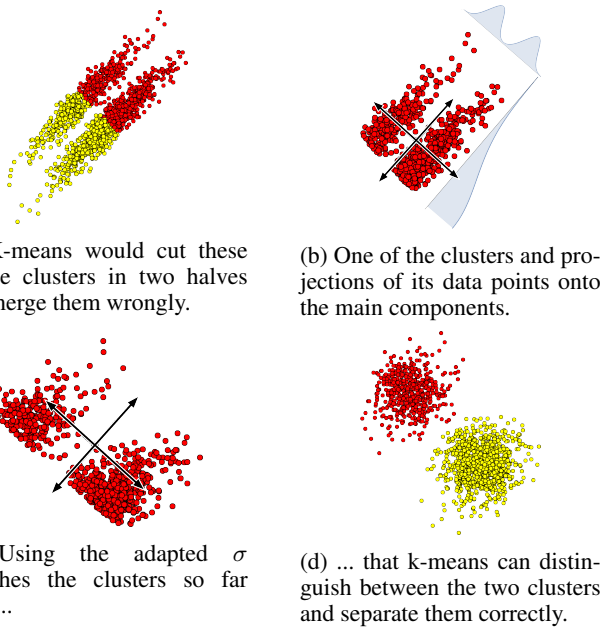
$$\bar{\sigma} = \sigma + p_d \cdot \sigma \tag{2}$$

This doubles the standard deviation for a uni-modal direction, while a multi-modal direction is kept the same. The effect is shown in Fig. 3. In Fig. 3b the standard deviations of the projections are in an equilibrium. Applying the earlier transformation does not change the data set. Using $\bar{\sigma}$ interprets the need to change the data set very differently. The uni-modal direction is contracted much more compared to before, as $\bar{\sigma}$ is twice as large as σ. This leads to Fig. 3c. The clusters are now so far apart, that k-means can distinguish between them and the clusters are correctly clustered. Using $\bar{\sigma}$ allows for clusters to become better separated from each other in the data. For a single cluster like the one shown in Fig. 2 nothing much changes, but for a multi-cluster Voronoi cell like the one in Fig. 3 our transformation starts pushing clusters away from each other, reducing the difficulty for k-means to cluster the data correctly.

We use the Dip-test as a measure for the possibility of multiple clusters, as it is rather precise and has a good runtime of $\mathcal{O}(n \log(n))$ (n the number of data points tested). We increase $\bar{\sigma}$ linearly in Eq. (2) from σ to 2σ with the likelihood of uni-modality, because, following Occam's Razor, it is the least complex approach.

2.3 Interaction of Voronoi cell-Transformations

There are two things we need to consider: 1) The preliminary clusters found by k-means are most likely flawed. We have shown how we intend to deal with this. 2) We cannot analyse a cell by itself. We focus on this now. Consider two data points very close to each other but in different Voronoi cells. If we apply the combination of Z-transformation/PCA only on the data points in a single Voronoi cell, the two data points might end up far apart from each other. This would lead to large gaps in the data and might even rip apart a correct cluster because it is part of two different Voronoi cells. Hence, we cannot analyse a cell alone and need a way how a transformation of one Voronoi cell affects another cell.



(a) K-means would cut these simple clusters in two halves and merge them wrongly.

(b) One of the clusters and projections of its data points onto the main components.

(c) Using the adapted σ stretches the clusters so far apart...

(d) ... that k-means can distinguish between the two clusters and separate them correctly.

Figure 3: How the adapted σ helps with distinguishing between clusters.

Applying the transformation of one cell to all data points in all cells is possible. However, this would lead to clusters no longer being able to obtain uniform variance, since the direction in which cells contract/expand can conflict with each other. Thus, we need to lessen the impact of the transformation with distance to the Voronoi cell. This way clusters can be re-shaped into uniform variance, but not deter the transformation of a different cell. If the impact is lessened continuously, close data points with different cell-assignments would stay close and transformation would not cause rips in the data.

We now need to cover some basics. The formula for the Z-transformation is given by Eq. (1). We want to keep the cells roughly where they are, so we need to move a data point back to where it came from; thus, we add μ, the mean of the feature, to the data point.

$$x' = \frac{x - \mu}{\bar{\sigma}} + \mu \tag{3}$$

This essentially just expands/contracts the data points in a specific direction, but leaves the center of the data points where it is. In our setting, μ is equivalent to the projection of the cluster-center as found by k-means onto the principal component found by PCA, because we wish to scale the clusters in the specific directions of PCA, but not to move the position of the clusters. The intention is to force the clusters into a shape with uniform variance, but not necessarily to change their position.

