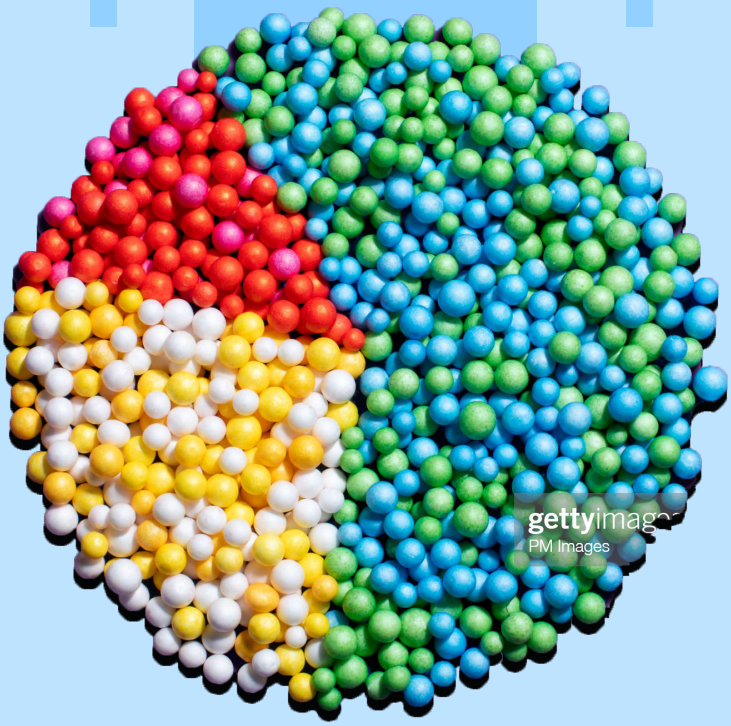
VIEW POINT

THE IDEA BEHIND DIP SCALING AND DIP TRANSFORMATION

Infosys Data Mining Center



Infosys®
Navigate your next

The use of any clustering algorithm must be preceded by cleaning data, handling missing data and integrating data. At that point the data is well-organized, but still its true structure may be hidden from the algorithm and applying inappropriate scaling or transformation would not reveal it. In this article we present the Dip Scaling and Dip Transformation techniques, which bring to light the data structure and improve the clustering algorithm performance.

The goal of this work is to overview and extend the methodology described by Benjamin Schelling and Claudia Plant in their article. We will provide explanation of Dip Scaling and Dip Transformation methodology with a stronger focus on statistical properties of the data, implementation within Python environment. We will also consider possible applications on structured and unstructured data types like pictures.

However, it starts with two main questions:

1. What does it mean to have *clusters* (or structure) in data with respect to its distribution of variables?
2. How can we reveal any hidden structure of our data with the proposed technique?

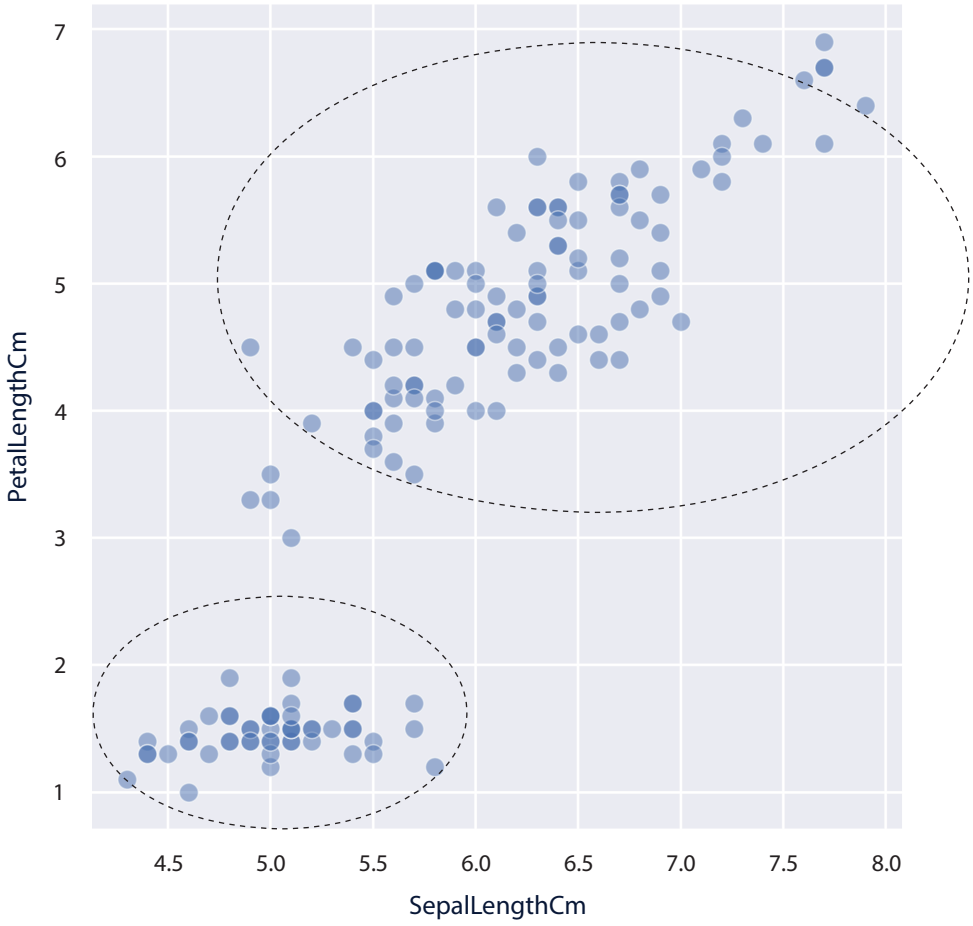
Let's to go through the definition of two main terms we deem to be of importance: **distribution** and **modality**.

"The *distribution* of a variable is a description of the relative numbers of times each possible outcome will occur in a number of trials" Ref link

"*Modality* is a measure of the number of modes in a distribution of a numerical variable. A unimodal distribution has one mode, meaning that the distribution has one value that occurs noticeably more often than any other value." Ref link

Below chart displays the relation between 'sepal' and 'petal' length of an Iris flower dataset. We can visually identify two groups in presented data.

