Clustering Visualiser - Visualising partitioning-based clustering algorithms

Richard Paul* University of Vienna
Severin Staudinger† University of Vienna
Michael Trimmel‡ University of Vienna

ABSTRACT

This paper gives an overview of the implementation of the **Clustering Visualizer**, a novel clustering algorithm analysing tool. The main focus is a tool for beginner and experts in clustering techniques. The aim is to explore different types of partitioning algorithms, namely K-MEANS and K-MEDIANS with different types of initialisation and update strategies. The user will get deeper knowledge of the functionality and the procedure of the algorithm with the possibility to analyse each selected setting at the point of interest.

Index Terms: Partitioning—Visualization—Clustering techniques—Similarities;

1 INTRODUCTION

These days clustering algorithms capture a very important tool for mining data and similarities between high dimensional features. To find those complex structures in the data, a lot of different clustering approaches are available for different problem types.

A main group of techniques is covered by partitioning methods like K-MEANS and its derivates which are the clustering techniques of interest in this paper and in our project implementation. Partitioning methods are the more easy methods in the space of clustering techniques. A common problem of people which are at the beginning of learning how clustering algorithms work is that it is very hard to understand how such an algorithm works until a solution is found and the algorithm is converged. But not just beginners are the target group for such an analysis tool. Also much more expert users often wants to explore different settings or strategies and see their behaviour to the solution.

Our goal is to provide such an clustering algorithm analysis tool which brings deeper understanding to fresh users in this area, but also can be used as an analysis tool to explore different parameter space settings or differences between similar algorithms.

2 RELATED WORK

Beside the websites there are academic papers which targets some visualization techniques. Like in [4], which introduces a framework for different data flow models independent of their domain and additional a set of strategies for parameter space analysis. Further they characterize different the different analysis tasks possible.

Another very interesting literature for this topic is given in [2]. Here interactive multiple views visualizations are considered and maintained in form of visualization pipelines which are then optimized.

There are a lot of interesting websites which try to describe how a clustering algorithm is working. For example at this site[1] the user can see an animation of a partitioning algorithm. The negative aspect here is that the user has no influence to the procedure.

Another very sophisticated tool is presented here[2]. Here the user has much more influence to the parameter space of the techniques implemented and also its data sets. Some methods are similar to our basic implementation because they are very useful for analysing the behaviour of the algorithm, like iterating through the independent steps or choose the cluster centers at the start up.

3 APPROACH

The aim of the **Clustering Visualizer** is to bring the possibility to the user to get deeper knowledge of different clustering algorithms, their parameter space and the resulting behaviour of different parameter settings to the user. This is given by well implemented and well-wrought ideas and a highly sophisticated combination of different visualisation techniques.

The user has the ability to interactive manipulate different clustering techniques, namely K-MEANS and K-MEDIANS. This can be done by a number of different interactions like a set of different initialisation strategies which are implemented. Next to the default strategy of random cluster centers, the *D2*-strategy of [1], the *farthest* heuristic of [3] is implemented and additionally, the user has the possibility to choose the cluster centers himself to explore the effects of different initialisation centers.

Further two different update strategies are possible to choose. The default update strategy of *Lloyd* and the *MacQueen* update strategy which differs a little bit. Further algorithm specific parameters are the number of clusters.

A huge aspect of our visualisation approach are the small multiples of each iteration until convergence, the novel filters and the head to head comparison. The small multiples show an overview over each iteration until convergence. Here the user has the first overview over the whole procedure and can find the iterations where the algorithm run through the biggest variation easily.

The unique *cluster center path filter* and shows the user how the algorithm is converging and the way of the cluster centers with the update strategies of interest. For a better focus at the cluster center path the user can disable the data points and the center points to just show the path if necessary.

The head to head comparison allows comparison between different iterations, different update strategies and initialisation strategies or different settings in parameter space easily. This is a huge pro of this

*e-mail: a1302451@univie.ac.at
†e-mail: a01308699@unet.univie.ac.at
‡e-mail: a1449702@unet.univie.ac.at

[1] https://www.toptal.com/machine-learning/clustering-algorithms
[2] https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

visualisation and allows deep insights in the clustering procedures over the time period. Last but not least the user has the ability to choose if the iterations should be animated with different velocities or he can switch for himself through the different iterations.

