universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Framework for Visual Cluster Analysis and Consensus Clustering“

verfasst von / submitted by

Christian Permann, BSc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree

MSc.

Wien, 2020 / Vienna, 2020	
Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet	A 066 921
Studienrichtung lt. Studienblatt:/ degree programme as it appears on the student record sheet	Masterstudium Scientific Computing
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Inform.Univ. Dr. Claudia Plant

Danksagungen

Ich möchte meiner Betreuerin Univ.-Prof. Dipl.-Inform.Univ. Dr.Claudia Plant für Ihre Unterstützung bei dieser Arbeit danken. Unsere offenen und konstruktiven Gespräche haben mir sehr geholfen diese Arbeit zielstrebig und motiviert zu vollenden.

Contents

List of Figures

List of Tables

Part I

Introduction and Preliminaries

Chapter 1

Introduction

The generation of clusterings is quite a difficult task when little knowledge on the data is available. There exist a large number of clustering algorithms that all aim to find a fitting partitioning of the data, but they all come with different assumptions on the data. Taking k-Means as an example, the data is assumed to be Gaussian distributed which might not actually be the case for many data-sets, making it unsuitable for clustering such data.

As a solution for this, visual frameworks have been proposed that aim to help finding the right clustering by evaluating the results of different algorithms and parameter settings via quality metrics and ranking the clustering solutions. One of these frameworks is Clustervision [11] whereby different algorithms are run, their solutions are visualized and ranked according to the average of five different quality metrics. The user is then able to look at the candidates through different visualizations and choose a solution as final result. A problem with this approach is though, that each metric is biased towards a specific resulting cluster structure, possibly ranking it higher than others even if it does not fit the data properly. Even the authors of Clustervision themselves mention that "the effectiveness of these metrics in gauging the quality of the clustering is also difficult to determine due to the lack of ground truth". Also, as many solutions were calculated and only one of them is chosen in the final decision, a lot of calculated knowledge is ignored.

To overcome this problem, I propose to not use quality metrics for the choice of the best clustering, but to rely on robustness indicators for this selection. As robustness is difficult to evaluate by itself many different algorithms with different parameters should be used, analyzing where multiple methods or parameter sets find a similar result. Clusterings that are similar to many other results can be seen as robust and finding a consensus of those can result in a better overall clustering. To find such similar clusterings or groups of similar clusterings meta-clustering can be used. To then compute final candidates, each group of similar results can be merged to a consensus result, capturing the essence of the group.

Based on this idea I created a tool which will be further described in Chapter 4. Additionally, the tool aims to include the expert user into the process of finding the result. Most research on consensus clustering, as also seen in the survey of Vega-Pons and Ruiz-Shulcloper [13], only briefly mention how generating or pre-selecting base clusterings can impact the result of consensus clustering. For this, my tool allows the user to visually explore the clusterings created by the simple clustering algorithms, facilitating the choice of which ones to merge for computing a final result.

The rest of this thesis is organized as follows:

Part II

Theory

Chapter 2

Related Work

2.1 Clustering Algorithms

A lot of research as been put toward the topic of clustering, resulting in a large number of different methods. [15]

2.2 Visual Clustering Frameworks

Clustervision [11]

VISTA [6] ClusterMap and user result evaluation

iVIBRATE [7] extension VISTA?

simple visualizations ELKI implementation [12]

Frameworks with specific data-sets in mind:

Propose: "An Evolutionary and Visual Framework for Clustering of DNA Microarray Data" [5], tool for clustering DNA data with visualization

Even Memes [8]

2.3 Consensus Clustering

consensus cluster plus [14]

Chapter 3

Methods

3.1 Clustering

3.1.1 OPTICS

3.2 Consensus Clustering

3.2.1 DICLENS

3.3 Other Methods

3.3.1 Hungarians Method

Part III

Implemented Tool

Chapter 4

Using the Tool

This chapter describes what a user can do with the newly implemented tool.

4.1 I/O and Data Visualization

bla

4.2 Selection of Algorithms and Parameters

bla

4.3 Meta-View and Consensus Clustering

bla

Chapter 5

Implementation

5.1 Programming Language and Tools

For the implementation of my tool my language of choice was Java. With Java there are lots of available implementations for different clustering methods available, including the Frameworks ELKI [4] and WEKA [9], as well as the machine learning library Smile [3]. To improve the performance in some parts of the tool Java 1.8 was chosen, making it the lowest required Java version to run it.This was done, as it enables the use of parallel streams which make it simple to parallelize functions. Additionally, it is easy to create custom visualizations using the low level draw() method of JComponents in Swing [2]. All graphs and plots were created using a custom draw() method as this leads to much better performance than using small components for the graphs.

