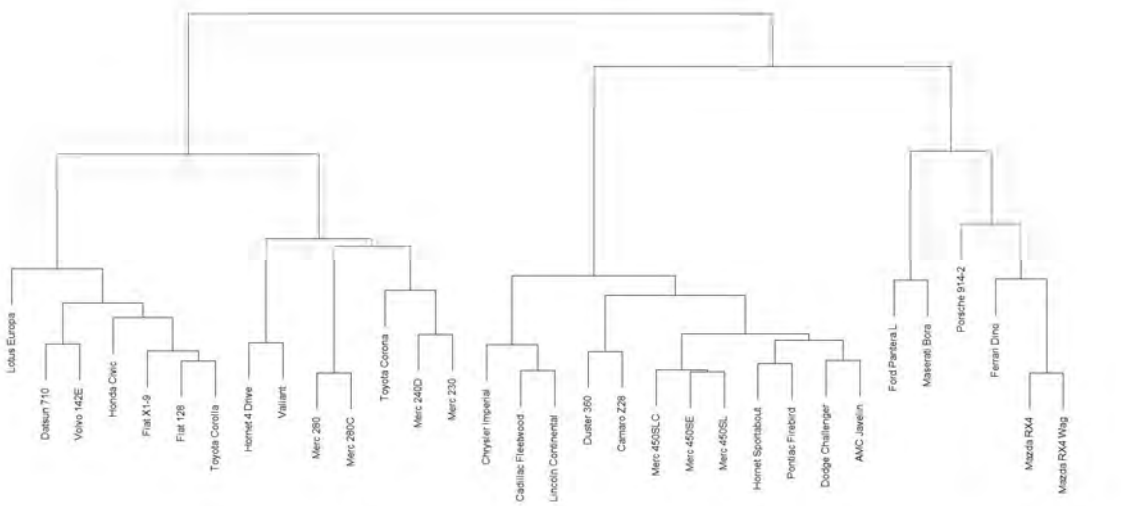
Clustering with optimised weights for Gower's metric

Using hierarchical clustering and Quasi-Newton methods to maximise the cophenetic correlation coefficient



By Jeroen van den Hoven.

Supervised by Eduardo Barbaro, Sandjai Bhulai, and Mark Hoogendoorn.

ABSTRACT

In this thesis, we design a weighting scheme for Gower's metric that optimises the cophenetic correlation coefficient (CPCC) of a hierarchical clustering. We use the limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm with bounds to optimise the weights of the Gower's metric. We validate our approach using both artificial and real datasets. We highlight that the weights are a crucial part of obtaining a more accurate clustering. Our algorithm increases the CPCC from 0.84 to 0.97, in the case of a dataset obtained from a mobile application. Furthermore, we can statistically confirm that our optimised hierarchical clustering algorithm performs better than all the benchmark algorithms tested in this thesis on multiple internal evaluation metrics. In addition to that, we extend our framework using an optimised K-medoids implementation with equally good results. Finally, we show that it is possible to analytically obtain a derivative of the CPCC when certain weights are set constant relative to each other.

ABSTRACT (NEDERLANDS)

In dit onderzoek formuleren we een algoritme om de gewichten van Gower's afstandsmaat te optimaliseren zodat de cophenetic correlation coefficient (CPCC) gemaximaliseerd wordt. We gebruiken het L-BFGS algorithme met eenvoudige restricties om de gewichten te optimaliseren. We valideren ons algorithm met behulp van kunstmatige en echte datasets. We concluderen dat de gewichten een vitaal onderdeel zijn bij het generen van een betere clustering. Ons algoritme verhoogt de CPCC van 0.84 naar 0.97 bij een dataset die gebaseerd is op het gebruik van een mobiele applicatie. Verder bevestigen we op statistische wijze dat ons geoptimaliseerde hierarchische clustering algoritme betere resultaten genereert dan onze benchmark algoritmes bij meerdere interne kwaliteitsmaten. Verder bewijzen we dat onze methode ook gebruikt kan worden voor een geoptimaliseerde K-medoids implementatie met vergelijkbare resultaten. Als laatste tonen we aan dat het mogelijk is om een analytische afgeleide van de CPCC af te leiden als we enkele gewichten constant houden ten opzichte van elkaar.

ACKNOWLEDGEMENTS

I am most grateful to Mirjana Starcevic and Esther Weusthof for the time they spent generating a largely pre-processed final dataset for me to work with. Furthermore, I would also like to thank Eduardo Barbaro, Sandjai Bhulai, and Mark Hoogendoorn for the time they spent supervising this project and discussing possibilities and results.

Table of Contents

1 INTRODUCTION

Using both numeric and categorical data in clustering algorithms can be a challenge since not every distance metric is capable of handling mixed data. Gower's metric is an example of a metric that is capable of combining numeric and categorical data. It also provides the opportunity to select weights for each individual variable, effectively altering the importance of each variable. These weights can be selected by the user, but often users set all weights to one. In this thesis, we use the weights of Gower's distance metric to optimise the cophenetic correlation coefficient of a hierarchical clustering. This introduction provides the reader with an introduction to the methods that are used in this thesis, the current usage of Gower's metric in clustering, and the possibilities of using weightings schemes. More information on these methods can be found in Sect. 2. We also provide our main research question and a general outline of the thesis.

Clustering is, in essence, the process of identifying representative groups in a dataset. These groups are called clusters. Each cluster represents a part of the objects that are clustered in such a way that each object is associated with one unique cluster. A successful clustering is one where objects within the same cluster tend to be similar to each other and dissimilar to objects from other clusters. For instance, if we were clustering animals, one cluster could represent the reptiles whilst another one could represent mammals. If we were clustering people, then one group might represent rich people, another the middle class and so forth.

We can visualise such a clustering with, for instance, a dendrogram. In a dendrogram, each object starts as its own leaf, a cluster of one object. Then the two clusters that are the most similar are joined to form one cluster. This step is repeated until we only have one cluster remaining. Objects that are considered to be similar are merged into one cluster at the bottom of the dendrogram, whilst dissimilar objects are merged at the very top of the dendrogram. An example of a clustering of cars can be found in the dendrogram of Figure 1. This uses the *mtcars* dataset (Henderson & Velleman, 1981), which contains information on 10 attributes of 32 cars, taken from the 197 *Motor Trend* magazine.

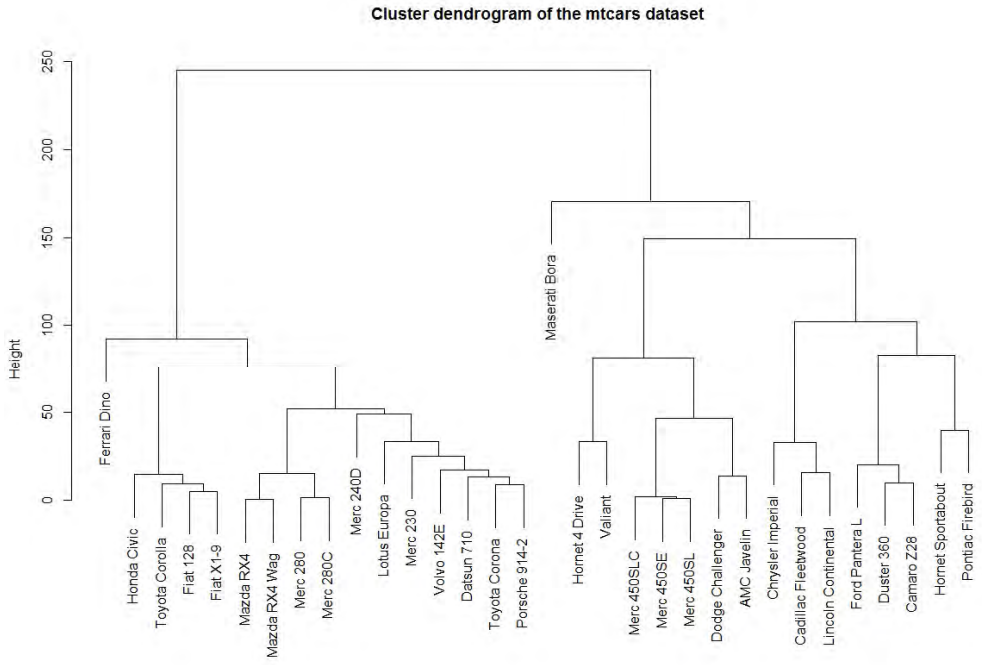


Figure 1: Cluster dendrogram of the mtcars dataset. All attributes were used in the clustering. We used average linkage and Euclidean distance.

The height in the dendrogram represents the distance / dissimilarity between two clusters when they were merged: the higher two clusters are merged, the more dissimilar they are. As an example, the distance between the Hornet Sportabout and the Pontiac Firebird is 35. Note that these distances have no units

and are only meaningful in this specific clustering setup in relation to the other distances in the dendrogram.

We see a few groups in this dendrogram: muscle cars like the Camaro Z28 and Pontiac Firebird, cheaper cars like the Honda Civic and Toyota Corolla, and older cars like the Porsche 914-2 and Toyota Corona. Furthermore, we see that the Mercedes 450SLC, 450SE, and 450SL are considered to be very similar since they are merged at a relatively low height. However, the Mercedes 230, 240D, 280, and 280C are merged with the other three Mercedes' at the very top of the dendrogram, meaning that their clusters are considered to be very dissimilar. We can also see that the Maserati Bora is considered to be very dissimilar to all other cars since it is merged at a relatively high height for the first time. This can indicate that it is considered to be an outlier. Outliers are not interesting in the same way as other clusters since they tend to not represent a large portion of the objects.