4 Implementation

Technically the basic framework of the clustering tool is implemented in Java[3] and the visualisation is done in JavaFX[4]. Our first approach to use the Processing[5] framework was discarded when we focused some issues at the beginning of the high fidelity prototype. There have been huge issues by integrating the visualisation of Processing into the JavaFX-GUI. Therefore we decided to implement also the visualisation in JavaFX. This decision led to much more code and work, but also to a more structured and clean project.

For building our project we used Apache Maven, which is great for modular programming used in large projects. We also included the logging framework log4j2, which offers the user great opportunities like adjusting logging levels, outputs and more even when the app is running. This is great for bug fixing and reconstructing unexpected behaviour.

5 Results

The resulting clustering visualisation tool is a really nice application which is in the position to allow deep insights into the procedure of the algorithms implemented. As opposed to implement multiple clustering algorithms, the final version is focused on K-MEANS and some derivations of it. The decision was done after the feedback of the M3 milestone. At the startup of the application, the user gets the first overview over the whole procedure of the algorithm until convergence like shown in figure 1. This is the implementation of **Shneiderman's Mantra**: Overview first, zoom and filter, then details-on-demand.

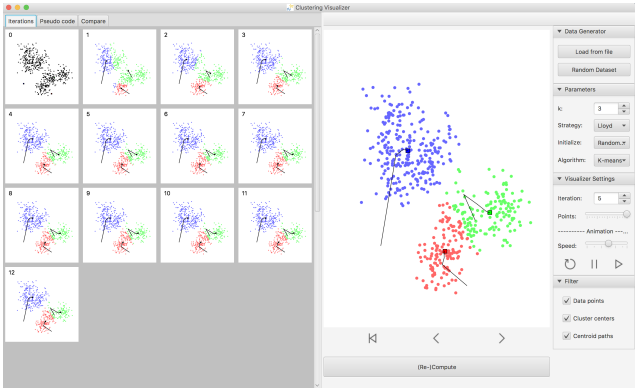

Figure 1: Overview over the whole procedure until convergence - Small multiples to figure out easily the steps of interest

A huge ergonomic aspect is that the iteration of interest can be selected on the left side by a mouse click event. Then the view on the right side is updated immediately to the chosen iteration step. This is a part of the zoom and filter process of the mantra mentioned above and a huge help to find iterations where critical steps are performed. For further details (on-demand), tooltips are implemented when the mouse is hovered over points or cluster centers. They provide additional information for the user and are shown in figure 2.

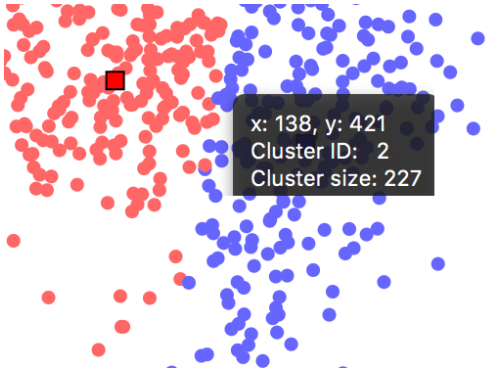

Figure 2: Tooltips which show additional information on demand when hovering over points or cluster centers (Shneidermans Mantra)

Another very interesting and informative view is the head to head comparison shown in figure 3. This view has so much power to show different behaviour of different clustering algorithms, initialisation or update strategies or different settings in parameter space to the user. One specific scenario of use would be for example to compare different initialisation techniques and its influence to the convergence. An additional feature here is that the iterations can be synchronised or run independently step by step next to each other.