5.2 File Input and Output

Loading and saving data is needed for the tool to make it more usable. For this reason it is possible to not only import data for clustering but also to save workflows and whole groups of clustering results for later reuse.

5.2.1 Clustering and Point Data

Besides being able to create new data sets from within the tool, importing data from CSV or ARFF files is possible. Those file types are supported as they are very common, especially CSV is usually used to store such information. ARFF is commonly used, as it is also the preferred data type of WEKA [9]. For this reason the ARFF importer of WEKA was adapted to load in these files such that they can be used with this tool.

When importing data, it is also possible to define an index of where the cluster labels are located. When this is done the tool will not load the specified column as data but use it to define clusters for the ground truth, which is then also handled differently by the different views of the tool.

The starting window, in which clusterings can also be loaded back into from the meta-view, also allows for exporting data to CSV. When doing this, the first column in the resulting file will contain the cluster identifiers starting with one. This way computed results can be exported and used in other tools or saved for later.

5.2.2 Clustering Workflows

As many runs of clustering algorithms with different parameter ranges can be performed and repeatedly defining this workflow can be tedious, the tool allows exporting and importing these workflows from the cluster workflow window. For this the underlying clustering objects are serialized and saved into .CWF (Cluster Workflow Files). It is important to note, that these files do not include the parameter settings for random seed or the meta clustering. This is intentional as I do not consider them to be part of the workflow and they should be set by the user for every run, if the default values are not satisfactory.

5.2.3 Pre-Clustered Data

Due to the clustering algorithms possibly being time consuming, either because of their complexity or just sheer number of runs, the results of these algorithms can be saved from the meta-view. To do this, the clustering results are serialized and saved as .CRF (Clustering Result Files). To load those files one can define the different parameters of the meta clustering and then import the .CRF. This way, only the meta clustering is computed again. The meta clustering is not saved in this file such that the user can more easily try different settings for it, by just saving the result and running the meta clustering again. As this is usually much faster than the clustering algorithms that generate the base clusterings, using this may save a considerable amount of time.

5.3 Methods and Interfaces

To ensure that new or missing methods can easily be added to the tool, all methods where a selection can be made from within the tool can be added to by implementing onto the corresponding interface. To add methods one needs to first implement it and if needed it's OptionsPanel and then add it to the corresponding static method found in the view using it.

The following sections describe which interfaces and standard methods are implemented, as well as which functions and classes must be edited to add a new custom methods.

5.3.1 Data Manipulation

Data manipulation can be performed as a first step in the starting window to create data-sets or modify them such that they can be used for clustering or visualization. It is also of relevance when loading clustering results back into the starting view to analyze results using dimensionality reduction techniques. For this step three interfaces are of interest:

- Generators: IGenerator
- Reducers: IReducer
- Normalizers: INormalizer

Generators

Generators allow the user to add or replace data points. Two initial implementations for generators are available, firstly the SinglePointGenerator which allows adding points to the data by clicking the scatter plot. When a point is added this way its coordinates are set to where the user clicked on the screen translated into the coordinates of the data space, such that the point appears below the cursor. Secondly, the ELKIGenerator is available which uses an XML-style input, defining how the clusters should be generated. This class uses the GeneratorXMLDatabaseConnection of ELKI [4] an is able to create data points according to different random distributions. The definitions file for this input can be found in ELKIs JAR file, their GIT repository [1] or in the /lib folder of the implementation.

Reducers

Reducers allow the user to manipulate the data such that the dimensionality or potentially (in the future) the number of points can be reduced. Per default three dimensionality reduction techniques are available. The simplest one, DimensionRemover, enables the user to just define a dimension by its index and delete it. This can be useful, if the data is not well cleaned beforehand and irrelevant or possibly disrupting columns are present in the data. For more advanced dimensionality reduction, the PCAReducer and tSNEReducer are available. The PCAReducer, which is adapted from the implementation from Smiles [3] uses Principal Component Analysis [CITATION] to reduce the dimensionality of the data to what the user defined. In contrast the tSNEReducer adapted from Jonsson [10] uses t-Distributed Stochastic Neighbor Embedding [CITATION] and also requires a perplexity parameter.