Figure 1

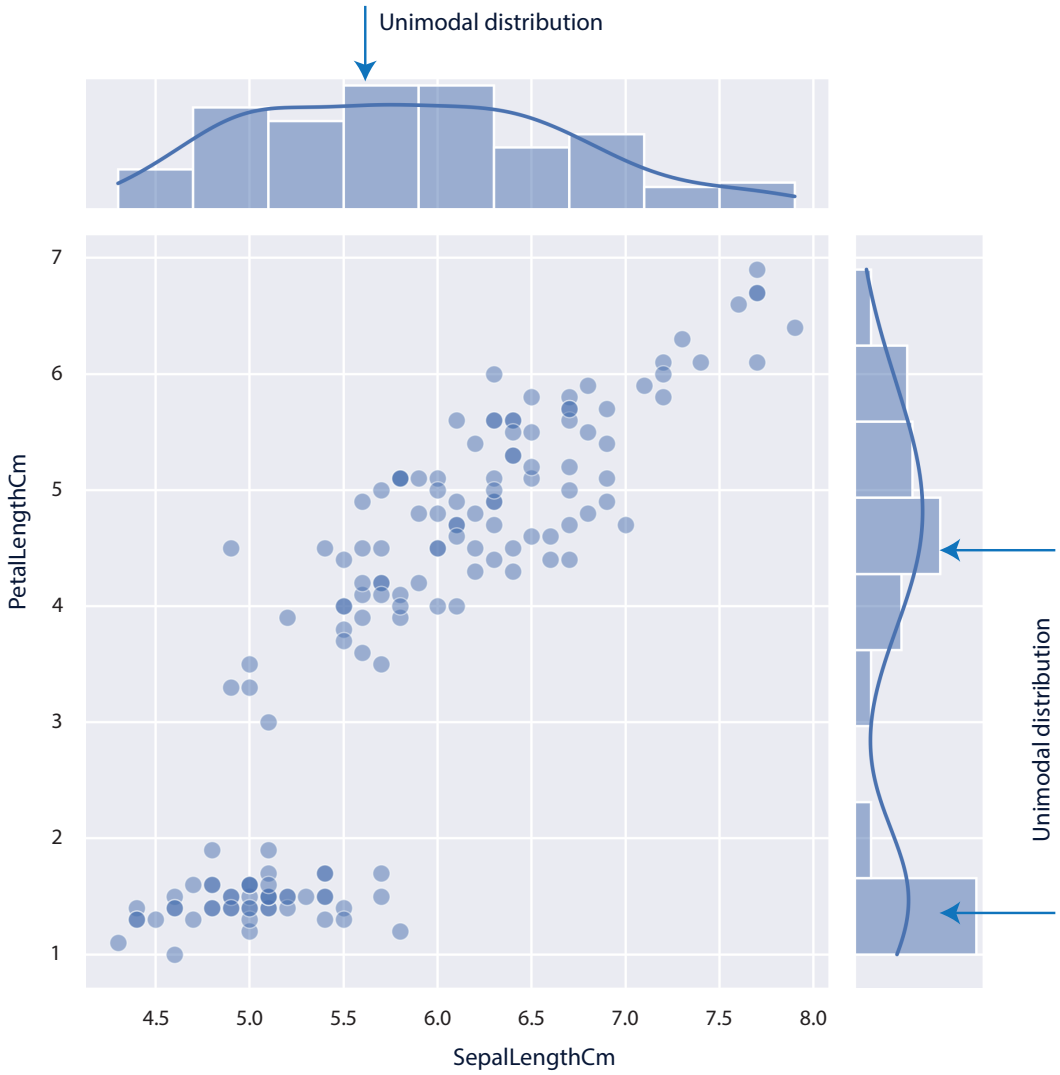


If we project presented data onto X and Y axes, we can assess distributions of both variables. Visually speaking, the distribution of petal length (Y-axis) seems to have two peaks (*multimodal distribution*) and the sepal length (X-axis) has only one peak (*unimodal distribution*).

Having more than one peak (two in our example) in dimension means that data within this dimension is somehow presented in groups. This means that multi-modal dimensions are the ones that we are looking for, while performing cluster analysis, because they contain valuable information about the data structure.

In our example, the dimension that is responsible for cluster formation is 'petal length' as it carries richer data structure (i.e., more modes) than 'sepal length' dimension. Therefore 'petal length' is more interesting for our cluster analysis.

Figure 2



Dip Value and Dip Scaling

How can we identify which dimension will be interesting for future analysis without looking at data presented visually?

Hartigan & Hartigan in the 1980s proposed a compact measure named Dip Value, which helps us answer this question. Let's consider the next example.

If we look at below charts, we can see that the top pair presents a histogram of perfectly normal data and its respective CDF (cumulative distribution function). Bottom pair presents a dimension with 3 modes (groups) and its respective CDF plot.

The idea of Dip Value is to measure the "distance" of samples of one-dimensional data to a unimodal distribution. In order to calculate this distance, we compare CDF to the closest S-curve. The further it will be, the more interesting it will be for future cluster analysis.

This number varies between 0 and 0.25; smaller value means that our dimension is closer to unimodality, while higher one - to multimodality. Once we know the value of the test, we can scale our dimensions accordingly, to highlight the one that is more interesting for us.

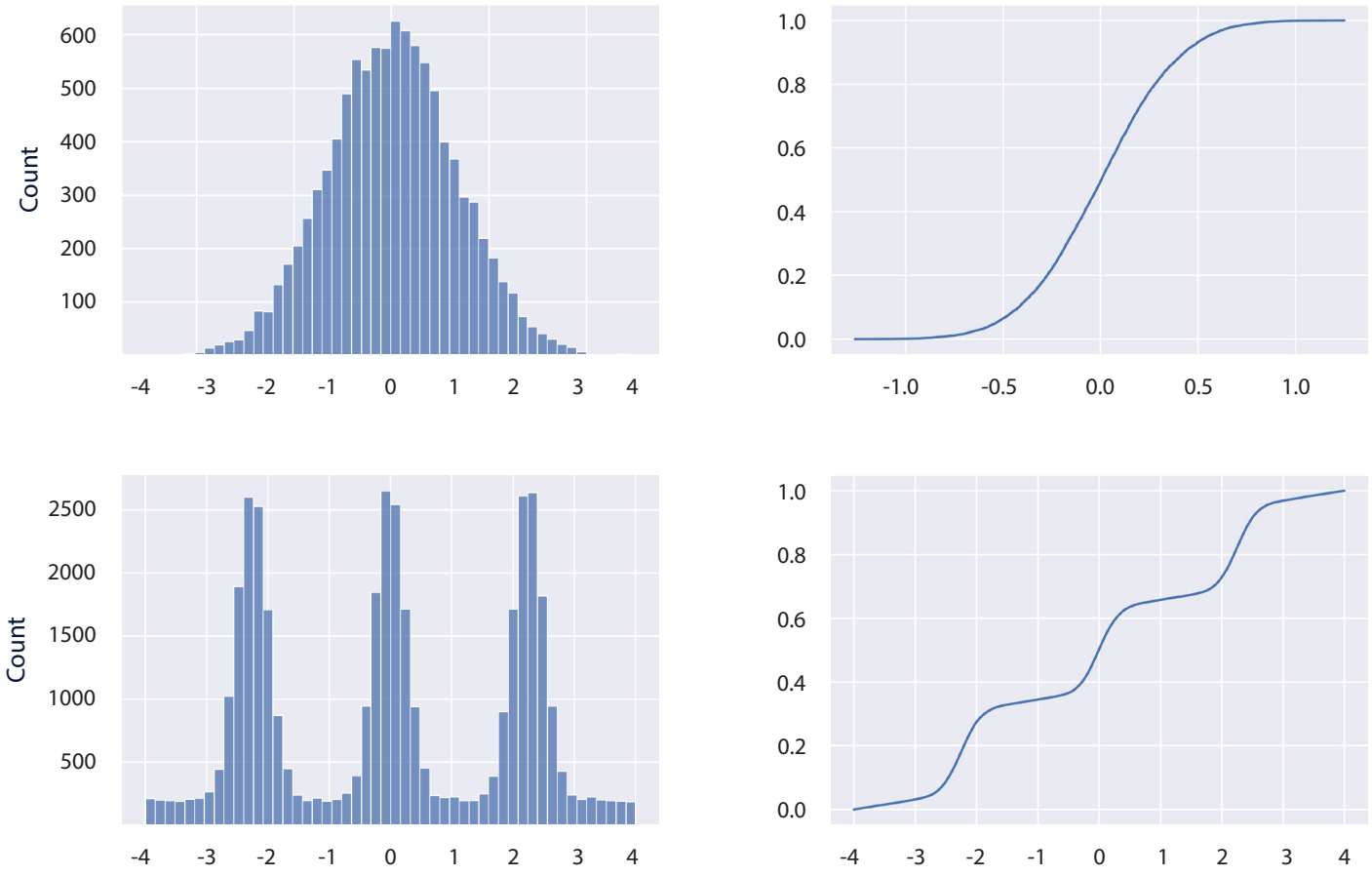
As dimensions with only one mode will have smaller value for this statistic, we will shrink them more by applying scaling, and we will therefore reduce their impact on the clustering results.