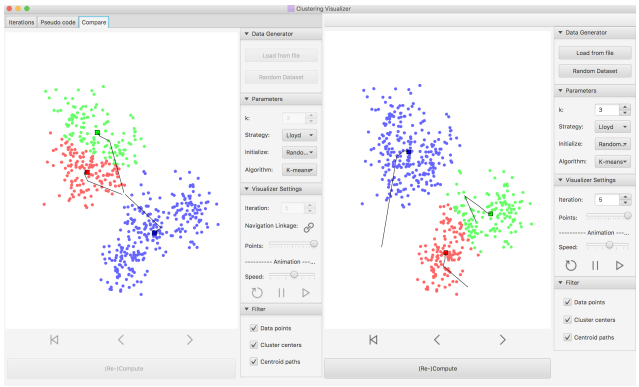

Figure 3: Overview over the head to head comparison of different initialisation steps, update strategies or cluster parameter space settings

Next to the comparison of different initialisation techniques another scenario of use would be the comparison of different algorithms. When the user would like to know which algorithm is performing better at the first iterations because maybe he does not want to iterate until convergence he can inspect the behaviour next to each other.

For more fresh users which do not have much pre-knowledge in clustering techniques another use case could be the pseudocode tab from figure 4. This allows the user to get information which line of the code is executed currently. This can be very informative and a huge help to understand the procedure of the algorithm. (Note: Because of time aspects this feature is not fully implemented yet)

[3] https://www.java.com/de/
[4] https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm
[5] https://processing.org

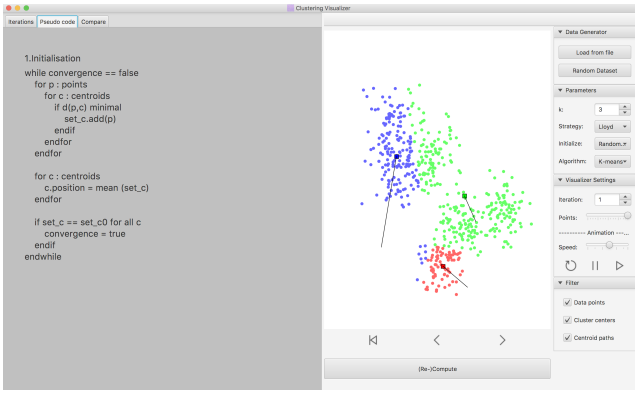

Figure 4: Overview over the pseudocode tab; In the final version each iteration step should be highlighted in the respective pseudocode line (not implemented yet)

More expert users will have more interest in different parameter space settings. These can be chosen as shown in figure 5. The common parameters like the number of clusters, the update strategy and the initialisation strategy can be selected manually and independent of each other when using the head-to-head comparison of the application. This feature also allows the user the option that he can compare for example K-MEANS with K-MEDIANS synchronised or independently depending if the synchronise-mode mentioned above is enabled or disabled.

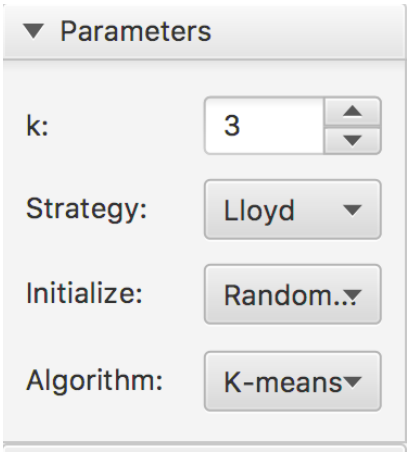

Figure 5: Parameter space setting tab. Here, beside the algorithm, the user can select different number of clusters, initialisation strategies and update strategies.

Most of the users may have interest in just discover the procedure of clustering algorithms and get a better understanding of them. Some others may want to use our application to analyse their own data/problem. For this specific type of users the application provides the possibility to load data from a file. With this feature the user can analyse and inspect his two dimensional data set easily with our tool. Figure 6 shows the tab where the user can choose between randomly generated data or loading his own data from a file of from example data sets provided.

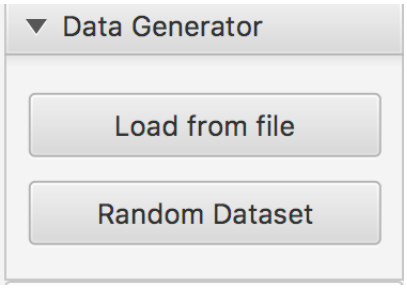

Figure 6: Data set panel where randomly generated data can be loaded or data from example files provided by the user. So the user has the ability to analyse his own data

In our tool the user has full control over how the progress of the algorithm chosen. This means, that he can iterate through the steps manually or can use the animation feature which is shown in figure 7. Here the user can jump to an iteration step by selecting the number in the text field. Further an animation feature is implemented where the user can choose the speed of the iterations between each step. Another interesting feature is the points slider. Here the user has control over the percentage of updated points per iteration. Further tools in the animation feature are the *pause* mode and the *rerun* mode. These features can as the name suggest pause or reset the animation.