The clustering algorithm of merging smaller clusters until one large cluster is left is called agglomerative hierarchical clustering (Alpaydin, 2009). This algorithm is the main clustering algorithm in this thesis. Hierarchical clustering is not the only way to perform a clustering. Non-hierarchical clustering is another example of clustering. There are two well-known non-hierarchical algorithms, the first of which is K-means clustering. K-means clustering works by selecting k centres for k clusters at random. It then assigns each object to its closest centre and recomputes the centres of each cluster by taking the average for each variable. This process is repeated until the clusters do not change. To illustrate this we use the *iris* dataset, of which a scatterplot can be found in Figure 2. The *iris* dataset (Fisher, 1936) contains information on 150 flowers from three species of iris flowers. For each flower, we have five variables available: the species, petal length, petal width, sepal length, and sepal width. For this example, we use only the first three variables.

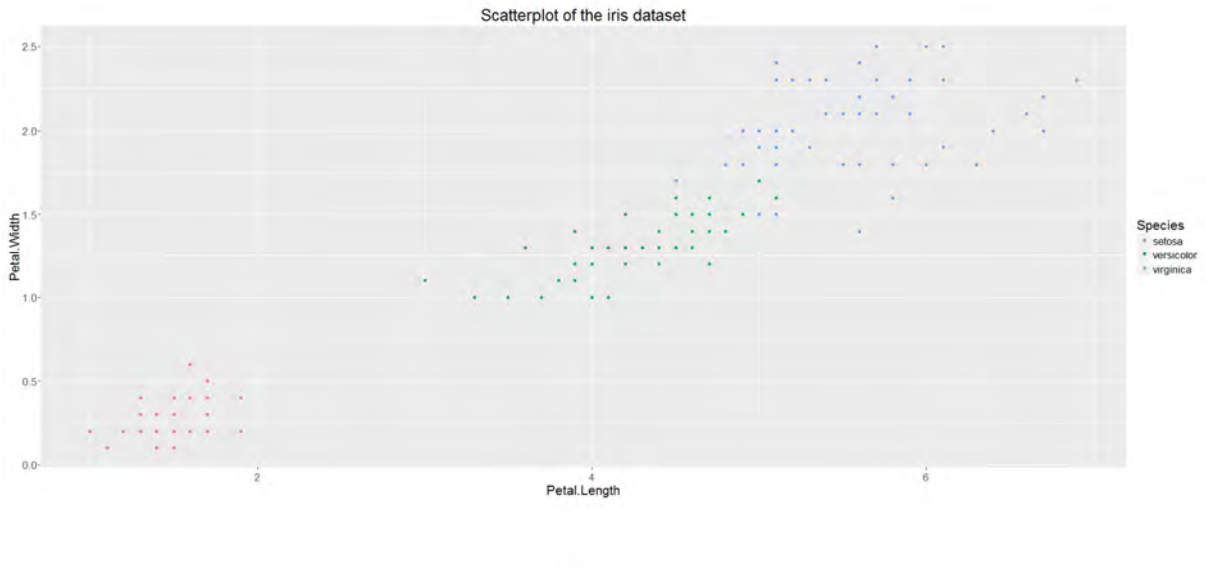


Figure 2: Scatterplot of petal length against petal width. The colours represent the different species of iris flowers. Some points may overlap since the measurements were rounded to the nearest millimetre.

We can see that the three species are grouped together based on the petal length and petal width with a low amount of overlap between species. We can cluster this dataset using the K-means (Alpaydin, 2009) approach and see how well the two figures match. The result can be found in Figure 3.

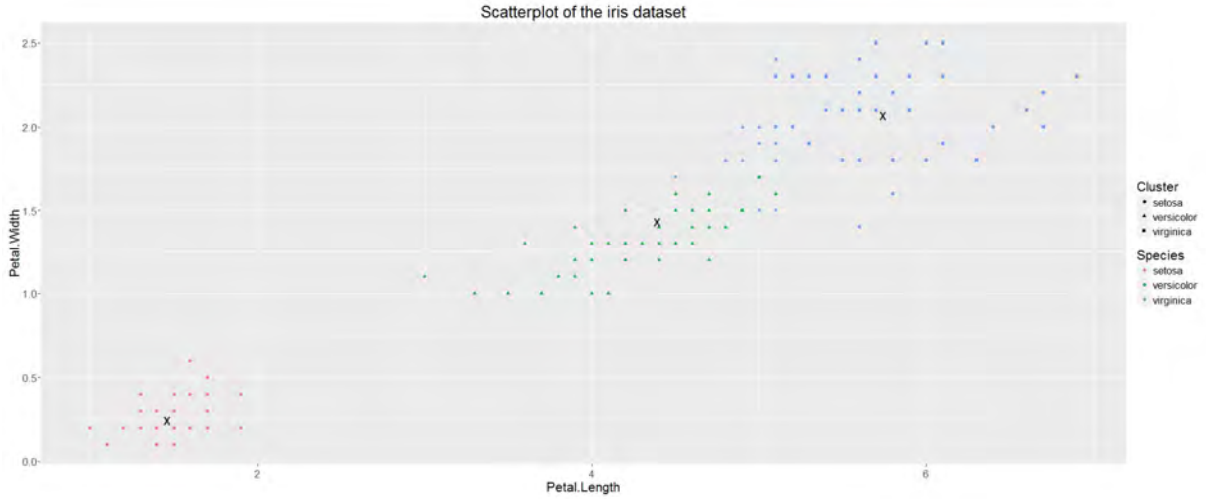


Figure 3: Clustering labels for K-means and the true labels. Clustering labels are represented by the shapes and the true labels are represented by colour. The cluster centres are denoted by the black crosses.

We can see from Figure 3 that the K-means algorithm is able to identify the different species well and does not misclassify many flowers. All *setosa* flowers form their own cluster. The only misclassifications occur at the border of the *Versicolor* (green triangles) and *Virginia* (blue squares) species. Here we see a couple of *Versicolor* flowers that have been classified as *Virginia* (green squares) and vice versa (blue triangles). The centre of each cluster is denoted by the black crosses.

Another well-known non-hierarchical algorithm is K-medoids. K-medoids (Theodoridis & Koutroumbas, 2006) uses a similar approach to K-means. The difference lies mainly in the selection of the centres. In K-medoids, the centre of a cluster is the object in that cluster that lies the closest to all the other objects. This algorithm is more robust to outliers than the K-means algorithm since one outlier can have a strong influence on the mean, but it will not have a strong influence on the distances between other objects. Executing the K-medoids algorithm on the *iris* dataset gives us the clustering from Figure 4. As we can see, the cluster centroids are now objects in our data, since the centroids overlap with at least one object. This has caused the centre of the *Versicolor* cluster to shift slightly to the right, causing a small number of objects to have switched between the *Versicolor* and the *Virginia* cluster. Note that some objects are still misclassified. The K-medoids algorithm seems to misclassify more *Virginia* flowers as V*ersicolor*, but fewer *Versicolor* as *Virginia*.

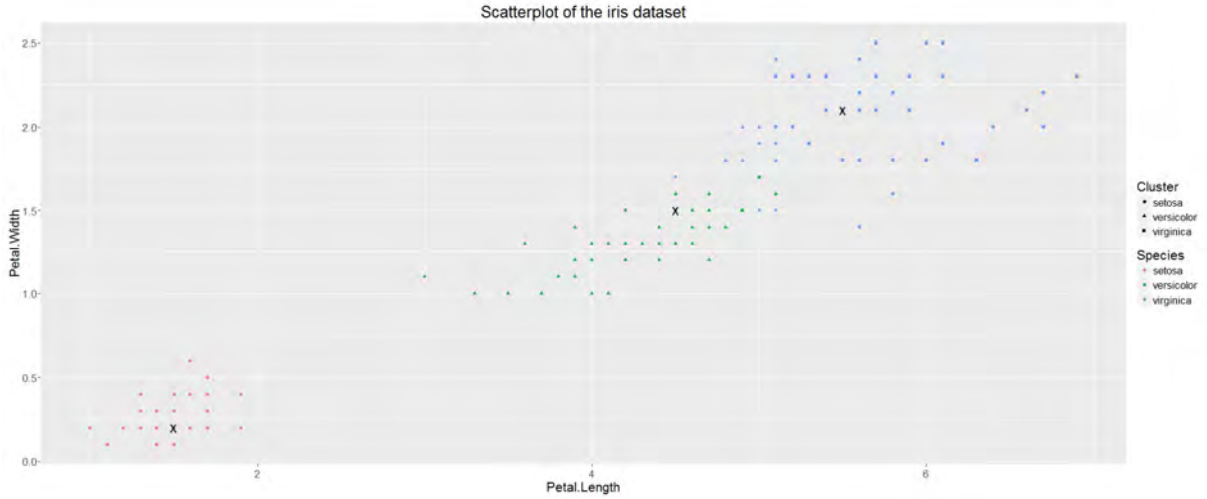


Figure 4: Clustering labels for K-medoids and the true labels. Clustering labels are represented by the shapes and the true labels are represented by colour. The cluster centres are denoted by the black crosses.