Now let's check on our example how scaling axes according to Dip Value can improve the formation of clusters.

As an example, we will use k-means algorithm which belongs to a statistical learning method widely used in cluster analysis. It is widely used for the finance sector.

The fact that the k-means method belongs to a class of statistical learning, suggests its success in producing effective result is inseparable from the statistical properties of the data itself. These model related assumptions are described in the article: K-means clustering is not a free lunch.

Figure 3



In order to understand the impact of technique on k-means clustering algorithm, we first need to understand when it fails. One of the assumptions of this algorithm is that data has a spherical distribution. If we plot the data, we should see groups in a form of close-to-perfect circles.

In case if our data leaning towards different shapes, like elliptical, then the algorithm might fail us. This is perfectly illustrated on figure 5. Colors show the k-means clustering results

Figure 4

Figure 5

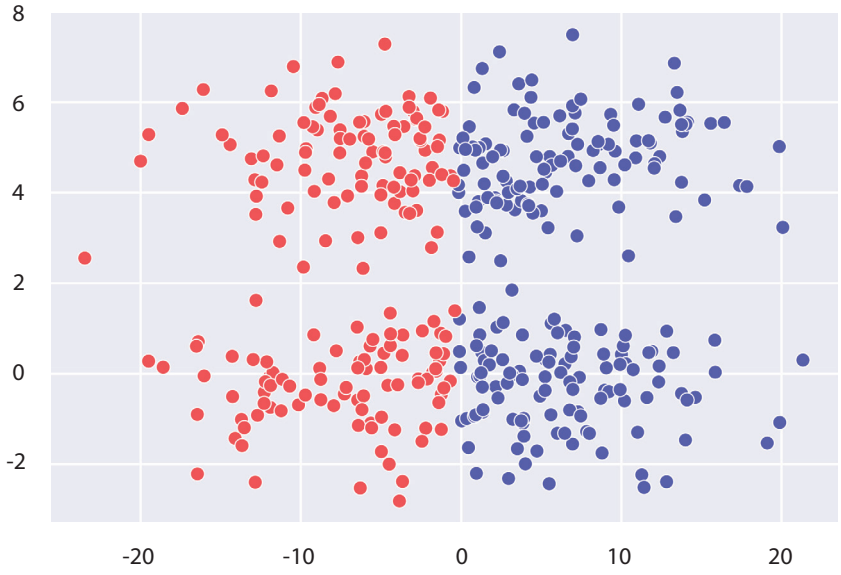


Figure 6 (a)

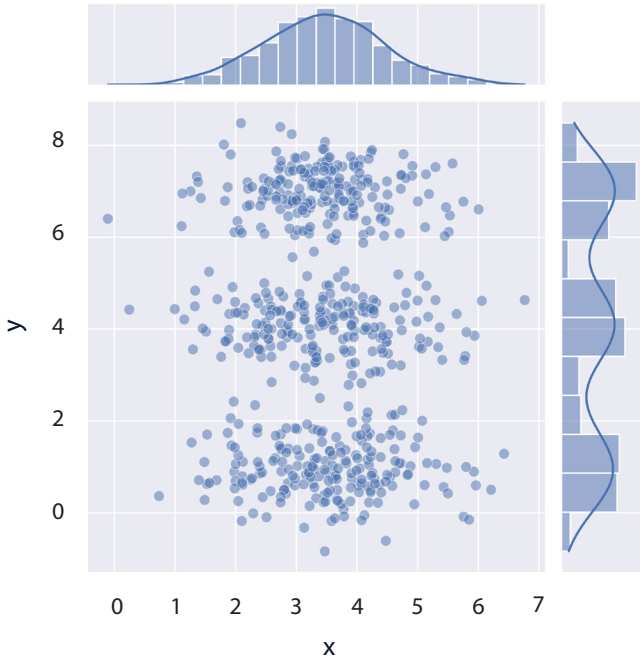
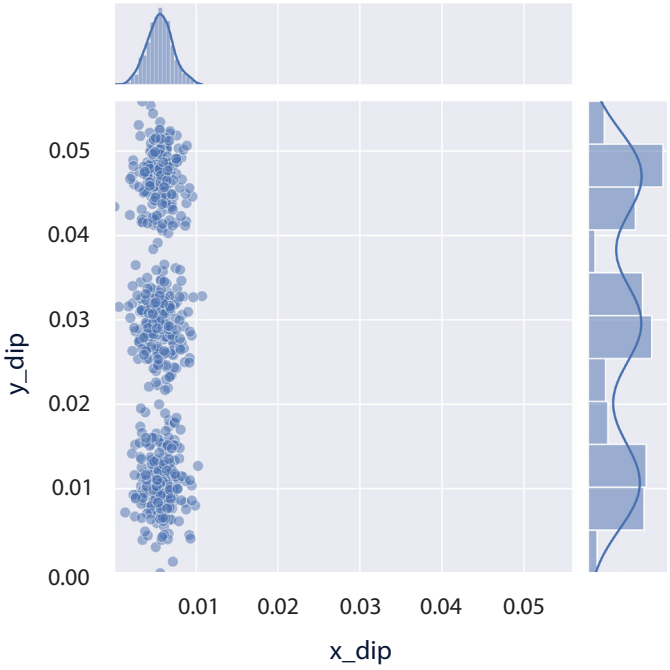


Figure 6 (b)



Let's have a look now at similar data and check if Dip Transformation can help us with this issue.