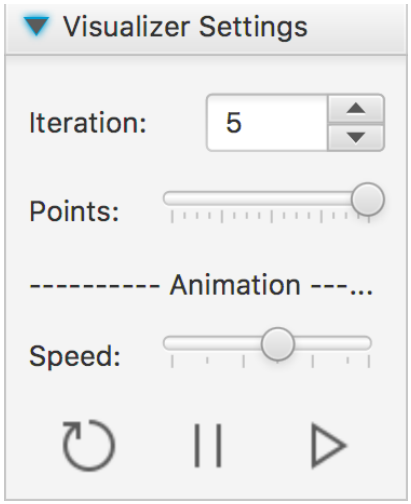

Figure 7: Visualizer parameter space setting panel. Here the algorithm type can be chosen. Also the common parameters can be chosen and additional different strategies for initialisation and update strategies for the iterations can be chosen by the user.

The next very important control features complements the filter tab. These are the data point filter, the cluster centers filter, the centroid path filter and last but not least the shapes filter. These filters control the functions which enable or disable the points which are clustered, the centers of each cluster and the path of the centers through each iteration. The shapes filter is a very powerful feature for people which are colour blind. Also when there are a lot of clusters just colour coding could be not enough to distinct the clusters easily. This filter covers these problems and the points are additionally coded by shape instead just by colour.

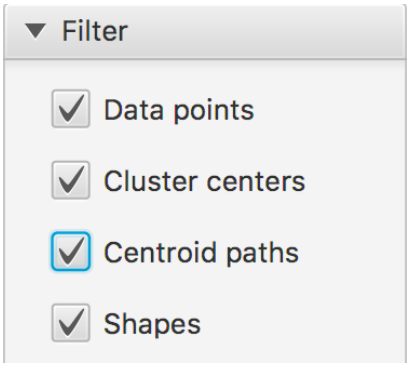

Figure 8: Filter tab. Here the user has the ability to enable and disable the data points, the centers and as a special feature the center paths of the cluster centers. Further a colour-blind mode is implemented which enables an additional shape encoding of the data points to the different types of clusters.

Since none of us is colourblind, we were not able to judge how much the shape-encoding actually helped in making the clusters easily distinguishable. Instead of running the program with deactivated colour-encoding, we decided to evaluate the shape-encoding by using a webtool called Coblis[6], where we uploaded screenshots of our applications clustering result and transformed them to what they look like for a colourblind person.

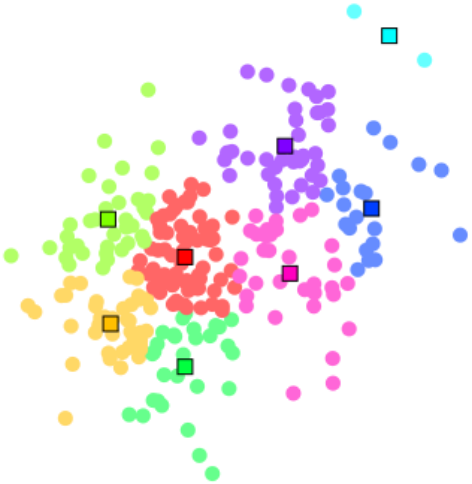

Figure 9: A small colour-encoded test sample.

The sample in figure 9 was clustered with $k = 8$, which is the maximum k our application supports at the moment, in order to evaluate shape-encoding for all different shapes, we provide. Of course the clustering result is more or less nonsense, but useful to demonstrate how shape-encoding can provide easier perception of the clustering result.

A problem that rises also for people who are not colour-blind can also be seen here: Since the cluster centers are mostly randomly set it is possible for two similarly coloured clusters to lay directly next to each other, which makes these clusters difficult to distinguish.