Our main interest is the effect of the transformation depending on the distance to the center of the cluster. Thus, we compute how much a data point is moved, depending on the distance to the center.

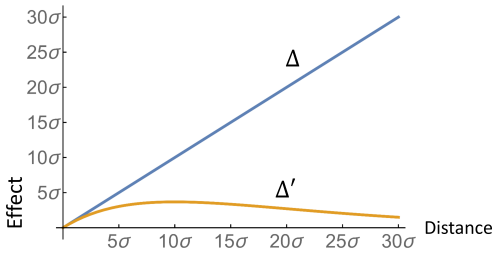


Figure 4: The effect of Δ and Δ' depending on the distance to the center of the cluster. The effect of Δ' is very similar to Δ up to a distance of $\approx 2\ \bar{\sigma}$ from μ, but the farther a data point is from the center of the Voronoi cell, the smaller the effect of the cell becomes.

$$\Delta = x' - x = \frac{x-\mu}{\bar{\sigma}} + \mu - x = \\ = (x-\mu)\left(\frac{1}{\bar{\sigma}} - 1\right) = d(x,\mu)\left(\frac{1}{\bar{\sigma}} - 1\right) \tag{4}$$

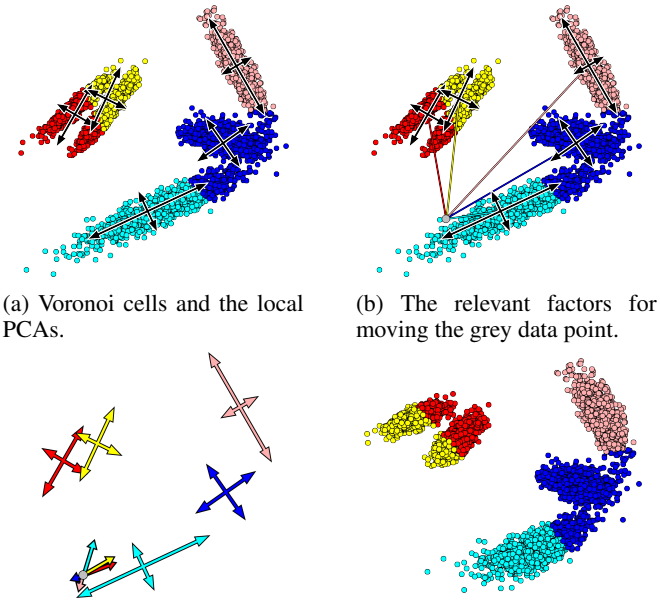
Δ is the difference between the old and the new position of the data point. $d(x,\mu)$ is the distance of x from μ. We see that the change of the data point is linearly dependent on the distance of the data point to the center of the transformation. We want to weaken this effect with distance. Instead of a linear, the effect should increase continuously and sub-linear, eventually declining to 0 with distance to the center. Discontinuity would mean rips in the data set. A straightforward solution, which worked well in the experiments, is to add a factor to Eq. (4) that ensures that the effect declines with distance, i.e. we add an exponential decay factor to it.

$$\Delta' = \frac{d(\mu,x)}{e^{\frac{d(\mu,x)}{(10\bar{\sigma})}}}\left(\frac{1}{\bar{\sigma}} - 1\right) \tag{5}$$

With the added decay factor, the influence of the transformation slows continuously with distance to the center of the Voronoi cell. The explicit form can be seen in Fig. 4. Close to the center of the cell, roughly in an area of $2\bar{\sigma}$, which is about 95% in a Gaussian distribution, the transformation behaves just like a normal Z-transformation. This effect diminishes with distance to the center and starting from a distance of $10\bar{\sigma}$ the effect becomes smaller and smaller in contrast to the effect given by Eq. (4). Clusters with different orientations can now obtain uniform variance.

We noticed that sometimes if multiple clusters are in a Voronoi cell the algorithm produces a sort of oscillating effect in the transformation. This is most likely due to the adapted standard deviation in Eq. (2). To avoid this we slow the transformation down. We doubled the speed in Eq. (2), so now we halve it and only apply $\frac{\Delta'}{2}$.

The factor $(\frac{1}{\bar{\sigma}} - 1)$, which has not been altered from Eq. (4) to Eq. (5), is basically the "sign" of the transformation. It determines if the data is stretched ($\bar{\sigma} < 1$), contracted ($\bar{\sigma} > 1$) or is left as is ($\bar{\sigma} = 1$), i.e. the data is already uniform in this direction. The ultimate goal is to bring all cluster in all directions to a uniform variance. This can be described as $\sum_{j=1}^{k}\sum_{l=1}^{d}|\bar{\sigma}_{lj} - c| = 0$, with $\bar{\sigma}_{lj}$ being the adapted variance in direction l for cell j and c a constant. This formula is the intrinsic objective function of PCE. It formalizes the intuition that all clusters should be brought to uniform variance. If the objective function is 0, PCE would stop, as no more change in the position of the data points is necessary. The uniform variance does not need to be 1. Any constant value c would be suitable, as we merely try to obtain the same variance in all directions, not necessarily $\bar{\sigma}_{lj} = 1$.