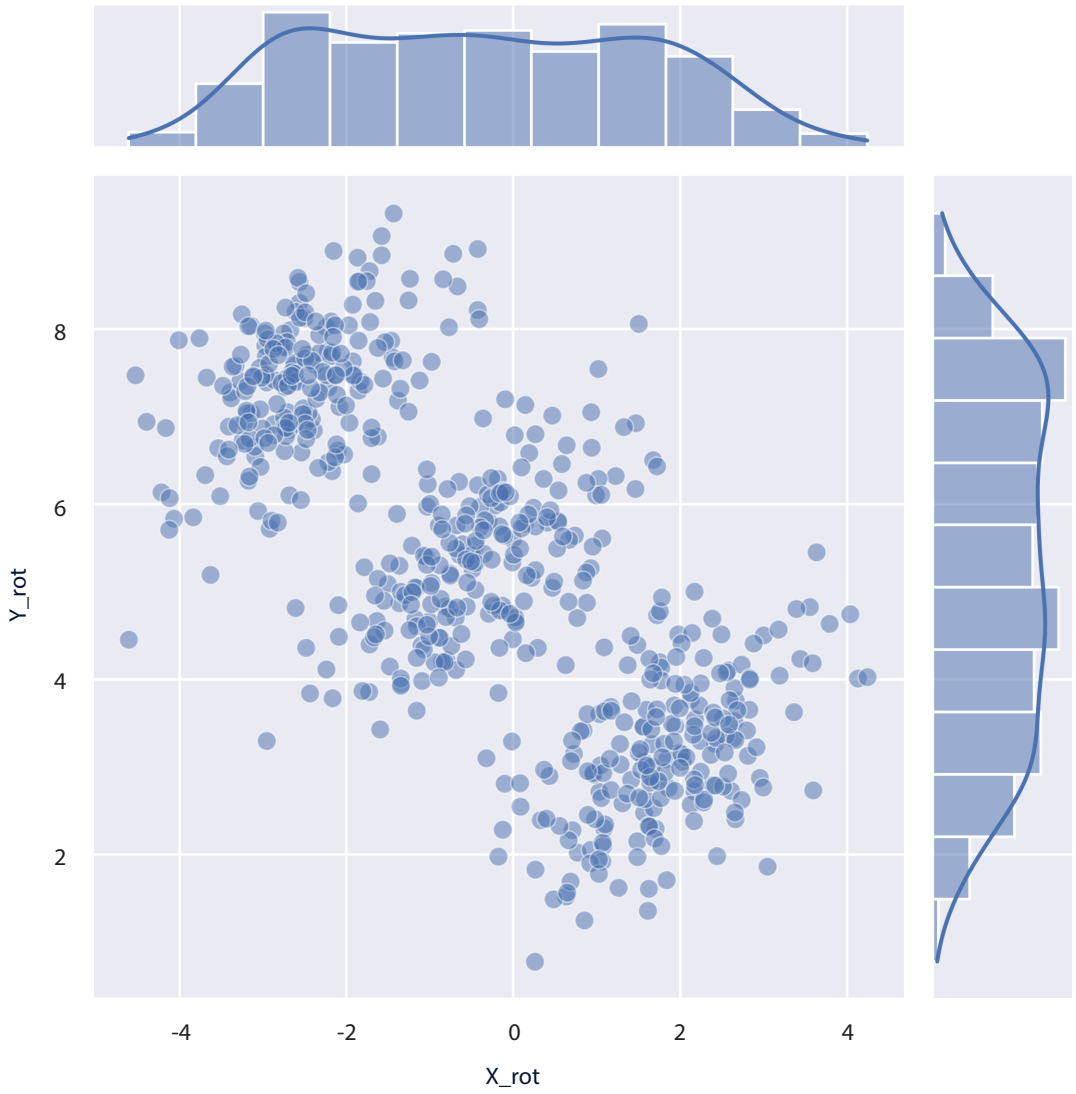
On the left side of the image (Figure 6(a)) we can see that the dimension on the x-axis is unimodal, while on the y-axis it has three modes. The Dip Value D_x on the x-axis equals 0.011 and it is small when compared to that D_y on the y-axis which equals 0.055.

After min-max normalization on both axes using respective intervals [0,0.011] and [0,0.055] respectively, we obtain the scatterplot shown on Figure 6(b). After this operation, called Dip Scaling, clusters are easier to find by k-means: they are not stretched across x-axes anymore and are presented in more circular form or stretched in a "good" direction. Note that changes of usual min max scaling are minimal in this case, since x and y values have the very same range.

Dip Transformation

Below example works perfectly for data which is axis parallel. But what if cluster structure was hidden? Let us spin the data around its mean by 45 degrees.

Figure 7



As we can see now, obtained dataset looks similar from both axes, and consequently Dip Values are very closed to each other – D_x = 0.015 and D_y = 0.014. It effectively means that we do not have any interesting dimension. This statement can guide our clustering algorithm, because scaling by the very similar number does not affect the data structure and has not impact on k-means. Note that neither min-max scaling would help in this case!

The example shows that Dip Scaling itself does not work in every situation, but it also shed a light on a new way of data transformation: we should seek the optimal angle to rotate the data before Dip Scaling. This is the basic idea standing behind Dip Transformation.

The example was instructive and illustrative. However, in this case k-means algorithm worked well without any transformation. Now, we are going to present few examples which show the Dip Scaling and Dip Transformation power together.

In the scatterplot presented on Figure 8(a) there are ten clusters. The distribution on x-axis is unimodal, while that on y-axis is multimodal. On Figure 8(b) there is the data after Dip Scaling.

Figure 9(a) and Figure 9(b) show the result of applying k-means algorithm on both the raw data and dip scaled data, respectively. Each cluster is marked by a different color. The output of k-means on raw data is poor. At the same time the algorithm perfectly recognized clusters after Dip Scaling.

Figure 8 (a)

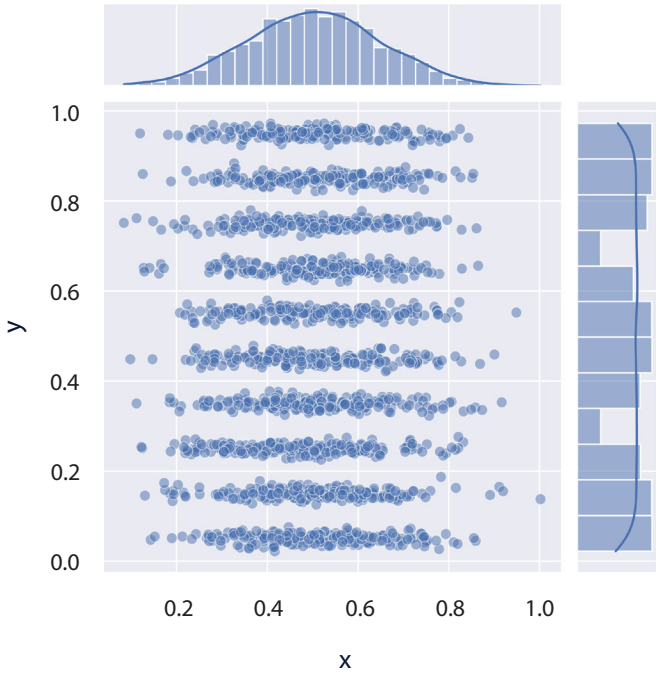


Figure 8 (b)