[6]http://www.color-blindness.com/coblis-color-blindness-simulator/

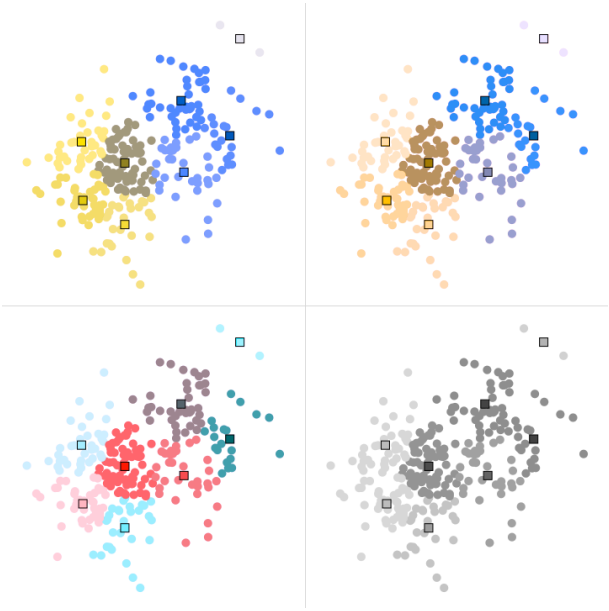

Figure 10: From top-left to bottom-right: red-blind, green-blind, blue-blind and monochromatic perception of the colour-encoded sample from figure 9.

The colourblind perceptions of figure 9 seen in figure 10 show that it can be very difficult to distinguish the clusters using colour-encoding only. Especially in the red-blind and green-blind version the original 8 clusters melted into 4 or 5 big clusters. The blue-blind version is not as that bad, but still faces problems when similar colours are close to each other. The monochromatic view is catastrophic. The clusters melted down into 2 big clusters, ignoring the 2-point cluster in the top-right corner.

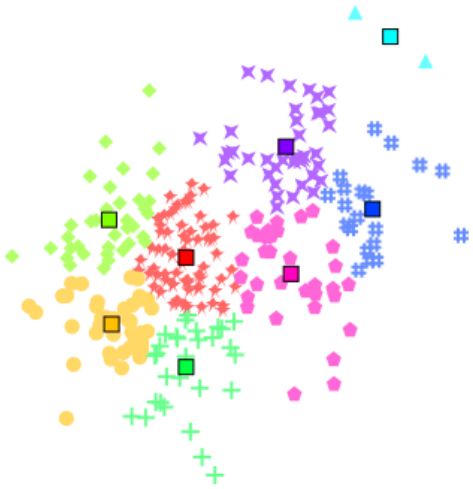

Figure 11: The same sample as in figure 9, but also shape-encoded.

Comparing the shape-and-colour-encoding to the colour-encoding, we can clearly see an improvement in visibility. The pink and red clusters are now easily distinguishable. Though there is also similarity between shapes as for example the diamond (grass green), the circle (yellow) and the pentagon (pink) are very similar. Next to each other they would not be easily distinguishable in shape only.

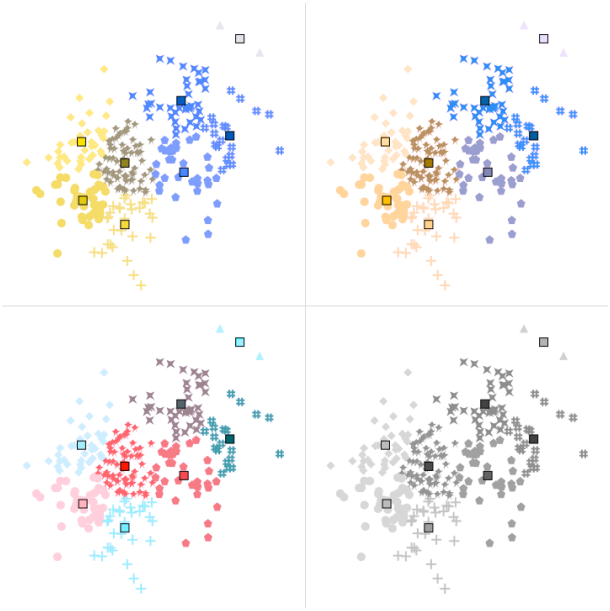

Figure 12: From top-left to bottom-right: red-blind, green-blind, blue-blind and monochromatic perception of the colour- and shape-encoded sample from figure 11.