(a) Voronoi cells and the local PCAs.

(b) The relevant factors for moving the grey data point.

(c) The effect of each cluster on the grey data point.

(d) Executing the change in position. The first iteration.

Figure 5: Applying the algorithm - the first iteration.

Algorithm 1 PCE

Require: Data D, number of clusters k
1: **procedure** PCE(D,k)
2: **while** Cluster assignments change **do**
3: Execute k-means on D
4: **for** j = 1,...,k **do**
5: Compute local PCA for Cluster j
6: Compute variance for main components of PCA
7: **end for**
8: **for** i = 1,...,n **do**
9: **for** j = 1,...,k **do**
10: Compute effect of Z-transformation along the direction of principal components of PCA from Cluster j on data point i with Eq. (5).
11: **end for**
12: **end for**
13: **for** i = 1,...,n **do**
14: Sum up the effects of Z-transformation/PCA from all Clusters on data point i as computed in Line 10 and execute it.
15: **end for**
16: **end while**
17: **return** D
18: **end procedure**

We illustrate the approach of PCE by going through the pseudo code with the running example. After executing k-means (Line 3 in the Pseudo-code in Algorithm 1) it is easy to see that clusters and Voronoi cells do not match (Fig. 5a). We use random initialisation (see Section 4). PCA is computed in all Voronoi cells (Line 5) and the effects of these local PCAs on the data points are computed (Line 10). We take the grey data point as an example. All relevant information for it is shown in Fig. 5b. To compute the effects of the clusters on it, we first project it onto the main components found by PCA and compute the distance to the centers. These distances give us with Eq.

5 the effect of the cells on the grey data point. In Fig. 5c we can see how the cells influence the grey data point, i.e. how they move it. Adding these changes up results in the overall effect on the grey data point. Computing these effects for all data points (Line 9-12) and executing them (Line 13-15) leads to Fig. 5d. The first iteration of PCE. The clusters are now closer to the Gaussian bubble shape with uniform variance they ideally have. The algorithm continues as one iteration might not make the clusters perfectly uniform in variance. K-means is updated, i.e. executed with the same centers, which have also been moved. Every iteration brings the clusters closer to uniform variance, and, eventually, the cluster assignments do not change any more. Identical cluster assignments in two successive iterations are interpreted as a stable configuration (Line 2/16), i.e. the clusters have become of such uniform variance, that another iteration would not improve the data any more. In the final result (Fig. 1c) the clusters are perfectly compact and suited for k-means.

2.4 Runtime

Following the pseudo code given in Alg. 1 we can estimate the runtime of the algorithm to be:
$\mathcal{O}(i \cdot (n \cdot k \cdot d + k(\mathcal{O}(PCA) + n \cdot log(n) \cdot d) + n \cdot k \cdot d + n))$
with i the number of iterations, d the dimensionality of the data set, k the number of clusters and n the number of data points. We left PCA as its own factor as it is by far the largest part in regards of runtime. PCA is $\mathcal{O}(d^3 + d^2 \cdot n)$, thus summed up the runtime is

$$\mathcal{O}(i \cdot k \cdot d^3 \cdot n \cdot log(n))$$

The cubic runtime in d seems excessive, but it is a surprisingly small issue as this is caused solely by PCA and the runtime would be linear without it. PCA is a standard data mining algorithm for which highly optimized and parallelized implementations exist. High-level APIs and distributed computing frameworks perform PCA with impressive speed and even a matrix with millions of entries can be decomposed in mere seconds [3]. There exist also approximative algorithms for PCA [13] reducing the runtime to $O(d^2\ log(d))$. The implementation of PCA in the commonly used scikit-learn package uses this implementation. So, while PCA is definitely the bottleneck for our algorithm, the cubic estimation is essentially a worst-case scenario, which is only encountered for the naïve implementation of PCA and, thus, easily avoided.

3 EXPERIMENTAL EVALUATION

Our goal is to make data sets more suited for k-means by making the clusters more compact. The direct result of this should be a notable increase in clustering quality when comparing k-means before and after PCE. Furthermore, this increase should be large enough for k-means to beat a wide range of various clustering algorithms. Thus, we tested our method on various publicly available real world data sets from the UCI repository [8] and compared with classical clustering method, like DBSCAN and SingleLink, as well as state-of-the-art approaches. The results can be seen in Table 1.