As shown in Figure 2, Figure 3, and Figure 4, clustering can be a good technique to identify existing groups in our datasets. This allows us to get more insight into our data, mainly when there is not much known about a dataset. A few other examples of the use of clustering is to cluster animals based on some of their characteristics, getting a clear insight into which animals are similar to each other based on those traits. The same could be done, for instance, for diseases/viruses, chemical compounds, and countries. A more business-oriented example of unsupervised clustering is getting insight into the different groups a company has as its customers.

However, we cannot cluster each dataset like we clustered the *mtcars* and *iris* datasets. Clustering algorithms work using dissimilarities or distances between objects, defined by a distance metric. A distance metric calculates the distance between two objects. If we cannot define the distance between objects, then we cannot perform the clustering. This issue can occur when not all data is of the same type. For instance, how does one define the distance between a user that is 1.80m long and has green eyes and someone else that is 1.70m long and has blue eyes? In this case, we have numeric variables (length) and categorical variables (eye colour). We cannot use a distance metric like ordinary straight line distance, also known as the Euclidean distance. This is not possible since we cannot subtract blue eyes from green eyes or square the result, which is needed for the Euclidean distance. Furthermore, using equal / not equal comparisons would work for the comparison on the eyes, but it would result in a huge loss of information for numeric values. This comparison would treat a difference of 0.01m the same as a difference of 10.000m. Clustering can be a great tool to get insights into data, but we need an algorithm, which can handle many types of data. This would allow us to use all relevant data that is available instead of just a subset, which could potentially omit important information.

There exist a few metrics that have been designed specifically for the purpose of clustering. One of these metrics is Gower's distance metric (Gower, 1971). It is defined as follows:

$$S_{ij} = \frac{\sum_{k=1}^{N} w_{ijk} s_{ijk}}{\sum_{k=1}^{N} w_{ijk}},$$

where:

- w_{ijk} : the weight for variable k between observations i and j.
- S_{ijk} : the distance between i and j on variable k.

This is, in essence, a weighted average of the distances on the different variables. The strength of Gower's distance metric lies in the calculation of S_{ijk}. S_{ijk} calculates the distance between i and j on variable k. Unlike traditional distance metrics, S_{ijk} does not apply the same formula to all variables. For categorical variables we use an equal / not equal comparison, but for numeric variables we use the absolute difference. To prevent one type of variable having more impact on the distance metric, all S_{ijk} are scaled to the range [0, 1]. For categorical variables, this means that we assign the value 0 to S_{ijk} when the categorical variables of i and j are equal and 1 when they are not. Numeric variables are scaled by dividing the absolute difference by the range of the variable. In formula notation this is denoted as follows, for the categorical and numerical variables respectively:

$$S_{ijk} = \begin{cases} 0 \; if \; X_{ik} = X_{jk} \\ 1 \; if \; X_{ik} \neq X_{jk} \end{cases},$$

$$S_{ijk} = \frac{|x_{ik} - x_{jk}|}{r_k},$$

where:

- $r_k = \max(x_{.k}) - \min(x_{.k})$
- X_{ik} = the value of variable k for object i.

Furthermore, new types of data can be added to Gower's distance metric by adding a new formula for S_{ijk} specifically for that type of data (Podani, 1999; Pavoine, et al., 2009). Gower's distance metric will be discussed further in Sect. 2.3.

Gower's metric allows us to assign a weight w_{ijk} to each individual variable, effectively changing the importance of that variable in the distance calculation. However, we did not find much literature regarding the selection process for these weights. Suggestions were made for using expert advice to choose the weights or an estimate of 'good' weights based on information known about the different variables, however, there does not seem to be a clear procedure to select good weights (Petchey & Gaston, 2009). In this paper, Petchey & Gaston (2009) discuss some agreements and disagreements they have had with a pair of researchers in their field regarding dendrograms and measures of function diversity. In their paper, they also discuss procedures on weighting variables. There, they mention that "*so little is known about appropriate or inappropriate trait weightings that further research seems appropriate, rather than outright rejection of any approach at this stage*" (page 2).

Even though there is very little information available as to how one should choose the weights, there are still a significant number of examples of people using Gower's metric in their clustering algorithms. However, in most cases they only mention that they set all weights equal to one, without providing an explanation (Cuadras, et al., 1998; Gavin, et al., 2003; Mouchet, et al., 2008; Gonçalves, et al., 2008; Ramos, et al., 2012), in other cases no mention on the selection of weights is reported (Thompson, et al., 2009). Some papers provide a short explanation (Schaffelke, et al., 2011) as to why the weights were set to one. Combining the observation of the trend of setting the weights to one with the findings from Petchey & Gaston (2009), we conclude that the option of setting all weights to one is often selected for lack of a clear better option.

However, we believe the selection of weights can be a valuable tool to get insight into the importance of different variables. We studied the influence of weights on clustering briefly in an earlier research (van den Hoven, 2015) and found that there is potential for improving the clustering in a proper weights selection procedure. Variables that can be used to separate the data into clusters could be assigned higher weights whilst variables that generate poorer separation could be assigned a lower weight. An example of why this could be useful is when this procedure results in some variables receiving a weight of 0, effectively removing them from the clustering. The result would be that we would not need to record these variables in the future, or at least not include them in the clustering. In general, it might help us distinguish between

important and less important variables and thus might even improve the quality of the clustering by removing the noise from less important variables.

Since there is potential for improving the clustering in selecting proper weights, it is logical that people have tried to develop a weighting scheme for Gower's metric. For instance, one could use the weights to balance the influence of categorical and numeric variables (Chae, et al., 2006). Another example is to choose the weights based on information from the variables (Montanari & Mignani, 1994). In this case, the weight for a categorical variable could be based on the number of possible values for that category or a weight could be set to 0.5 / 1 if the corresponding match is impossible / possible, instead of the original 0 / 1. The last example is important for variables where missing data can be present since we can make distinctions between objects that are very similar on a small selection of variables and objects that are similar on all variables. However, both these weighting schemes are based on assumptions about procedures that might work well for all datasets. This might result in good performance on some datasets but could also result in very poor performance on other datasets. For example, the weighting of a variable based on the number of levels would already be slightly more generalizable. However, considering that the results of (Montanari & Mignani, 1994) show that this weighting scheme is still more biassed than the standard Gower's metric with all weights set to one, this assumption seems to generalise poorly as well.

Whilst studying the literature we found another paper using a Quasi-Newton (Nocedal & Wright, 1999) method to optimise the weight vector to maximise their target function, the Mantel *r*, which is the correlation between two distance matrices (Rocha, et al., 2011). A Quasi-Newton method is used to find an approximation of the minimum of a function. More information on this method can be found in Sect. 2.8. This method requires that the target function is continuous, which may not always be a valid assumption given the nature of clustering algorithms. A small change in the weights would result in a relatively small change in the distances. However, a small change in the distances could result in a relatively large change in the clustering, causing the target function to not necessarily be continuous. This will be discussed further in Sect. 3.3. Though they did not fully discuss their method, the results from Rocha (2011) look promising with interesting implications on the importance of different variables. Since there is potential in selecting proper weights and since most people tend to set all weights to one, we want to develop a weighting scheme for Gower's metric that is not based on or at least relies less on external information or assumptions such as those mentioned previously.

Since Gower's metric can handle multiple types of data, we now need to be able to define the quality of a clustering. The quality of the clustering can be difficult to assess since we tend to have no information available as to what clusters exist and to which cluster each individual belongs. Since we tend to not know the cluster to which each user or object belongs a priori, we cannot use metrics which are designed to use information about the true cluster labels to determine the quality of the clustering. Instead of using these so-called external metrics, we often have to rely on metrics that do not use label information to determine the quality of a clustering. This makes performing unsupervised clustering a challenging process. For this purpose, we use the cophenetic correlation coefficient (Rohlf, 1962), CPCC in short. The CPCC allows us to calculate the correlation between a distance matrix and the corresponding dendrograms distance matrix. Note that the CPCC is thus based on a hierarchical clustering. This coefficient allows us to estimate how well the hierarchical clustering represents the distances between different objects in the data. Further details regarding the CPCC will be discussed in Sect. 2.4.

Our goal is to combine Gower's metric with the CPCC to design an algorithm that will be able to handle many types of variables and optimises the CPCC through selection of weights for Gower's metric. To summarise:

In what way should one choose the weights of Gower's distance metric to optimise the CPCC of the corresponding hierarchical clustering?

We are also interested in the influence of an optimised set of weights on other metrics that do not use the cluster labels, such as the silhouette and the within / between sum of squares ratio, which will be

discussed further in Sect. 2.6. We perform statistical tests to determine if the difference in these metrics of our optimisation algorithm is significantly better than the same metrics calculated for standard hierarchical clustering and K-medoids clustering.

To answer our research question, we have divided this thesis into the following sections: we provide further background for the rest of this thesis in Sect. 2. We discuss how we chose the parameters, optimisation techniques, and general outline of our algorithm in Sect. 3. We provide results on six artificial datasets and five real datasets, both in terms of the CPCC as well as the silhouette and within / between sum of squares ratio in Sect. 4. We report properties of the final dataset that we use for the validation of our algorithm and on the data pre-processing step in Section 5. The results from the tests on the final dataset can be found in Sect. 6. In Sect. 7 we develop a similar algorithm to our optimised hierarchical clustering algorithm for the K-medoids algorithm and provide results and comparisons to the hierarchical algorithm. In Sect. 0 we show that it is possible to assign the same weight to different variables and still use our hierarchical optimisation algorithm. Finally, we draw our conclusions in Sect 9 and discuss the results and provide recommendations in Sect. 0.