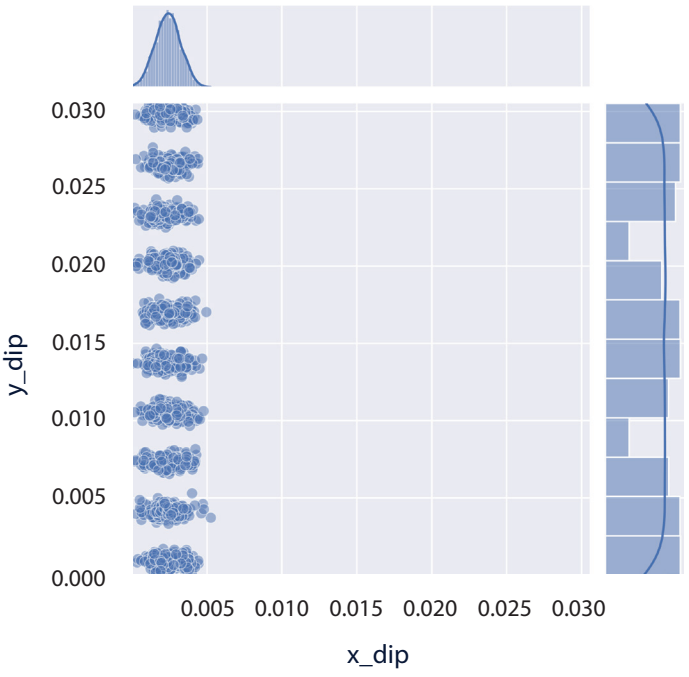


Figure 9 (a)

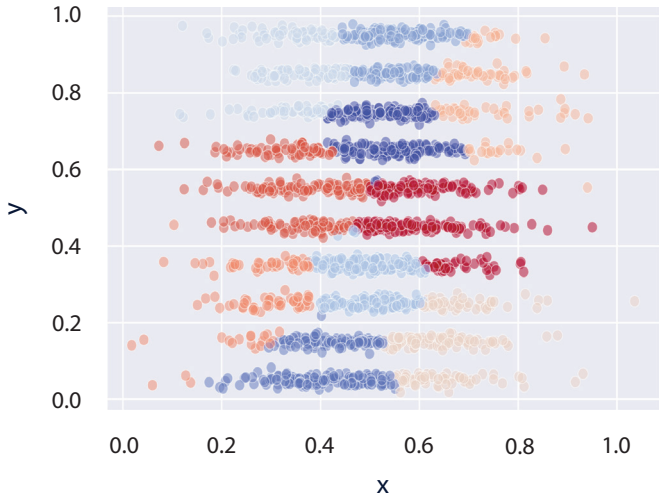
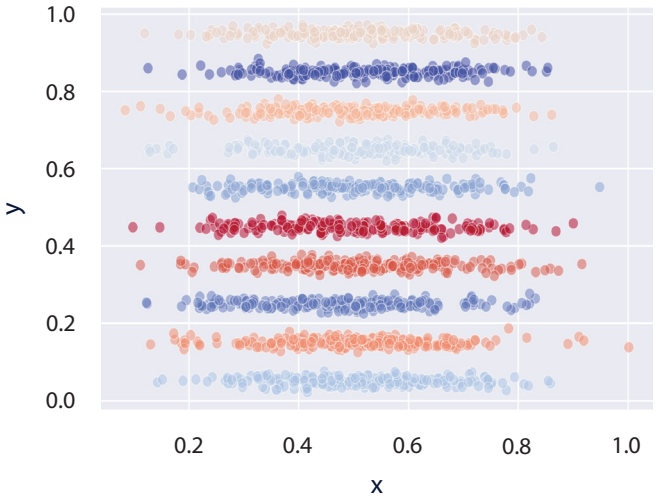


Figure 9 (b)



Let move to the next example. On a Figure 10(a) we can visually identify three groups. Note that the x and y distributions are very similar and basically unimodal. The x-dip is 0.013 and y-dip is 0.010. After Dip Scaling the data changes slightly and the output of k-means does not improve much. Again, min-max scaling procedure would not help much, since x and y values have the very same range. What we can see on Figure 10(b) is the result of Dip Transformation.

Dip Transformation algorithm starts from computing Dip Values D_1 and D_2 for two features, then finding their maximum D_{max} and applying Dip Scaling. Once data is scaled we go through the next steps:

1. Data is rotated by a small angle that is calculated for each iteration.
2. New Dip Values D'_1, D'_2 and D'_{max} are computed.
3. If $D'_{max} > D_{max}$, then $D_{max} := D'_{max}$ and Dip Scaling is performed.

Step 1-3 are repeated until all those small rotations sum up to 180 degrees. Some technical details are omitted; they can be found in the original paper. Figure 10(b) is the final output of Dip Transformation applied to data from Figure 10(a).

Figure 10 (a)