We measured clustering quality in Normalized Mutual Information (NMI) [27], which is currently widely used to estimate the success of a clustering method. NMI scales between 1.0 (perfect clustering result) and 0.0 (purely random cluster assignments).

On average, k-means improves by 0.11 in NMI. This is the notable increase we were looking for and it made k-means the best-performing method on the data sets. It is now outperforming a wide spectrum of clustering approaches. PCE managed to make the clusters better suited to k-means and to re-shape the clusters so that they fit into the Voronoi cell structure of k-means. In Fig. 1c we saw how well the clusters became suited for k-means and in the experiments we see that the effect of improving k-means is not limited to synthetic data.

Following the argumentation in [23] that a method that improves k-means will most likely also improve other methods, we also tested our approach with other standard clustering methods. In Table 2 we choose 4 of the data sets used in Table 1 and 4 of the standard clustering approaches, i.e. EM, SingleLink, DBSCAN, Spectral Clustering as well as k-means++ and tested by how much PCE could improve their clustering results. We also included other transformations and tested how much they could improve the clustering results. PCE is used here as a pre-processing step, which explains the choice of the other methods. We used here k-means++ instead of k-means, but their results barely differ (see Section 4).

From the 20 comparisons in Table 2 testing data sets and methods, PCE was the best choice in 18. An overview of these results for all of the 7 real-world data sets for these methods can be seen in Table 3. This is basically an abridged version of Table 2. The conclusion is the same: The quality of clustering could be notably improved. PCE could enhance the shape of the clusters, such that the data sets became easier to cluster. It is the best pre-processing step here and highly compatible with all of the 4 classical methods. EM can, to a degree, counteract stretched cluster, thus PCE is more effective on some data sets for it and has less influence on others. Interestingly, k-means performs better than EM and has on average a lead of 0.04 over EM. We wish to point out, that Spectral Clustering and SingleLink improve by 0.12 and 0.13 respectively (more than k-means).

In consideration of all these experiments, we can state that we succeeded in making these data sets easier accessible to clustering methods by moving data points so that clusters become more compact and easier to find.

3.1 Parameters

Data sets were normalized in the [0,1]-range for all methods. We tried to be as fair as possible to all comparison methods. Transformation methods like PCA are all shown in combination with k-means. Methods like k-means++, which entail random aspects, have been executed 100 times and the average NMI is given. The correct number of clusters, which e.g. EM needs, is always given to the algorithm. EM is initialized with k-means as that lead to better results. DBSCAN is difficult to parametrize and thus, we follow the lead from [23]. We computed the average distance between data points, a, and tested every combination of $minPts \in \{1, 2, 3, 5, 10, 50\}$ and $\epsilon \in \{0.05 \cdot a, 0.1 \cdot a, 0.2 \cdot a, 0.4 \cdot a, 0.6 \cdot a, 0.8 \cdot a, a\}$. Only the best NMI is reported. For PCA and ICA we followed the lead in [20]. Thus, for PCA we use the often-applied setting that 90% of the variance is kept and for ICA the number of dimensions is the number of clusters. We also tried keeping all dimensions, but this did not change the message of the reported results (PCE performed still better on all data sets). These results are not included due to space-restrictions. For SingleLink we found that CompleteLinkage lead to clear better results compare to SingleLinkage and is thus used here as the cluster creation criterion. EWKM has a parameter λ which should be chosen in the range $[1, 3]$. We tried for each run $\lambda \in \{1, 2, 3\}$ and took the best result. For kernel k-means we tried Gaussian and Polynomial kernels, as they are two of the more popular choices. DEC and IDEC are difficult to parametrize. We pretrained ten autoencoders for each

Table 1: Comparison between methods measured in NMI. For non-deterministic methods is the average of 100 runs given. The correct number of clusters is given for every methods that can use it. Parameters and technical details in Section 3.1. Best result shown in bold.