2 BACKGROUND

In this section, we present a short summary of the terms and techniques used in this thesis. Before we present the theoretical background of this thesis, we define the most common terminology that is used throughout this thesis.

- Instance: The set of measurements corresponding to one object that we want to cluster. This corresponds to one row in a tidy database, if the columns represent the variables (Wickham, 2014).
- Clustering: The process of identifying groups in a dataset in such a way that instances in the same group are more similar than those in different groups.
- Cluster: A group of similar instances.

2.1 NON-HIERARCHICAL CLUSTERING

K-means

K-means clustering is one the non-hierarchal clustering algorithms that we use. The algorithm selects K centres for K clusters and assigns each instance to the closest cluster. It then recomputes the centre of each cluster by taking the average for each variable of all instances that are part of the cluster and repeats the process (Alpaydin, 2009). We present the algorithm here:

1. Initialise K vectors M_i, $i \in \{1, 2, \ldots, K\}$, representing our K clusters. This can be done at random, by choosing K different instances from our original dataset, or another procedure.
2. Until we achieve convergence, do the following:
    2.1 Assign each instance X_i to its nearest cluster centre.
    2.2 Recompute the cluster centre for each cluster by averaging all instances in that cluster.

The assignment of instances to clusters is done in the following way:

$$b_{i,j} = \begin{cases} 1 \; if \; d(X_i, M_j) = \min\limits_{k \in \{1,\ldots,K\}} d(X_i, M_k) \\ 0 \; otherwise \end{cases},$$

Where $b_{i,j}$ indicates if instance X_i belongs to cluster M_j. $d(X_i, M_j)$ is the distance between cluster M_j and instance X_i. Recomputing the new clusters is done in the following way:

$$M_j = \frac{\sum_{i=0}^{K} b_{i,j} X_i}{\sum_{i=0}^{K} b_{i,j}}.$$

The algorithm converges when M_j has stabilised for each $j \in \{1, \ldots, K\}$, or when a maximum number of iterations have been completed. It has stabilised when $b_{i,j}$ does not change for any instance. K-means has a time complexity of $O(n^2)$. K-means is not the only non-hierarchical clustering algorithm: K-medoids is another example.

K-medoids

K-medoids is a non-hierarchical technique similar to K-means. However, instead of using the centre of a cluster, it uses the centroid. The centroid is defined as the instance that has the lowest average distance to each other instance in the cluster. Contrary to K-means, K-medoids can be used for datasets where the Euclidean distance is not defined, making it a better comparison algorithm in our situation. One way to optimise the selection of k clusters is to use the Partitioning around Medoids algorithm from (Theodoridis & Koutroumbas, 2006):

1. Initialise k of the n instances as the first medoids.
2. Associate each instance to its closest medoid.

3. While the total cost decreases:
    a. For each medoid *m* and non-medoid *o*:
        i. Swap *m* and *o*, recompute the costs.
        ii. If the total costs of the clustering increased, undo the swap.

2.2 HIERARCHICAL CLUSTERING

Apart from non-hierarchical algorithms, we also use hierarchical algorithms. As for the hierarchical clustering, we only look at agglomerative hierarchical clustering (Alpaydin, 2009). This algorithm combines at each combination step the two clusters that are the closest to each other until only one large cluster remains. By tracing the way in which the final cluster was formed, we can see how different instances are related to each other. Hierarchical clustering is slower than K-means since it has a time complexity of $O(n^3)$. The function used to define the distance between clusters is called the linkage function, of which we consider the following selection:

Single linkage

Single linkage selects the distance between the closest two members of a cluster as the distance between those clusters (Legendre & Legendre, 1998):

$$d(G_i, G_j) = \min_{X^r \in G_i, X^s \in G_j} d(X^r, X^x),$$

where:

- $d(x,y)$: The distance between instance x and y, depending on some distance measure.
- G_i : Cluster i.
- X_i : Instance i.

Complete linkage

Complete linkage selects the distance between the furthest two members of a cluster as the distance between those clusters (Sorensen, 1948):

$$d(G_i, G_j) = \max_{X^r \in G_i, X^s \in G_j} d(X^r, X^x),$$

Average linkage

Average linkage selects the average distance between all pairs of members of the clusters as the distance between those clusters (Sokal & Michener, 1958):

$$d(G_i, G_j) = \frac{1}{|G_i||G_j|} \sum_{X^r \in G_i, X^s \in G_j} d(X^r, X^x),$$

where $|G_i|$ is the size of cluster i.

Centroid linkage

Centroid linkage selects the distance between the centroids of the clusters as the distance between those clusters (Sokal & Michener, 1958):

$$d(G_i, G_j) = d(G_{C,i}, G_{C,j}),$$

where $G_{C,i}$ is the centroid of cluster G_i. As mentioned in the hierarchical and non-hierarchical algorithms, an important part of performing a clustering is being able to define a distance between two instances. For this we, need a distance metric.

2.3 GOWER'S DISTANCE METRIC

An important part of finding a clustering on a dataset with variables of mixed types is finding a distance metric that is capable of handling different types of variables, such as categorical and numeric. Gower's distance metric is capable of doing this by calculating the components of the distance between two instances X_i and X_j differently for each variable. For instance, take two instances X_i and X_j with both two variables, denoted by X_{ik} and X_{jk} for $k \in \{1, 2\}$. Assume the first variable is categorical and the second is numeric. For the first variable, the categorical variable, the difference between the values of X_{ik} and X_{jk} is defined as an indicator function (Gower, 1971):

$$S_{ijk} = \begin{cases} 0 \; if \; X_{ik} = X_{jk} \\ 1 \; if \; X_{ik} \neq X_{jk} \end{cases}.$$

For the second type of variable, the numeric variable, the difference between the values of X_{ik} and X_{jk} is defined as:

$$S_{ijk} = \frac{|x_{ik} - x_{jk}|}{r_k}.$$

Here, r_k is defined here as the range of variable k, $\max(x_{.k}) - \min(x_{.k})$. These two types of variables are the only ones that we discuss here, since they are the only relevant ones for this thesis. Gower's metric is capable of dealing with other types of variables (Podani, 1999; Pavoine, et al., 2009).

The next step is to combine these S_{ijk} values into Gower's metric. This is done in the following way:

Equation 1: Gower's metric

$$S_{ij} = \frac{\sum_{k=1}^{N} w_{ijk} S_{ijk}}{\sum_{k=1}^{N} w_{ijk}},$$

where:

- w_{ijk} : the weight for variable k between observations X_i and X_j.
- S_{ijk} : the difference between X_{ik} and X_{jk}.

It is important to note that we use $w_{ijk} = w_k$ when S_{ijk} is defined, effectively assigning one weight per variable. If S_{ijk} is not defined, for instance, because there are missing values in the data, then w_{ijk} is equal to 0.

2.4 COPHENETIC CORRELATION COEFFICIENT (CPCC)

One problem of clustering is that there are no predefined clusters. This makes the process of determining whether or not a clustering is good difficult. To be able to distinguish between clusterings we need an evaluation method.

As discussed in the introduction, we can visualise a hierarchical clustering. Such a visualisation is called a dendrogram. A dendrogram is a tree-like structure showing the relationship between different clusters by following the process through which the clustering was generated. An example can be found below in Figure 5. Instances that were merged early in the clustering process will be merged at a low heights and vice versa. This representation can tell us how well the data was clustered: when we observe clusters with a relatively low intra-cluster height compared to their inter-cluster height, then these clusters may be real clusters.

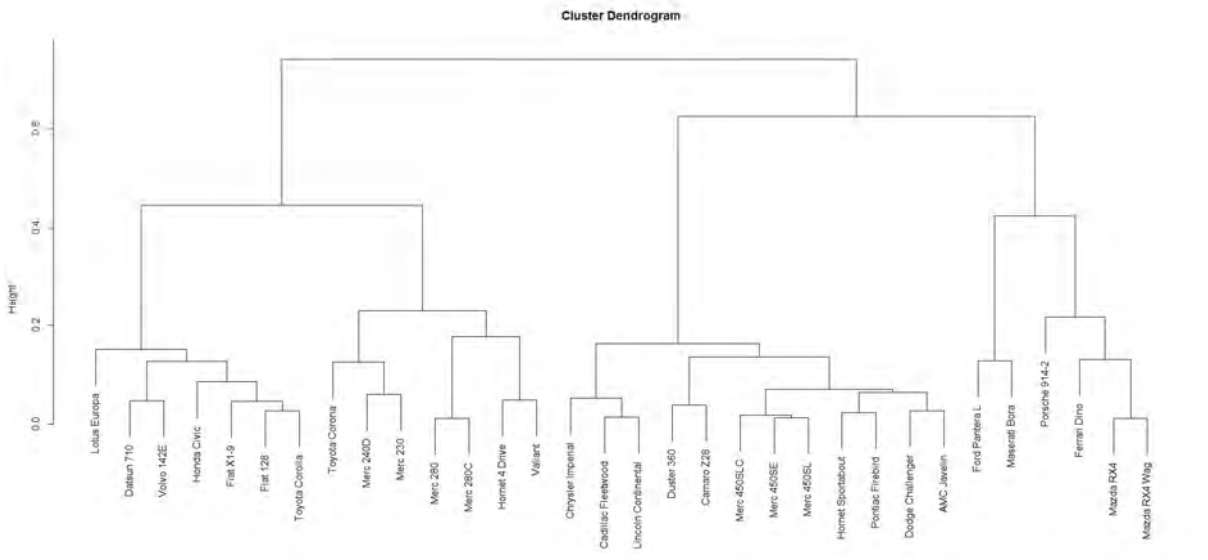


Figure 5: Cluster dendrogram of the mtcars dataset. All attributes were used in the clustering. We used average linkage and Gower's distance metric.