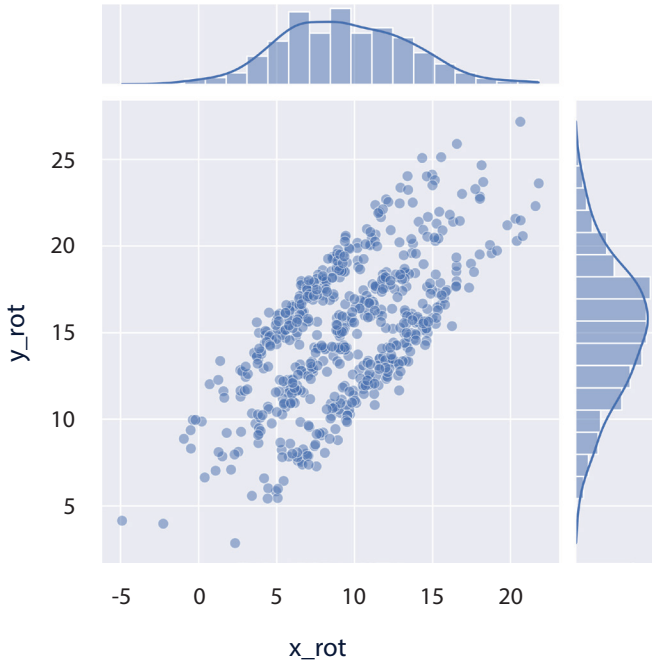
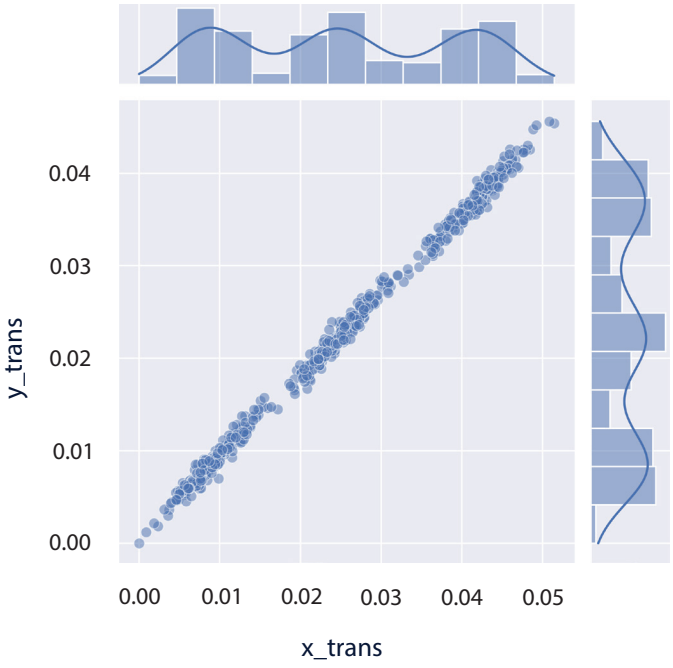
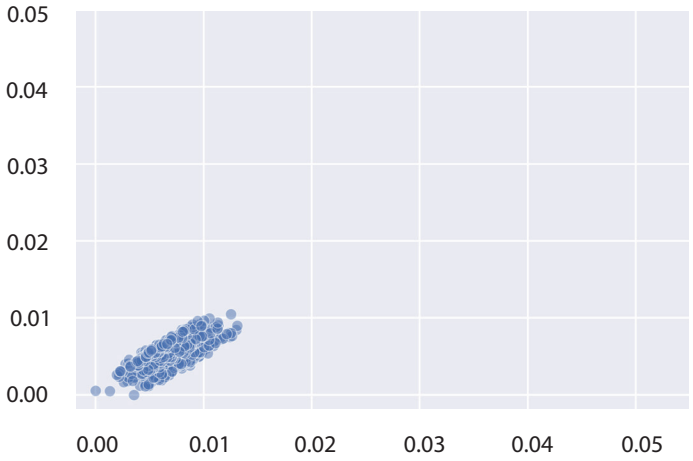


Figure 10 (b)

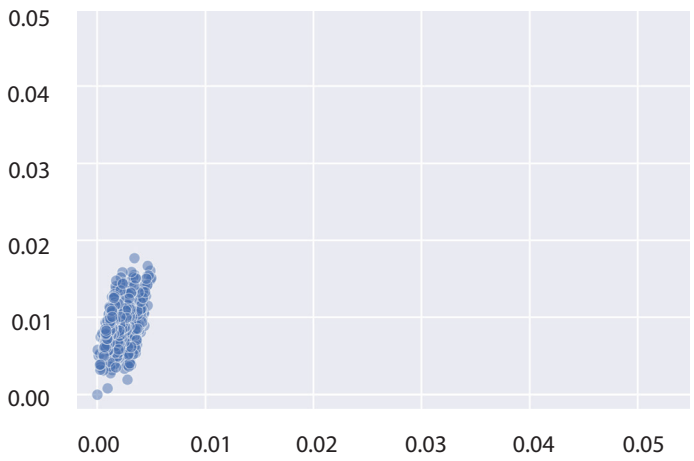


On the next page you can see how the data is transformed step by step using rotations and Dip Scaling during Dip Transformation.

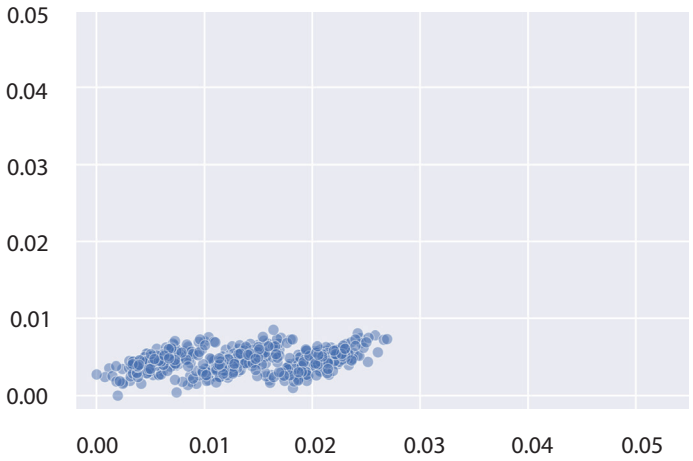
Frames of Dip Transformation animation



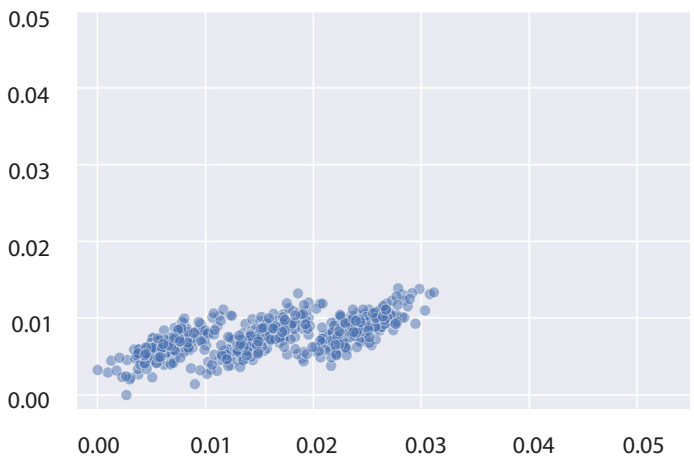
Initial step



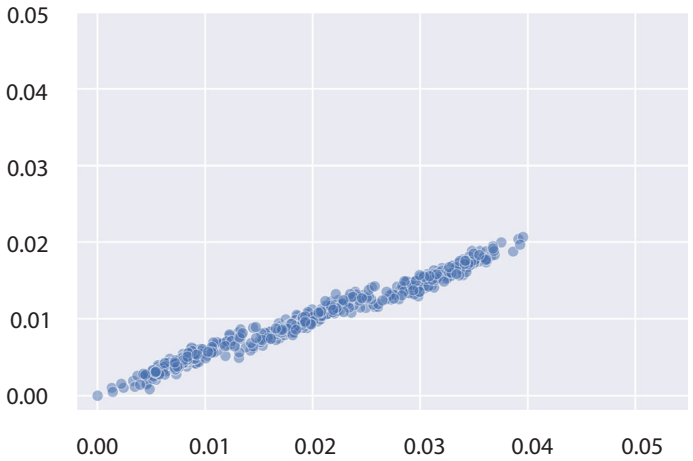
Step 1



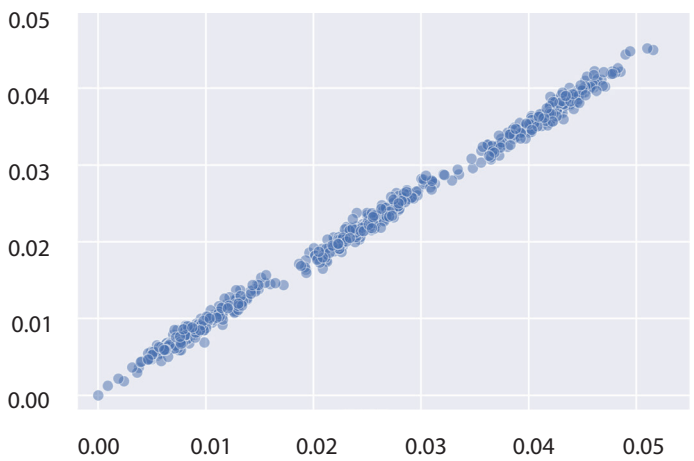
Step 2



Step 3



Step 4



Step 5

The aim of Dip Transformation is to prepare data for k-means algorithm. In Figure 11(a) there are clusters found by k-means after applying min-max scaling. We can compare them to the clusters found on dip transformed data in Figure 11(b).