Anyhow we find the shapes to be useful to make clusters more distinguishable. The red-blind and green-blind versions demonstrate this very well. The big yellow cluster in the bottom-left corner of the red-blind version in figure 10 has fallen apart into three not necessarily immediately, but surely easier distinguishable clusters, also because the shape accentuate the colour by letting it happier lighter or darker.

The monochromatic view is also a nice example for the general effectiveness of shapes, but also a nice example for the weakness of similar shapes, which we can observe in the left half of the view.

In our opinion better ways to provide colour-blind-friendly visualizations would be to distribute the colour in a more sophisticated way to the clusters than we do it now. Technically it would be quite possible to assign cluster colours in a manner that similar colours are spread as far as possible from each other, since we first finish the overall clustering calculation before we start visualizing it. This means that all iterations and all states of the calculation are available at the moment visualization.

6 Discussion

Our application has many advantages and strengths. The main advantage is that the user has the opportunity of exploring and comparing different clustering algorithms in a wide range of different aspects. So one is able to compare different parameter settings, load in or create data sets and have full control of the area of interest.

Furthermore the tool is totally conform with Shneiderman's Mantra. So we give an overview first in the iterations tab with the small multiples, then the user can zoom and filter in the algorithm area and can have details on demand when hovering over cluster centroids.

The zoom is not an optical one, but it's like zooming into the algorithm itself. Meaning that our zoom option allows to slow down the visualisation steps not only down to iteration-wise steps, but point-wise steps, so that the user can reproduce the assignment of every point to a cluster. Another strong feature of our tool is the possibility of highlighting a whole cluster when hovering over its corresponding cluster center. As described in the previous chapters a major advantage of our tool is the colourblind mode, in which the user can see form-encoded clusters instead of just colour-encoded ones.

One of the main challenges, besides bug fixing, was the design choice, since it was very hard to include all controls and options in the application.Also we had to ensure that the application looks cool even on different screen aspect ratios. Our application is meant to run on monitors with screen aspect ratios between 1366*768 and 1920*1080.

Another hard issue was the linkage between the main algorithm area with the comparing one. It was challenging to synchronise the navigation of the two different windows without producing unexpected behaviour. After hours of testing and bug-fixing everything worked fine, but the navigation linkage was indeed no easy thing to implement.

7 Further work

There are a lot of extensions for our app which can be done in the future. We thought of giving the user the opportunity of exploring the algorithm implementation itself in an even deeper way.
Therefore we'd like to realise a pseudo code visualisation, where one can see in super slow motion how and why each point is assigned to a specific cluster. Furthermore we would like to implement some more partitioning-based algorithms like K-means++ or K-modes. A small but clever feature would be an option for saving and storing created data sets as well as animations.

Further things would be an implementation of more classes of clustering algorithms like density based algorithms with DBSCAN or OPTICS for example. Also a free web application would be possible.

8 Who did what

- Richard Paul: Visualisation techniques, app design meetings, visualisation implementation, tooltips, app navigation, algorithm parts, app design, bug fixing, colour-blindness analysis in the report.
- Severin Staudinger: Visualisation techniques, app design meetings, visualisation implementation, small multiples, algorithm navigation, algorithm parts, app design, major bug fixing, minor parts in report (discussion, further work), parts in tooltips, cluster highlighting, animations
- Michael Trimmel: Visualisation techniques, app design meetings, algorithms, report.

References

[1] O. Bachem, M. Lucic, H. Hassani, and A. Krause. Fast and provably good seedings for k-means. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., *Advances in Neural Information Processing Systems 29*, pp. 55–63. Curran Associates, Inc., 2016.

[2] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: enabling interactive multiple-view visualizations. In *VIS 05. IEEE Visualization, 2005.*, pp. 135–142, Oct 2005. doi: 10.1109/VISUAL.2005.1532788

[3] Z. He. Farthest-point heuristic based initialization methods for k-modes clustering. Dec 2014. doi: 10.1109/TVCG.2014.2346321

[4] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Mller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, Dec 2014. doi: 10.1109/TVCG.2014.2346321