A measure that uses the dendrogram to calculate the quality of the hierarchical clustering is the cophenetic correlation coefficient (Rohlf, 1962), the CPCC. This is a measure of how well a dendrogram matches the distance matrix, the matrix containing the distances between each pair of observations, which was used to construct the clustering. The CPCC is defined as the correlation between the underlying distance matrix and the distance between instances in the dendrogram, the cophenetic distance (Sokal, 1973).The cophenetic distance between two instances is defined as the height in the dendrogram where two instances are joined for the first time. For instance, in Figure 5 the cophenetic distance between the Datsun 710 and the Honda Civic would be 0.15. The formula for the CPCC is:

Equation 2: CPCC

$$c = \frac{\sum_{i<j}(x(i,j)-\bar{x})(t(i,j)-\bar{t})}{\sqrt{(\sum_{i<j}(x(i,j)-\bar{x})^2)(\sum_{i<j}(t(i,j)-\bar{t})^2)}},$$

where:

- $x(i,j)$: the Euclidean / Gower's distance between instances i and j.
- $t(i,j)$: the distance between instances i and j in the dendrogram, defined as the height in the dendrogram where the two instances are joined for the first time.
- $\bar{x}$: the average Euclidean / Gower's distance between instance.
- $\bar{t}$: the average distance between instances in the dendrogram.

The fit is deemed reasonably good if the cophenetic correlation coefficient lies in the range [0.7, 0.8] on a scale of [0, 1], good when it is in the range (0.8, 0.9] and very good for any value larger than 0.9 (Rohlf, 1988). We also look at the index of agreement (IoA) as a possible alternative to the CPCC.

2.5 INDEX OF AGREEMENT (IOA)

The index of agreement, presented by (Willmot, 1981), is a measure to determine how well a model fits the corresponding data. According to Willmot the index of agreement "*is not a measure of correlation or association in the formal sense, but rather a measure of the degree to which a model's predictions are error free.*". It is defined as:

Equation 3: IoA

$$d = 1 - \frac{\sum_{i=1}^{N}(P_i - O_i)^2}{\sum_{i=1}^{N}(|P_i^*| + |O_i^*|)^2},$$

where:

- P_i : the predicted value of instance *i*.
- O_i : the observed value of instance *i*.
- P_i^* : $P_i - \bar{O}$
- O_i^* : $O_i - \bar{O}$

Here we use the cophenetic distances as the predicted values and the Gower's distances as our observed values. An index of agreement of 1 indicates perfect agreement, whilst a value of 0 indicates complete disagreement.

The CPCC and IoA show us how well the dendrogram fits the data, but it does not tell us how many clusters we should select. For this, we will be looking at some selection criteria for the number of clusters.

2.6 SELECTION CRITERIA

Once we have our weights and hierarchical clustering, we use the silhouette and the within-between sum of squares ratio (WB-ratio) to determine the number of clusters.

WB-ratio

The WB-ratio is the ratio between the within-cluster sum of squares and the between sum of squares. The sum of squares is defined as the sum of squared errors. In clustering, this is the squared distance between an instance and the centre of its cluster. A low WB-ratio indicates that the intra-cluster distances are relatively small compared to the inter-cluster distances. Therefore, a clustering with a low WB-ratio can be considered to be better than a clustering with a high WB-ratio. An explanation including a visualisation will be presented at the end of this section.

Silhouette

Another criterion to decide whether or not a clustering is good is the (average) silhouette (Rousseeuw, 1987). The silhouette is a measure of how well an instance is matched to its own cluster compared to the closest other cluster. By looking at the average silhouette over all instances, we can determine whether or not the current clustering is appropriate. By doing this for multiple different numbers of clusters, we can determine a good estimate for the number of clusters.

We define the following variables:

- a_i = average dissimilarity of instance *i* to all other objects in its own cluster. This variable has value 0 for a cluster of size 1.
- $d_{i,c}$ = average dissimilarity of instance *i* to all other objects in cluster *c*.
- b_i = $\min_{C \neq A} d_{i,C}$.

The silhouette s_i of instance *i* is then as follows:

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}.$$

We can see that:

$$-1 \leq s_i \leq 1.$$

The meaning of s_i can be determined from the definition of the variables:

1. If s_i is close to 1, then a_i is much lower than b_i, indicating that instance *i* is assigned to the proper cluster.

2. If s_i is close to -1, then a_i is much higher than b_i, indicating that instance i is assigned to the wrong cluster.
3. If s_i is close to 0, then a_i is approximately equal to b_i, indicating that it is unclear to which cluster instance i should be allocated.

By looking at the average silhouette S we can determine whether or not instances have been properly assigned to a cluster. This can be used to determine the number of clusters by computing multiple clustering configurations using different numbers of clusters. Then we compute the average silhouettes for each possible number of clusters. We can then select the appropriate number of clusters based on the value of the average silhouette and the number of clusters, selecting one where the average silhouette is high. This procedure is visualised in Figure 6:

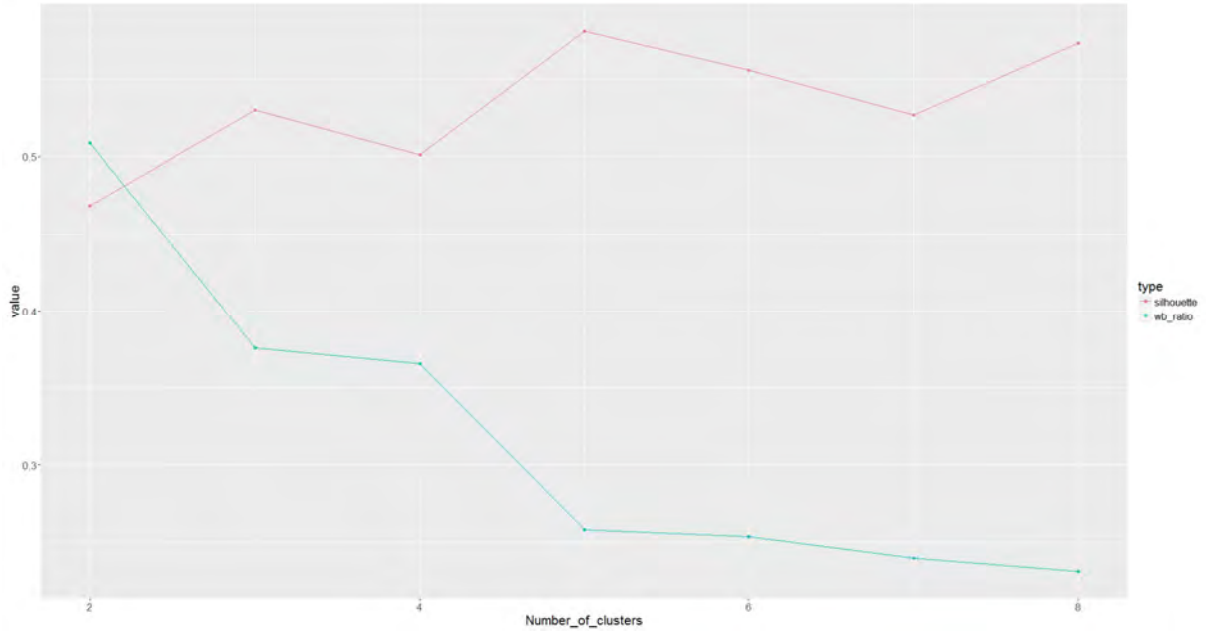


Figure 6: the WB-ratio and the average silhouette for the mtcars dataset.

We see that the WB-ratio keeps decreasing but stops decreasing as fast as before starting at $k = 5$. This is also where the silhouette achieves its highest value of 0.58. Since both of these selection criteria indicate that $k = 5$ is a good choice for the number of clusters, we conclude that we should select $k = 5$.

Since we are developing a new clustering algorithm, we want to determine if our algorithm is significantly better than existing algorithms. For this, we will be using the Quade test.

2.7 SIGNIFICANCE TESTS

To test for significant differences between different clustering algorithms based on our statistics, we use the Quade test. The advantage of this test is that it is based on the ranks R_{ij} of the data, not on the actual data. This makes this test non-parametric since it does not assume the normality of the underlying data. The Quade test is first of all used to determine if there is a significant difference between the ranks of any one pair of different clustering algorithms. The posthoc Quade test is then used to determine if there is a significant difference between the optimised hierarchical algorithm and the benchmark algorithms.

In all cases, the data used for the test looks as follows. Each column contains the results of one clustering algorithm and each row contains the results of one dataset (Sect. 4) or a number of clusters (Sect. 6). We calculate the rank of each row by assigning the value 1 to the lowest value in that row, 2 to the second lowest etc. until all entries of the row have a value. Ties are resolved by averaging the ranks of the ties. For instance, a tie on rank 2 and 3 would result in a rank of 2.5 for both entries (Friedman, 1937).