Normalizers

Normalizers provide functionality for normalizing the data points. This may be useful whenever different dimensions contain values in strongly differing ranges, while their concrete values do not have any direct relation to the other dimensions. Here, two default implementations are available, Normalize and Standardize. The Normalize class adapts the values such that in each dimension the range is set to the interval $[0, 1]$ while preserving relative distances. The Standardize class adapts these ranges such that the average is 0 and the standard deviation is 1.

Adding Methods

To additional Methods they must be implemented using the corresponding interface and added to the class StartWindow. To add them to this class the static methods initGenerators(), initReducers() and initNormalizers() must add the new method to the list of methods. As an example adding a new generator (newGeneratorX) would require adding "generators.add(new newGeneratorX());" to the function initGenerators().

5.3.2 Clustering Algorithms

Clustering Algorithms can be selected and tuned in the clustering workflow window such that the algorithm is used to create base clustering results for the meta-view. For additional information on the algorithms themselves, see Section [REFERENCE] The following interfaces are relevant for clustering:

- Clustering Algorithms: IClusterer
- ELKI Clustering Algorithms: IELKIClusterer
- Custom Clustering Algorithms: ICustomClusterer

Clustering Algorithms

The interface IClusterer is used as helper interface such that all clustering algorithms can use the same high level calls for compatibility. The actual implementations implement either IELKIClusterer if they use the ELKI database objects in their logic or ICustomClusterer if they use a data matrix. To further help not needing to re-implement generally needed methods the class AbstractClustering should also be extended by the actual implementation. With these class and interfaces the actual implementations can focus more on just the clustering logic.

ELKI Clustering Algorithms

bla bla bla

Custom Clustering Algorithms

bla bla bla

Adding Methods

bla bla bla

5.3.3 Meta-Clustering

The algorithm used for meta clustering is OPTICS [CITATION]. One of the requirements for this methods is that the used distance measure must be metric, otherwise the algorithm can only produce an approximate clustering at best. The distance measure used by the algorithm can be changed and additional distance measures can also be added. For this, the following interface is of interest:

- Clustering Distance Measure: IMetaDistanceMeasure

Clustering Distance Measure

The interface IMetaDistanceMeasure is used to define distance measures that can be used for OPTICS for meta clustering. These distance measures should represent how different two clustering solutions are from each other while needing to fulfill the requirements of a metric. For a measure (indicated with $d(x, y)$) to be considered metric the following conditions must be met:

- $d(x, y) \geq 0$: distances must be non-negative
- $d(x, y) = 0$ iff $x = y$: the distance between two objects is zero if and only if both objects are equal
- $d(x, y) = d(x, y)$: the distance must fulfill the symmetry condition
- $d(x, z) \leq d(x, y) + d(y, z)$: the triangle inequality must hold

The default implemented distance measures are Variation of Information and Clustering Error.

5.3.4 Consensus Clustering

bla bla bla

5.4 Visualization

To visualize the data in different ways and provide a functional Graphical User Interface (GUI) the Java Swing [2] Toolkit was used. It allows for both high level calls creating objects withing windows, e.g. panels, buttons or combo-boxes, as well as low level calls by directly changing the behavior of the draw() function. For the basic GUI the panels and buttons of the tool were created with high level calls, while the different graphs and plots were implemented by overriding the draw() function. Overriding the draw() function immensely improved performance as the already available high level objects are computationally costly, when needing many of them, by comparison.

5.4.1 Scatter Plot

bla bla bla

5.4.2 Switching between Clusterings

bla bla bla

5.4.3 Scatter Plot Matrix

bla bla bla

5.4.4 OPTICS Plot

bla bla bla

5.4.5 Heat Map

bla bla bla

5.4.6 Filter Panel

bla bla bla

5.4.7 Cluster comparison

bla bla bla

Part IV

Testing

Chapter 6

Experiments

6.1 Solutions unattainable from simple Clustering

bla

6.2 Multiple Solutions

bla

Part V

Concluding Thoughts

Chapter 7

Future Work

7.1 Improvements for Tool

bla

7.2 Research Consensus Clustering

bla

7.3 Visual Frameworks

bla

Chapter 8

Conclusion

8.1 Lessons learned

bla

8.2 Reflection of Work

In this article I showed

Appendices

Appendix A

Abstract

A.1 English abstract

Finding a good clustering solution for an unexplored data-set is a non-trivial task. Due to the large number of clustering algorithms which usually have lots of parameters, clustering results may differ strongly from each other and the underlying ground truth. With only little knowledge on the data the evaluation of which result best represents the underlying cluster structure is difficult. To find a fitting selection for the result, there exist visual frameworks that aim to simplify this choice by ranking the results according to quality measures. As those measures also have the downside of being biased towards specific structures (weather or not they fit the data) they are problematic for selecting a final result. For this reason, I propose to purely use indicators of robustness for the creation of a clustering result. This is done by meta-clustering results from different clustering algorithms and results and calculating consensus clusterings from each group of simmilar results. Additionally this process is supported through visualizations, giving the expert user the possibility to use his knowledge to further improve on the final result.