Data set	Iris	Vertebrae C.	Seeds	Wifi-local.	Breast Cancer	Breast Tissue	Wine	Run. Ex.
PCE	0.84	**0.52**	**0.82**	**0.91**	**0.81**	**0.55**	**0.88**	**0.93**
k-means	0.71	0.26	0.67	0.82	0.74	0.50	0.84	0.76
DipTransformation	0.84	0.29	0.78	0.65	0.72	0.51	0.71	0.67
DipScaling	0.81	0.27	0.73	0.69	0.73	0.52	0.72	0.61
Z-Transformation	0.64	0.30	0.74	0.80	0.73	0.49	0.86	0.76
PCA	0.74	0.27	0.66	0.83	0.74	0.46	0.85	0.75
ICA	0.57	0.27	0.63	0.61	0.65	0.43	0.76	0.75
k-means++	0.72	0.26	0.67	0.81	0.74	0.49	0.84	0.77
Spectral	0.59	0.27	0.60	0.77	0.79	0.49	0.87	0.89
STSC	0.58	0.27	0.53	0.84	0.36	0.30	0.86	0.42
IDEC	0.25	0.11	0.29	0.40	0.17	0.30	0.22	0.46
DEC	0.24	0.11	0.28	0.43	0.17	0.30	0.23	0.47
DipMeans	0.58	0.00	0.44	0.84	0.29	0.00	0.00	0.76
SynC	0.58	0.13	0.48	0.80	0.64	0.29	0.59	0.48
DBSCAN	0.61	0.21	0.42	0.49	0.76	0.44	0.42	0.88
SingleLink	0.74	0.17	0.61	0.49	0.48	0.39	0.78	0.67
EM	**0.87**	0.49	0.64	0.90	0.54	0.48	0.86	0.89
FossClu	0.73	0.07	0.57	0.87	0.43	0.36	0.61	—
SubKMeans	0.66	0.30	0.73	0.80	0.73	0.45	0.87	0.77
kernel k-m. (Gauss.)	0.66	0.27	0.67	0.80	0.40	0.44	0.83	0.78
kernel k-m. (Polyn.)	0.69	0.25	0.63	0.69	0.75	0.49	0.74	0.69
EWKM	0.67	0.25	0.60	0.71	0.14	0.49	0.58	0.73

Table 2: 4 of the real-world data sets from Table 1 and we show how PCE fares on them in combination with clustering methods besides k-means in comparison with the most closely related methods.

EM	Vertebrae.	Wifi-local.	BreastTissue	Wine
PCE	**0.53**	0.89	**0.52**	**0.88**
orig	0.49	0.90	0.48	0.87
PCA	0.27	0.90	0.49	**0.88**
ICA	0.29	0.82	0.41	0.84
DipTrans.	0.15	0.90	0.48	0.83
Z-trans	0.49	**0.92**	0.47	0.85
SingleLink				
PCE	**0.45**	**0.87**	**0.50**	**0.83**
orig	0.17	0.49	0.39	0.78
PCA	0.17	0.64	0.41	0.47
ICA	0.03	0.13	0.23	0.02
DipTrans.	0.16	0.71	0.36	0.39
Z-trans	0.02	0.40	0.38	0.61
DBSCAN				
PCE	**0.33**	0.44	**0.52**	**0.55**
orig	0.21	**0.49**	0.44	0.42
PCA	0.18	0.40	0.45	0.50
ICA	0.18	0.36	0.40	0.46
DipTrans.	0.18	0.47	0.46	0.52
Z-trans	0.26	0.46	0.44	0.43
Spectral				
PCE	**0.49**	**0.85**	**0.52**	**0.88**
orig	0.27	0.76	0.50	0.87
PCA	0.26	0.75	0.50	0.75
ICA	0.28	0.63	0.45	0.84
DipTrans.	0.21	0.76	0.49	0.83
Z-trans	0.32	0.74	0.50	**0.88**
k-means++				
PCE	**0.50**	**0.91**	**0.53**	**0.87**
orig	0.27	0.84	0.50	0.84
PCA	0.27	0.83	0.46	0.84
ICA	0.28	0.59	0.43	0.85
DipTrans.	0.24	0.78	0.51	0.75
Z-trans	0.30	0.80	0.46	**0.87**

Table 3: Average NMI of standard clustering methods on the 7 real world data sets from Table 1.

	EM	SingleL.	Spectral	DBSCAN	k-means++
PCE	**0.71**	**0.65**	**0.75**	**0.51**	**0.75**
orig.	0.68	0.52	0.63	0.48	0.66
PCA	0.61	0.50	0.52	0.48	0.65
ICA	0.59	0.12	0.60	0.43	0.59
DipTrans.	0.62	0.50	0.64	0.50	0.65
Z-trans.	0.67	0.45	0.63	0.47	0.65

data set as described in [29] and setting the latent dimension equal to the number of clusters. DEC and IDEC were each run on the ten pretrained autoencoders for 200 epochs, which ensured convergence.

4 DISCUSSION AND CONCLUSION

In the following, we discuss point by point some of the aspects of PCE and its relation to other methods.