Quade test

The Quade test can be used to determine if there is a significant difference between any set of variables. For this purpose, the Friedman test (Friedman, 1937) could also be used, but the Quade test is deemed more powerful when we have less than four columns (Pohlert, 2016). Apart from the ranks $R_{i,j}$ it also needs the ranks of the ranges of each row, Q_i. This gives rise to the following formulae:

$$S_{i,j} = Q_i * (R_{i,j} - \frac{(p+1)}{2}),$$

$$S_j = \sum_{i=1}^{n} S_{i,j},$$

$$The\ test\ statistic\ \hat{F} = \frac{(n-1)B}{A-B} \sim F_{p-1,(n-1)(p-1)},$$

where:

- $A = \sum_{i=1}^{n}\sum_{j=1}^{p} {S_{i,j}}^2$
- $B = \frac{1}{n}\sum_{j=1}^{p} {S_j}^2$
- p = the number of columns.
- n = the number of rows.
- $\bar{r}_j$ = the mean rank of column j.

The H_0 hypothesis is that the p groups originate from the same distribution, whereas H_a is that at least one group differs from another. The posthoc Quade test uses the following inequality to determine whether or not to reject the hypothesis that there is no significant difference between columns i and j:

$$\left|S_i - S_j\right| > t_{1-\alpha/2,(n-1)(p-1)}\sqrt{\frac{2b(A-B)}{(n-1)(p-1)}},$$

which has a student-t distribution (Pohlert, 2016).

Holm's p-value correction

To reduce the number of type 1 errors when doing posthoc tests it is necessary to use a p-value correction method for the Quade and Conover tests. The probability of this occurring with the Nemenyi test is relatively low since it effectively already uses a correction method. In this thesis, we use Holm's method (Holm, 1979). This method works as follows:

1. Compute the p-values of the tests you want to use.
2. Sort them in ascending order.
3. Set $k = 1$.
4. While we do not reject or when we rejected all null hypotheses, do:
    a. Determine $R^{(k)} \leq \frac{\alpha}{n-k+1}$. *Equation 4*
    b. If Equation 4 holds, reject the null hypothesis of the test corresponding to R(k), increase k by 1 and continue.
    c. If Equation 4 holds, do not reject any more null hypotheses and stop the procedure.

Where:

- $R^{(k)}$ = the kth p-value of the sorted list of p-values.
- α = the significance level.
- n = the number of tests done.

In this thesis, we use this correction method for Quade's and Conover's tests. We will report the value of $R^{(k)} * (n - k + 1)$, so that we can compare all adjusted p-values to our chosen α but still draw the same conclusions.

Now we have all elements that we need to compute our clustering, determine its quality, and compare the results to other algorithms. The last thing we need is a way to optimise the weights of Gower's metric.

2.8 BOUNDED LIMITED-MEMORY BROYDEN–FLETCHER–GOLDFARB–SHANNO ALGORITHM (L-BFGS-B)

We attempted to find an optimal set of weights for Gower's metric through random search in a previous research (van den Hoven, 2015). However, this procedure tends to be time-consuming and does not guarantee a set of weights that corresponds to a high CPCC. A way to optimise this methodology is to use a Quasi-Newton algorithm. These algorithms try to find the minimum of a function using the first and second order derivative. The method that we use is called the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with bounds (L-BFGS-B) (Nocedal & Wright, 1999). This method is a modification of the BFGS algorithm. We start by describing the BFGS method and continue to the modifications that the L-BFGS-B method makes.

BFGS method

The BFGS method is used to find the minimum of a function. We can use this algorithm to find a maximum by multiplying our target function *f* with -1. In our case, we would use the CPCC or the IoA as our target function. The algorithm (Nocedal & Wright, 1999) works as follows. Given a starting point x_0, convergence tolerance $\varepsilon > 0$, inverse Hessian approximation $\boldsymbol{H_0}$:

1. $k = 0$
2. $while\ \left(\left|\left|\nabla f_k\right|\right| > \epsilon\right)$
   a. Compute search direction $p_k = -H_k * \nabla f_k$
   b. $x_{k+1} = x_k + \alpha_k * p_k$
   c. $s_k = x_{k+1} - x_k$
      $y_k = \nabla f_{k+1} - \nabla f_k$
   d. $H_{k+1} = (I - \rho_k s_k {y_k}^T) H_k (I - \rho_k y_k {s_k}^T) + \rho_k s_k {s_k}^T$
   e. $k = k + 1$

where:

- f : the function to optimise.
- ∇f_k : the gradient of f at step k.
- $\left|\left|f\right|\right|$: the (Frobenius) norm of f.
- H_k : the Hessian approximation at step k.
- x_k : the estimate for the optimum at step k.
- ρ_k : $\frac{1}{{y_k}^T s_k}$.
- I : the identity matrix.

For more information about the Frobenius norm, see (Nocedal & Wright, 1999). The main advantages of this method are that each iteration can be performed at a cost of $O(n^2)$ and it converges relatively quickly. Furthermore, the method is robust, meaning that small changes in the function will not result in major changes in the output. For our case n is the number of variables in our dataset.

L-BFGS

Memory issues may occur since it is likely that we want to cluster using many variables. This is why we decided to use a limited memory version of the BFGS method. Another reason is that this version of the algorithm is implemented in **R**'s *optim* function which allows the use of bounds on the variables, which we need for our target function. The core of this function is C code, so there is very little chance that we will be able to implement our own algorithm and have it outperform this premade function. The limited memory algorithm (Nocedal & Wright, 1999) works as follows:

Given a starting point x_0, convergence tolerance $\varepsilon > 0$, and number of vectors to store m:

1. $k = 0$
2. $while\ (||\nabla f_k|| > \epsilon)$
    a. $Choose\ a\ H_k^0$
    b. $Compute\ search\ direction\ p_k = -H_k * \nabla f_k$
    c. $x_{k+1} = x_k + \alpha_k * p_k$
    d. $if\ k > m\ remove\ the\ vector\ pair\ \{s_{k-m}, y_{k-m}\}\ from\ storage$
    e. $s_k = x_{k+1} - x_k$
    f. $y_k = \nabla f_{k+1} - \nabla f_k$
    g. $k = k + 1$

The algorithm to compute $\boldsymbol{H_k}$ is the following:

1. $q = \nabla f_k$
2. $for\ i = k-1, k-2, \dots, k-m$
    a. $\alpha_i = \rho_i {s_i}^T q$
    b. $q = q - \alpha_i y_i$
3. $r = H_k^0 q$
4. $for\ i = k-m, k-m+1, \dots, k-1$
    a. $\beta = \rho_i {y_i}^T r$
    b. $r = r + s_i(\alpha_i - \beta)$
5. $Return\ H_k \nabla f_k = r$

where:

- f : the function to optimise.
- ∇f_k : the gradient of f at step k.
- $||f||$: the (Frobenius) norm of f.
- H_k : the Hessian approximation at step k. In this method it is calculated according to the algorithm that follows.
- x_k : the estimate for the optimum at step k.
- I : the identity matrix.
- ρ_k : $\frac{1}{{y_k}^T s_k}$.

One method to choose $\boldsymbol{H_k^0}$ proposed by (Nocedal & Wright, 1999) is the following:

$$H_k^0 = \frac{{s_{k-1}}^T y_{k-1}}{{y_{k-1}}^T y_{k-1}} I.$$

Since we used the **R** built-in function to execute the L-BFGS algorithm, we are not sure if this method is used to choose $\boldsymbol{H_k^0}$.

3 METHODS

In this section, we discuss the properties of the methods we use and their influence on each other. The goal of this is to know which settings are best suited for our goal of optimising the clustering. We analyse Gower's metric (Gower, 1971), which is useful for making a selection in the target function that we wish to optimise. We look at two possible candidates for this function: the CPCC (Rohlf, 1962) and the IoA (Willmot, 1981). We continue by analysing the influence of the linkage function on our target function and choosing a preferred linkage function, which we use as the default setting for our models. Furthermore, we define our optimisation algorithm and try to extend this algorithm such that it is able to find global optima of the weight vector. Finally, we look at the influence of the bounds on the target function and the difference in weights, followed by a short summary of our algorithm setup.

3.1 ANALYSIS OF GOWER'S METRIC

We study Gower's metric to make sure we do not overlook any properties which might be important for the optimisation of the weights. When studying Gower's metric and the CPCC in two dimensions we came across an interesting pattern for the CPCC in two dimensions. Figure 7 demonstrates this behaviour. It is a level plot of one of the artificial datasets we use for the validation in Sect. 4, the *cluster* dataset. In this level plot, the value of the CPCC is visualised as a function of the weights for Gower's metric. The x and y-axis represent the weights for the first and second variable respectively. The colour represents the value of the CPCC. The formulae for these metrics can be found in Sect. 2.4, Equation 2 and Sect. 2.3, Equation 1 respectively.