A.2 Deutsche Zusammenfassung

Eine gute Clustering Lösung für wenig erforschte Daten zu finden ist eine komplexe Aufgabe. Wegen der großen Anzahl an Clustering Algorithmen, welche meist auch viele verschiedene Parameter benötigen, können sich die Ergebnisse stark untereinander, aber auch von dem richtigen Ergebnis, unterscheiden. Mit nur wenig Wissen über die Daten ist auch die Evaluierung welches Ergebnis am nähersten zu der der unterliegenden Wahrheit, beziehungsweise am besten der Struktur der Daten entspricht eine schwere Aufgabe. Um eine solche Auswahl besser treffen zu können wurden visuelle Frameworks erschaffen, die mittels Qualitäts-Metriken die verschiedenen Ergebnisse bewerten und gereiht anzeigen. Da diese Metriken aber auch das Problem haben gewisse Strukturen in Ergebnissen zu bevorzugen zeigen sie sich wiederum bei der Entscheidung über das endgültige Ergebnis als problematisch. Aus diesem Grund schlage ich vor die Eigenschaft wie robust ein Ergebnis ist für die finale Entscheidung heranzuziehen. Um dies zu tun werden die Clusterings auf Meta-Ebene nochmals geclustert, wobei ähnliche Ergebnisse in einer Gruppe mittels Consensus Clustering zu einer Lösung zusammengeführt werden. Dieser Prozess wird weiters durch Visualisierungen unterstützt, so dass ein Experte mit Hilfe seines Wissens die Lösung möglicher Weise noch weiter verbessern kann.

Bibliography

[1] ELKI xml generator definitions file. `https://github.com/gaoch023/ELKI/blob/master/src/main/java/de/lmu/ifi/dbs/elki/application/GeneratorByModel.xsd`. Accessed: 2020-01-30.

[2] Java Swing package definition. `https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/package-summary.html`. Accessed: 2020-02-01.

[3] Smile - statistical machine intelligence and learning engine. `http://haifengl.github.io/`. Accessed: 2020-01-30.

[4] Achtert, E., Kriegel, H.-P., and Zimek, A. Elki: A software system for evaluation of subspace clustering algorithms. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management* (Berlin, Heidelberg, 2008), SSDBM '08, Springer-Verlag, p. 580–585.

[5] Castellanos-Garzón, J., and Díaz, F. An evolutionary and visual framework for clustering of dna microarray data. *Journal of Integrative Bioinformatics 10* (12 2013).

[6] Chen, K., and Liu, L. A visual framework invites human into clustering process. pp. 97 – 106.

[7] Chen, K., and Liu, L. Ivibrate: Interactive visualization-based framework for clustering large datasets. *ACM Trans. Inf. Syst. 24*, 2 (Apr. 2006), 245–294.

[8] Dang, A., Moh'd, A., Gruzd, A., Milios, E., and Minghim, R. *An Offline–Online Visual Framework for Clustering Memes in Social Media.* Springer International Publishing, Cham, 2017, pp. 1–29.

[9] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: An update. *SIGKDD Explor. Newsl. 11*, 1 (Nov. 2009), 10–18.

[10] Jonsson, L. t-SNE Java implementation. `https://github.com/lejon/T-SNE-Java`. Accessed: 2020-02-01.

[11] Kwon, B. C., Eysenbach, B., Verma, J., Ng, K., deFilippi, C., Stewart, W. F., and Perer, A. Clustervision: Visual supervision of unsupervised clustering. *IEEE Transactions on Visualization and Computer Graphics 24* (2018), 142–151.

[12] Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K. A., and Zimek, A. A framework for clustering uncertain data. *Proc. VLDB Endow. 8*, 12 (Aug. 2015), 1976–1979.

[13] Vega-Pons, S., and Ruiz-Shulcloper, J. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence 25* (May 2011), 337–372.

[14] Wilkerson, M. D., and Hayes, D. N. ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking. *Bioinformatics 26*, 12 (04 2010), 1572–1573.

[15] Xu, D., and Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science 2* (08 2015).