Source code: Source code, data sets and labels can be found here: `https://dm.cs.univie.ac.at/research/downloads/`

Continuity: The effect of a single direction in a cell on a data point is given with Eq. 5. Summarizing these effects for all k Voronoi cells and d directions gives:

$$PCE_i(x) = \sum_{j=1}^{k} \sum_{l=1}^{d} \left(x + \frac{1}{2} \Delta'(d(\mu_j, x)_l, \bar{\sigma}_{lj}) \right)$$

where $d(\mu_j, x)_l$ is the distance of the projection of x onto the lth direction of PCA and $\bar{\sigma}_{lj}$ the adapted variance in this direction. This is the formula for a single iteration of PCE. It is easy to see from it that PCE is continuous, i.e. data points that are close before will be close afterwards.

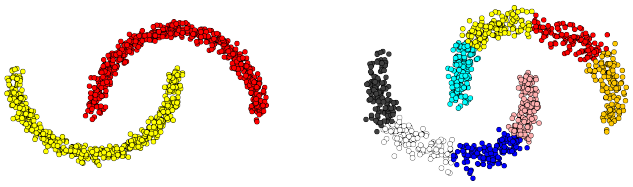


Figure 6: An unfitting data set with a wrong number of clusters for PCE. Orig. data on the left, PCE-result on the right.

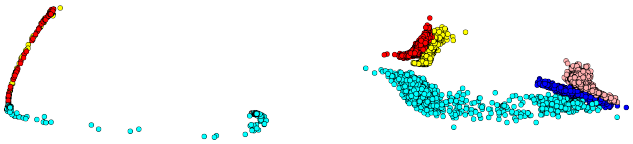


Figure 7: The running example with DEC and IDEC. Rotated by 90°. Plotted with groundtruth-label to help recognition.

Theorem 1 *PCE is continuous.*

Proof: Let $y = x + \epsilon$ with $x, y, \epsilon \in \mathbb{R}^d$.
$|PCE_i(y) - PCE_i(x)| = |PCE_i(x+\epsilon) - PCE_i(x)|$
$\sum_{j=1}^{k}\sum_{l=1}^{d}\left(\epsilon + \frac{1}{2}\Delta'(d(\mu_j, x+\epsilon)_l, \bar{\sigma}_{lj}) - \frac{1}{2}\Delta'(d(\mu_j, x)_l, \bar{\sigma}_{lj})\right)$
Since Δ' is continuous itself, $|PCE_i(y) - PCE_i(x)| \rightarrow 0$ for $\epsilon \rightarrow 0$ and thus a single iteration is continuous. As the concatenation of continuous functions is again continuous, PCE is continuous. ■

This proof is valid for a wide range of possible Δ'-functions. Our Δ' is the standard Z-transformation with an adapted variance and an exponential decay factor. In the proof, we only used that our Δ' is continuous. Thus, this proof also holds e.g. for a logarithmic or polynomial decay factor or a completely different definition of Δ', as long as it is continuous.

Continuity is a crucial feature for a transformation as it ensures that the local neighbourhood is kept the same in at least an approximative fashion. Distances between data points change, but if data points are close, they will still be close after the transformation. As a practical example, consider the data shown in Fig. 6. It is unsuited to PCE as its clusters are too far from a convex shape. PCE is not capable of separating the clusters, but it will not cause any "damage" (we used a wrong number of clusters, but similar behaviour could also be observed if the number of clusters is given as 2, 3, . . .). The continuity and that PCE merely stretches and contracts data points causes a very careful transformation, which 1) does not produce rips in the data, 2) keeps the basic shape intact and 3) is unlikely to transform the shape if not enough information is available, i.e. if the measured variance/dip-values are not deviating relevantly from each other. If this is the case, PCE is unsuited for the data set; resulting in PCE refraining from doing anything relevant. Fig. 6 depicts such a case. PCE is unsuited for the data set, but it does not change much and the data set becomes no more difficult to cluster. PCE is "careful", if not enough information is present for it.

Contrary to PCE, Deep Learning-based methods are far more extreme in their transformation approach. In Fig. 7 is shown how DEC/IDEC transform the running example. The running example is a rather simple data set, but DEC/IDEC restructure it completely. Neural Networks often seem like a black box, where many decisions are barely comprehensible. This is such a case. The basic shape is heavily distorted and the structure of the clusters now unrecognisable. The advantage of PCE over Deep Learning-based methods, which are almost the only other non-linear transformations there are, is 1) its approach and decision-making is easy to understand 2) it is far more conservative and refrains from destroying structure.