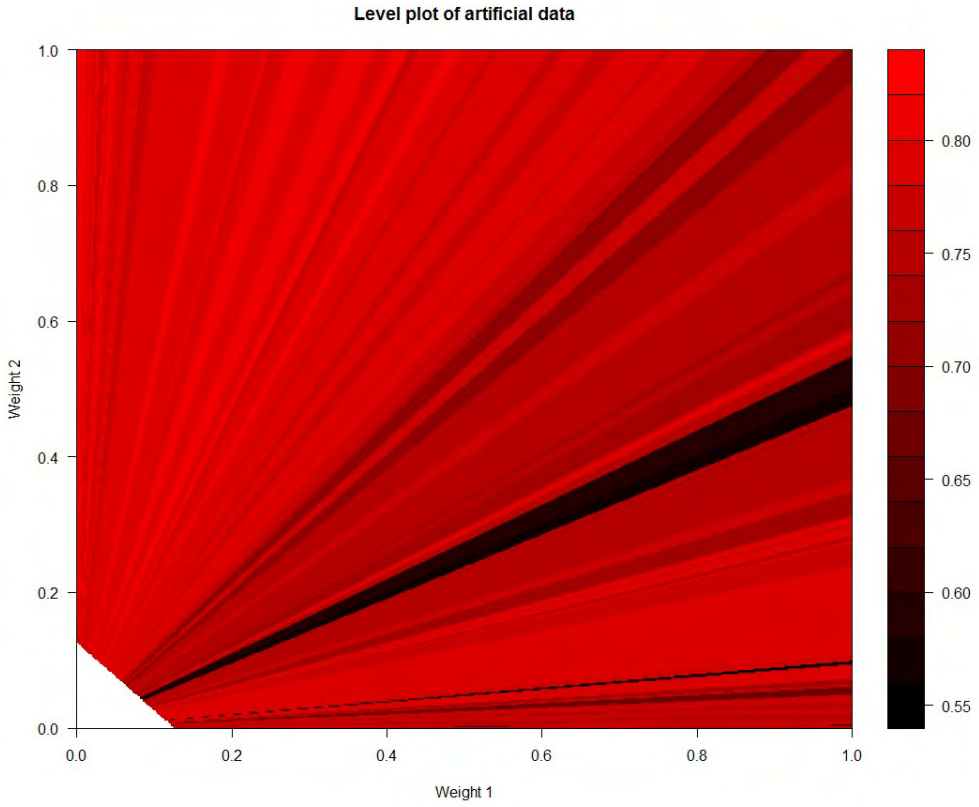


Figure 7: CPCC plot of an artificial dataset, the cluster-data set (see Appendix 12.1). The x-axis corresponds to the weight of the first variable, the y-axis corresponds to the weight for the second variable. The colour represents the CPCC. Note that we had to do some discretization in both the weights, therefore the lines are slightly jagged. Furthermore, the area in the bottom left corner corresponds to undefined behaviour of the distance function, which causes errors in the clustering method and prevents calculation of the CPCC.

There is a clear pattern in Figure 7. It looks like the CPCC does not change if the ratio of *weight 1* to *weight 2* is kept constant. This is most apparent in the blue line below the line $x = y$. This indicates that there might be some redundancy in the weights, since we can express this relationship between *weight 1*, and *weight 2* as:

$$W_1 = aW_2,$$

where a is a positive constant. This means that we can express the CPCC as a function of just one of the weights since the other weight is determined if one is fixed. Upon further inspection of Gower's metric, we find that one variable can be determined from the others. Given two sets of weights W and W^*, where the equation $aW = W^*$ holds for some constant $a > 0$. Then we have, using Equation 1 for Gower's metric:

$$S_{ij}{}^* = \frac{\sum_{k=1}^{N} w_k^* S_{ijk}}{\sum_{k=1}^{N} w_k^*},$$

$$S_{ij}{}^* = \frac{a\sum_{k=1}^{N} w_k S_{ijk}}{a\sum_{k=1}^{N} w_k},$$

Equation 5: Gower's metric equality

$$S_{ij}{}^* = \frac{\sum_{k=1}^{N} w_k S_{ijk}}{\sum_{k=1}^{N} w_k} = S_{ij}$$

Equation 5 indicates that the only important part in choosing the weights is the ratios between the weights, not the sum of the weights. All possible values for the CPCC / IoA should be achievable with any sum of weights $C > 0$ and $w_i \geq 0$ using:

$$\sum_{i=1}^{N} w_i = C.$$

A visualisation of a case where $N = 3$ implies that this equation indeed holds for multiple dimensions, as implied by Equation 5. Since this proof holds for the distance metric, the consequences of this result should apply to both the CPCC and the IoA. The most interesting implication is that this allows us to reduce the number of dimensions by one. We have to choose only $m - 1$ weights if we set C to a predefined value. We use $C = 1$ throughout the rest of this thesis. The reason for this is twofold: first, setting $C = 1$ allows us to easily interpret the weights. In this case, the weights represent the relative importance of variables more intuitively since they range from zero to one and sum to one, allowing us to interpret them as fractions. Second, this setting allows us to simplify some computations regarding the derivative of the CPCC later on in Sect. 4.3. In the next section, we will investigate which method will be used for our target function.

3.2 THE TARGET FUNCTION

The next step is to take a look at four possible target functions for optimising our clustering. The CPCC will be used since it is intended to define the quality of a hierarchical clustering. The IoA is considered as an alternative since it was designed with the intention to be a more robust version of the correlation coefficient r.

1. The CPCC, using the weighted Gower's distance matrix and the dendrogram distance matrix.
2. The CPCC, using the unweighted Gower's distance matrix and the dendrogram distance matrix.
3. The index of agreement (IoA) with the weighted Gower's distance matrix and the dendrogram distance matrix.
4. The index of agreement with the unweighted Gower's distance matrix and the dendrogram distance matrix.

The difference between using the weighted and unweighted Gower's distance matrix is that the method using the unweighted matrix correlates the dendrogram distance matrix with a distance matrix based on Gower's metric with all weights set to one. In contrast, the weighted matrix uses the same set of weights which are used by the clustering algorithm to calculate Gower' distance. We can use Equation 5 to improve the visualisation aspect of the evaluation of the most suited target function when we use only two variables to cluster on. This equation allows us to calculate the weight of the second variable as follows: $W_2 = 1 - W_1$. This allows us to create graphs where the value of the first weight is on the x-axis and the CPCC / IoA is on the y-axis. The second weight is then implied through Equation 5. We report the results of this analysis in the figures below, marked Figure 8, Figure 9, Figure 10, and Figure 11 from top to bottom:

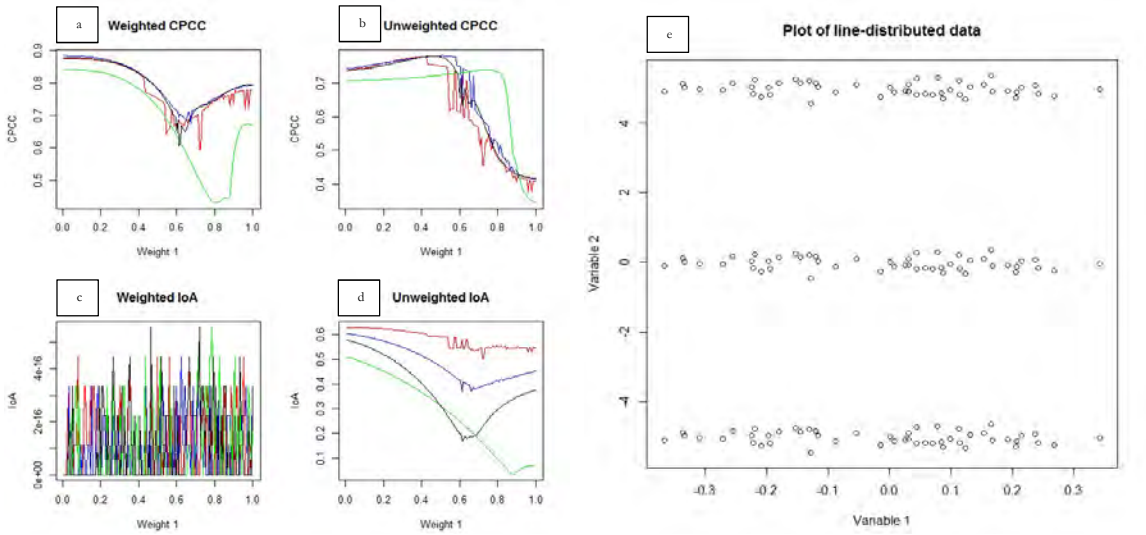


Figure 8a, b, c, d, and e: Influence of datasets and linkage functions on the weighted CPCC (a), unweighted CPCC (b), weighted IoA (c), and unweighted IoA (d) for a line-shaped dataset (e). The red line represents complete linkage, the blue line represents average linkage, the green line represents single linkage, and the black line represents centroid linkage. Note that the scales of the x-axis are the same, but those of the y-axis are not.