Till now we have presented two-dimensional Dip Transformation algorithm. The question which arises here is: does this method work in higher dimensions?

Suppose that the data has n-features, $n>2$. Multidimensional Dip Transformation applies steps 1-3 of the algorithm to each pair of features once per cycle. D_{max} is global for all of them. The algorithm will stop when the sum of all rotations is more than $180\times n$ degrees.

Due to optimization purpose, rotation angles are irregular: some feature pairs are rotated more than others at the end of the algorithm.

Idempotence of Dip Scaling and Dip Transformation

Let's assume that the data consists of *n* numerical features. Then each sample can be represented as a vector in R^n. Many known transformations, like min-max scaling, standardization, or normalization, are linear transformations. They have also very important property - idempotence.

A transformation $T:R^n \to R^n$ is idempotent if $T=T \circ T$. Roughly speaking, if a transformation is idempotent, then applying it twice, or more times, gives the same result as applying it only once. So, a single use of the idempotent transformation delivers a final output, with no further need to employ it again.

The linear transformation of one feature does not change its modality. Since Dip Scaling technique scales each variable to interval $[0,d_i]$ where d_i is a Dip Value of the *i*-th feature, then it is an idempotent linear transformation.

Figure 11 (a)

Figure 11 (b)

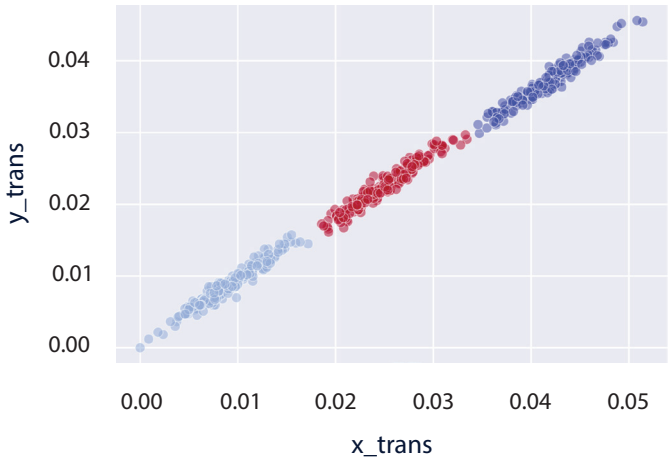


Figure 12 (a)

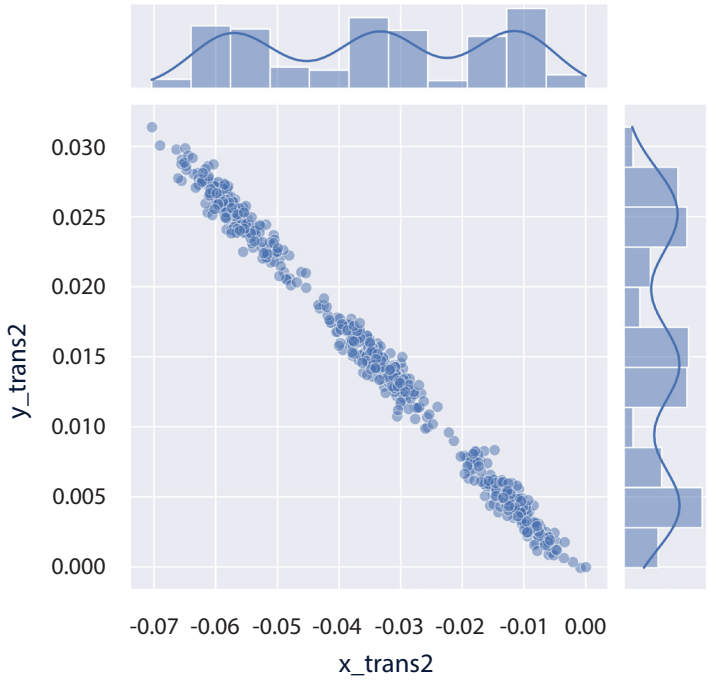
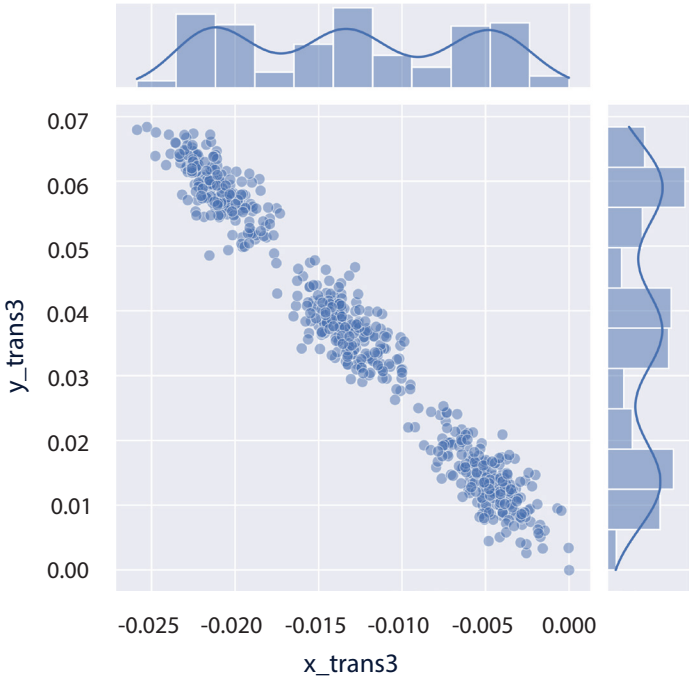


Figure 12 (b)



For Dip Transformation it is more complicated. On Figure 11(b) there is a scatterplot of data transform by Dip Transformation. Let us compare it to the second and third iterations of Dip Transformation given in Figure 12(a) and Figure 12(b), respectively.

The iterations are different. This proves that Dip Transformation is not idempotent. It could be if the iterations of Dip Transformation improved the output. Our experience says that it does not.

Implementation of Dip Scaling and Dip Transformation

The Authors of the original paper implemented Dip Transformation in Java. In order to make this method accessible for a wider audience, we have implemented it in sklearn-type package in Python. For those who are familiar with sklearn, it is easy to use. The documentation, source code and examples of use are available in public repository.