Non-convex clusters: We have seen in Fig. 6 that PCE cannot handle all types of data sets. If the clusters are too far from a convex shape or are massively overlapping, PCE will have problems. Though, it will most likely refrain from acting in such a case, which does not deterriorate things. It might be possible to adapt PCE towards these cases by employing other ways of estimating the local shape of data points instead of PCA. We intend to analyze this in future works.

The effect of Initialisation: PCE, as well as k-means or EM, is deterministic after the initialisation has been decided. This raises the question of the effect the initialisation has. The two main initialisation methods for k-means are random initialisation (RI), where the centers are assigned a random data point, which is used in this paper, and k-means++. K-means++ is often the gold-standard and improves over RI, but this is not the case here. The average difference between RI to k-means++ on these data sets is for k-means merely 0.005 in NMI, with k-means++ being slightly **worse**. The same effect can be seen with EM (0.006 in NMI) and PCE (0.006 in NMI). That is not to say that initialisation is not a relevant factor. Taking the best of 10 runs of k-means (according to the objective function of k-means) leads to an average improvement of 0.011 in NMI on the real world data sets for k-means evenly distributed on all data sets. The same strategy improves EM by about 0.005 and PCE even by 0.019 in NMI.

On the tested data sets, PCE could notably improve the results. This means, that it is rather likely that there are stretched clusters in the data, comparable to the running example. K-means++ chooses new centers based on distances, which means that it might choose two data points from such a stretched cluster, or give two close clusters like the yellow/red one in the running example only one starting center. RI does not take distances into account, which means that the stretch of the clusters makes no difference, whether a cluster gets a starting center. Thus, on these types of data sets k-means++ is not per se the best choice of initialisation strategy.

Regarding EM and objective functions: One variation of the retort mentioned in the introduction is "Why not use EM instead?" EM can, to a degree, take care of stretched clusters and thus, overcome the Voronoi cell structure of k-means. This ignores, that EM is a clustering method, while PCE is a transformation approach, that can be used as a pre-processing step for clustering methods. It gives EMs ability to handle stretched clusters to methods like k-means or SingleLink. Also, PCE improves k-means so far, that it is on average better by 0.10 in NMI compared to EM.

Furthermore, PCE can also improve EM. PCE changes the position of data points, which changes the optima towards which an algorithm can converge to. Thus, the loss landscape of the objective function itself is changed. This opens stable configurations which the algorithm could not reach before. The Seeds-data set, for example, improves for EM from 0.64 to 0.78 in NMI. We intend to analyse this change in local optima more thoroughly in future works.

Table 4: The NMI-values for the running example for k-means before and after PCE for wrong values of k. Better result bold.

	k=2	k=3	k=4	k=6	k=7	k=8	k=9	k=10
k-m.	0.35	0.62	0.71	0.76	0.73	0.70	0.68	0.66
PCE	**0.36**	**0.63**	**0.81**	**0.89**	**0.84**	**0.79**	**0.75**	**0.71**

Wrong k: What happens if k is wrong? We tested the running example for wrong values of k and could still observe an improvement (Table 4). The same holds for the real-world data sets. We set $k \pm 1$ its real value. In both cases, we still got an increase in average NMI.

PCE optima: Different optima of k-means can lead to different final transformations, and similar k-means optima can lead PCE to similar final transformations. Sometimes, though, also different optima will lead with PCE to similar final transformations. Our working hypothesis is that PCE has – similar to k-means – various stable states to which it can converge to, satisfying the objective function, i.e. having uniform variance in all directions. This is a question which we will analyze in more detail in the future.

Conclusion: The usual approach in Data Mining is to find a clustering method for a data set and, if none fits, to create a new approach. PCE is the other way around; if the data set does not fit, we make it fit into the assumptions of the clustering method. We devised a method that iteratively re-shapes the clusters, moves them further apart from each other and makes them more compact by forcing them into a shape with uniform variance. We tested PCE with extensive experiments and showed that it also holds up under real-world conditions, where clusters are usually messier than in synthetic examples. It improved not only k-means but also the standard clustering methods. Since they approach clustering in very different ways, we assume that a wide range of algorithms could benefit from PCE.

REFERENCES

[1] David Arthur and Sergei Vassilvitskii, 'K-means++: The advantages of careful seeding', in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, Philadelphia, PA, USA, (2007). Society for Industrial and Applied Mathematics.

[2] Christian Böhm, Claudia Plant, Junming Shao, and Qinli Yang, 'Clustering by synchronization', in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pp. 583–592, New York, NY, USA, (2010). ACM.

[3] Reza Bosagh Zadeh, Xiangrui Meng, Alexander Ulanov, Burak Yavuz, Li Pu, Shivaram Venkataraman, Evan Sparks, Aaron Staple, and Matei Zaharia, 'Matrix computations and optimization in apache spark', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 31–38. ACM, (2016).

[4] Theofilos Chamalis and Aristidis Likas, 'The projected dip-means clustering algorithm', in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN '18, pp. 14:1–14:7, New York, NY, USA, (2018). ACM.

[5] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass, 'Unsupervised cross-modal alignment of speech and text embedding spaces', in *Advances in Neural Information Processing Systems 31*, eds., S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 7354–7364. Curran Associates, Inc., (2018).

[6] Pierre Comon, 'Independent component analysis, a new concept?', *Signal Processing*, **36**(3), 287 – 314, (1994). Higher Order Statistics.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin, 'Maximum likelihood from incomplete data via the em algorithm', *Journal of the Royal Statistical Socoety, Series B*, **39**(1), 1–38, (1977).

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2019.

[9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*, Springer series in statistics New York, 2001.

[10] Karl Pearson F.R.S., 'Liii. on lines and planes of closest fit to systems of points in space', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572, (1901).

[11] Sebastian Goebl, Xiao He, Claudia Plant, and Christian Böhm, 'Finding the optimal subspace for clustering', in *Proceedings of the 2014 IEEE International Conference on Data Mining*, ICDM '14, pp. 130–139, Washington, DC, USA, (2014). IEEE Computer Society.

[12] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin, 'Improved deep embedded clustering with local structure preservation', in *IJCAI*, pp. 1753–1759, (2017).

[13] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, 'Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions', *SIAM review*, **53**(2), 217–288, (2011).

[14] Jiawei Han, Jian Pei, and Micheline Kamber, *Data mining: concepts and techniques*, Elsevier, 2011.

[15] J. A. Hartigan and P. M. Hartigan, 'The dip test of unimodality', *Ann. Statist.*, **13**(1), 70–84, (03 1985).

[16] L. Jing, M. K. Ng, and J. Z. Huang, 'An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data', *IEEE Transactions on Knowledge and Data Engineering*, **19**(8), 1026–1041, (Aug 2007).

[17] Duc Le, Zakaria Aldeneh, and Emily Mower Provost, 'Discretized continuous speech emotion recognition with multi-task deep recurrent neural network.', in *INTERSPEECH*, pp. 1108–1112, (2017).

[18] J. B. MacQueen, 'Some methods for classification and analysis of multivariate observations', in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, eds., L. M. Le Cam and J. Neyman, volume 1, pp. 281–297. University of California Press, (1967).

[19] Samuel Maurus and Claudia Plant, '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, New York, NY, USA, (2016). ACM.

[20] Dominik Mautz, Wei Ye, Claudia Plant, and Christian Böhm, '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, New York, NY, USA, (2017). ACM.

[21] Ismail Bin Mohamad and Dauda Usman, 'Standardization and its effects on k-means clustering algorithm', *Research Journal of Applied Sciences, Engineering and Technology*, **6**(17), 3299–3303, (2013).

[22] Dau Pelleg and Andrew Moore, 'X-means: Extending k-means with efficient estimation of the number of clusters', in *In Proceedings of the 17th International Conf. on Machine Learning*, pp. 727–734. Morgan Kaufmann, (2000).

[23] Benjamin Schelling and Claudia Plant, 'Diptransformation: Enhancing the structure of a dataset and thereby improving clustering', in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 407–416, (Nov 2018).

[24] Benjamin Schelling and Claudia Plant, 'Dataset-transformation: improving clustering by enhancing the structure with dipscaling and diptransformation', *Knowledge and Information Systems*, (Aug 2019).

[25] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller, 'Nonlinear component analysis as a kernel eigenvalue problem', *Neural computation*, **10**(5), 1299–1319, (1998).

[26] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouët, 'Are your data gathered?', in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pp. 2210–2218, New York, NY, USA, (2018). ACM.

[27] Nguyen Xuan Vinh, Julien Epps, and James Bailey, 'Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance', *J. Mach. Learn. Res.*, **11**, 2837–2854, (December 2010).

[28] F. Wang, N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, 'Parallel driving in cpss: a unified approach for transport automation and vehicle intelligence', *IEEE/CAA Journal of Automatica Sinica*, **4**(4), 577–587, (2017).

[29] Junyuan Xie, Ross Girshick, and Ali Farhadi, 'Unsupervised deep embedding for clustering analysis', in *Proceedings of The 33rd International Conference on Machine Learning*, eds., Maria Florina Balcan and Kilian Q. Weinberger, volume 48 of *Proceedings of Machine Learning Research*, pp. 478–487, New York, New York, USA, (20–22 Jun 2016). PMLR.

[30] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong, '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, (2017).