The main part of the method computes Dip Value, which is a part of Hartigan & Hartigan's dip test for unimodality. Although there are several packages implementing it in Python, we modified one of them using numpy in order to make it fast.

Dip Scaling and Dip Transformation use rotations and scaling operations, which are matrix transformations. The compositions of such operations are given by matrix product. We can use this property to transform even gigantic data sets with an already trained model (e.g. on smaller sample).

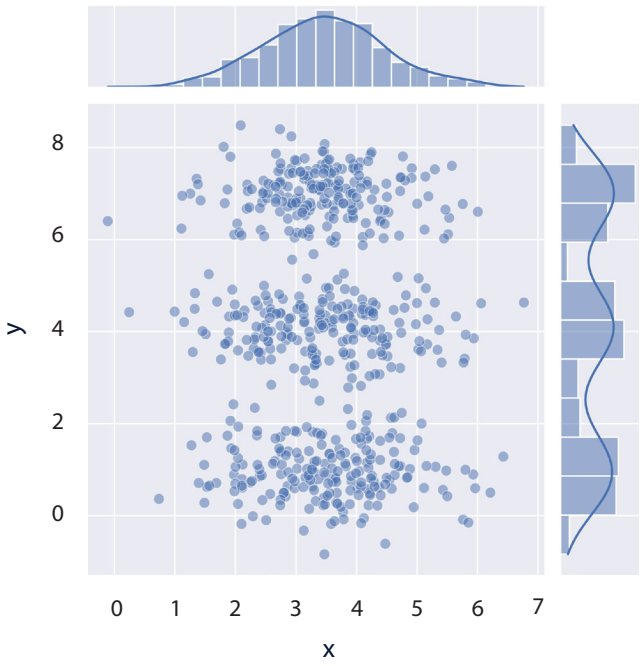
Limitations of Dip Transformation

Dip Transformation is designed for improving k-means algorithm's efficiency. One of k-means method assumptions is that clusters do not overlap and can be separated by hyperplanes. Dip Transformation is a reversible matrix transformation (it just stretches and rotates data in several directions), and therefore the transformation of a hyperplane is a hyperplane again. This means that we need to assume hyperplane separability of clusters, to successfully apply Dip Transformation. More precisely, the first limit is that to obtain spherical clusters by Dip Transformation, they must be elliptical at the beginning.

The second limit is the number of modalities per dimension. Due to the nature of Dip Value, a uniform distribution is unimodal. And if the number of modalities in a dimension is too big, its CDF may look like a uniform CDF. Then there is a chance that highly multimodal dimension will be considered as unimodal one.

Possible Applications

Figure 6 (a)



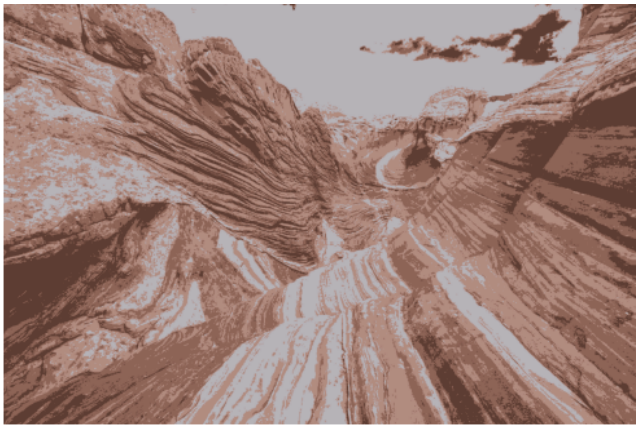
Let's have a look at Figure 6(a) once again. If we reduce the x-dimension, and restrict only to the y-dimension, we will still be able to properly recognize clusters. In general, finding an appropriate rotation and Dip Scaling would allow to reduce axes which bring little information on the cluster structure. Finally, one would be able to represent the data on two or three dimensions to further analyze and visualize it.

Applying Dip Transformation algorithm for image pre-processing allows to improve results for color-based image segmentation and color quantization. We use these techniques to find a different colorful objects and parts on an image. It is one of the important steps in some computer vision algorithms.

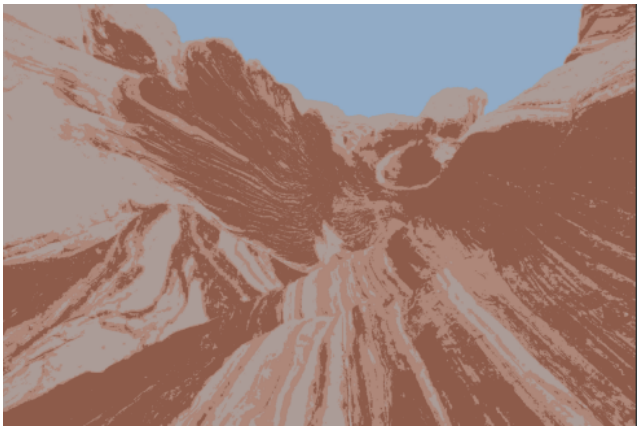
Also, it may help reduce the file size of the image. One of the simplest implementations is k-means algorithm used on the color channels of the image. Dip Transformation improves the separation of different colors, which makes result more visually colorful.

Below are shown some examples.

Original

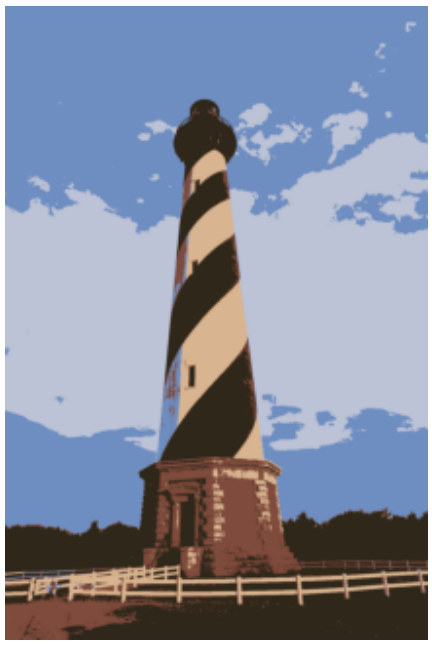


k-means, k=4



Dip transformation + k-means, k=4

Original



k-means, k=4

Dip transformation + k-means, k=4

Summary

Dip Transformation can improve k-means algorithm by "finding" hidden structure within dimensions. Usually, before applying k-means, one standardizes the data coordinate-wisely. Dip Transformation, or even Dip Scaling, nullifies any previous linear transformation of coordinates, which makes it the only transformation needed for k-means. The method has only rotation coefficient parameter. It also changes accuracy at the cost of computation time, which additionally makes it easy to use.

Research:

Kyrylo Myshnov
kyrylo.myshnov@infosys.com

Review:

Olivier Magnan
olivier_magnan@infosys.com

Kateryna Jastrebowa
kateryna.jastrebowa@infosys.com

Szymon Głąb
szymon.glab@infosys.com

Notes

For more information, contact askus@infosys.com

Navigate your next