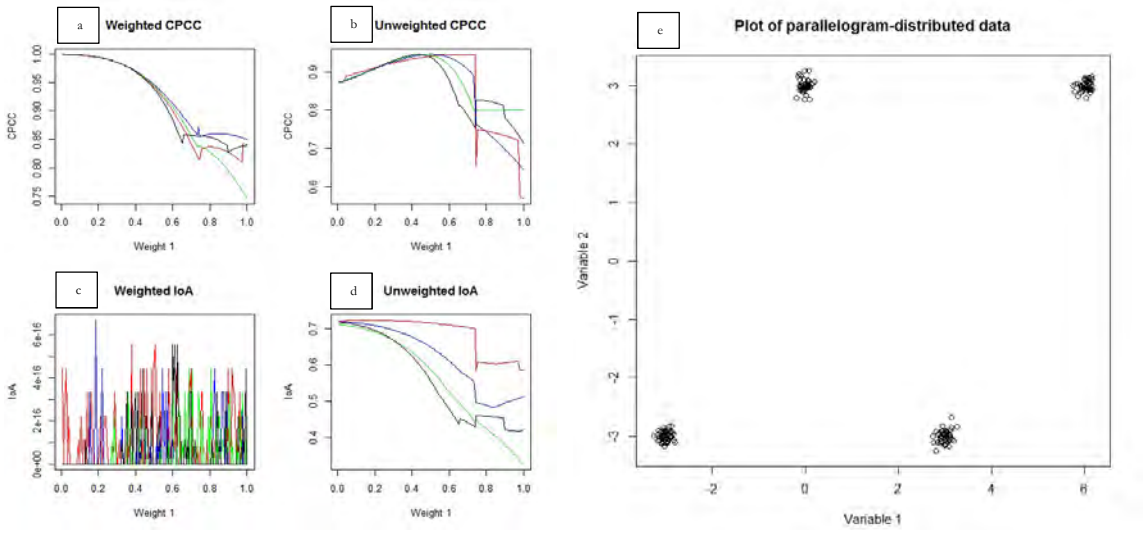


Figure 9a, b, c, d, and e: The same as Figure 8 for a parallelogram-shaped dataset.

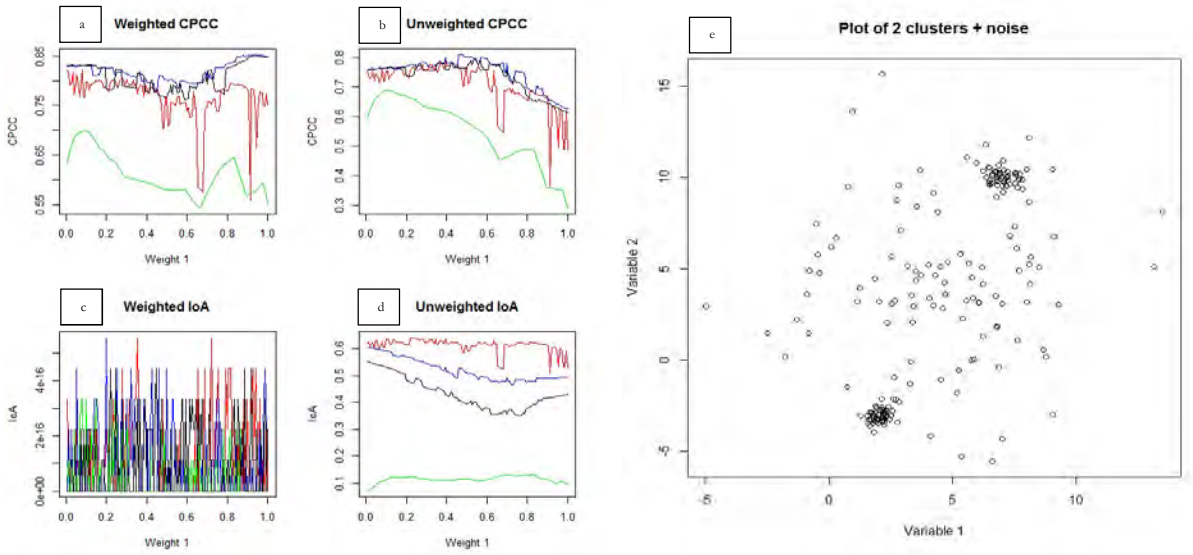


Figure 10a, b, c, d, and e: The same as Figure 8 for a dataset with two clusters and some noise.

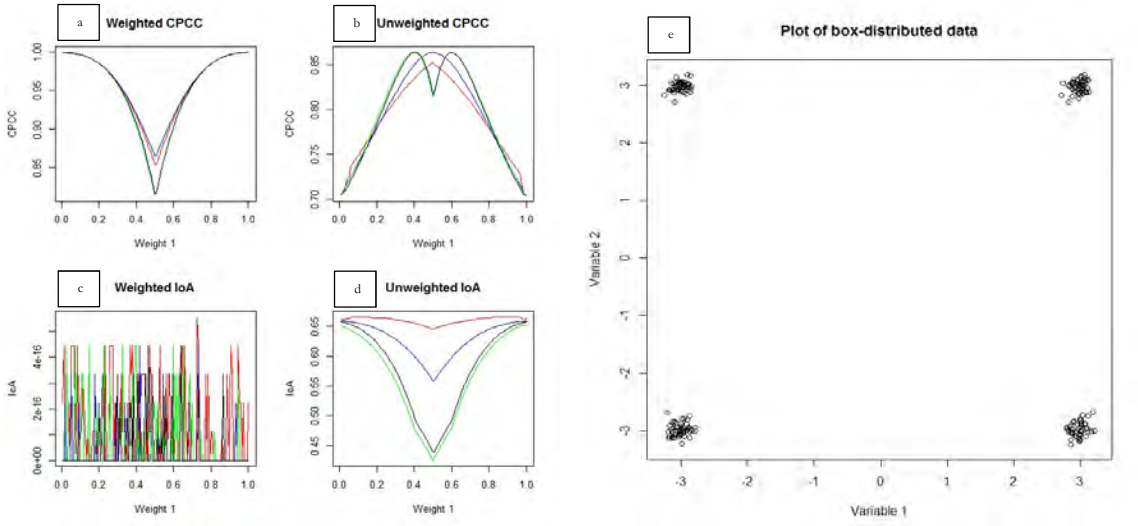


Figure 11a, b, c, d, and e: The same as Figure 8 for a box-generated dataset.

We can see that the weighted IoA is nearly always almost zero with erratic behaviour for all values of the weights, though it uses the same data as the rest. Compared to the other three metrics that do provide more normal results, this is especially unexpected. At this time we are not sure what causes this behaviour, especially since the unweighted IoA functions properly. Since the other target functions do seem to work, we will not further consider the weighted IoA as an option for our target function.

The unweighted CPCC seems to prefer a mix of both variables, which is to be expected since it correlates the cophenetic distances with an unweighted Gower's matrix. This matrix is calculated using a setup where both weights get the same value during the calculation of the distances. It is logical that the set of weights that generate the highest unweighted CPCC tend to be close to an even distribution of the weights, matching the way the hierarchical clustering (and thereby the cophenetic distances) have been calculated.

The unweighted IoA and weighted CPCC tend to be higher around the extremes of W_1, effectively removing one of the variables, even though this could potentially remove existing clusters from the dataset. For an example of this, look at the *box* dataset (Figure 11). If either one of the weights is set to zero, it would remove one of the two axes. If for instance the y-axis would be removed, then the clusters which are now positioned above each other (the two clusters at $x = -3$ and the two at $x = 3$) would be merged. This would cause us to make the wrong conclusions about the number of clusters.

It is interesting to note that when using the final algorithm neither of the optimised weights for the unweighted CPCC was set to be exactly equal to zero in the *box* dataset. One of the weights does come very close to zero. This causes the weighted CPCC method to not remove the clusters in question. The reason for this is that while the inter-cluster distance gets reduced when one of the weights is (almost) set to zero, so does the intra-cluster distance. We can prevent weights being set to zero by ensuring that the weights do not drop below a predefined boundary that is higher than zero in our final algorithm, though the *box* dataset shows us that this does not have to be done in all cases.

We choose to use the weighted or unweighted CPCC for now. A final selection between these two will be made in Sect. 4.1. Furthermore, we stop looking at the IoA. First, because finding a derivative of the IoA would be very difficult. The reason for using the derivative is that we use a quasi-Newton method to optimise the weights, which requires a derivative. More information regarding this subject will be discussed in Sect. 3.4. Second, since the CPCC seems to work as our target function, we do not require the IoA, which was not designed for doing the work that the CPCC was designed for: measuring the quality of a hierarchical clustering.

Analysis of the CPCC.

In Sect. 2.4 we explained the basic working of the CPCC, repeated below:

$$c = \frac{\sum_{i<j}(x(i,j)-\bar{x})(t(i,j)-\bar{t})}{\sqrt{(\sum_{i<j}(x(i,j)-\bar{x})^2)(\sum_{i<j}(t(i,j)-\bar{t})^2)}},$$

As we can see, the CPCC has two main components: $x(i,j)$ and $t(i,j)$. Of these $x(i,j)$ represents the distances calculated by Gower's metric from Sect. 2.3. The logical consequence is that we have a properly defined formula for $x(i,j)$ in the form of Equation 1.

However, $t(i,j)$, the cophenetic distance, is not defined in the same way as Gower's metric: it has no known explicit formula. $t(i,j)$ is based on the distances between objects in the dendrogram. This means that $t(i,j)$ is the result of an algorithm, the hierarchical clustering algorithm. To specify further, it depends on the linkage function and on $x(i,j)$ in a complicated way. This does not mean that it is, therefore, impossible to find a formula for $t(i,j)$ to use in the calculation of the CPCC, but in this case, we believe it would be difficult to do so. Knowledge of the exact values of $x(i,j)$ and its interaction with the linkage function is needed to determine $t(i,j)$, making the creation of an analytical formula for $t(i,j)$ difficult if not impossible.

The consequence of this is that finding an analytical derivative, which can be used in many optimisation algorithms, will most likely not be possible if we do not consider $t(i,j)$ as constant at each update step. This allows us to derive the derivative as if $t(i,j)$ is a constant for each combination of i and j, significantly increasing the chance of finding a working derivative. This process of updating $t(i,j)$ in steps will require a heuristic to find the optimal value of the CPCC. The heuristic would alternate changing $t(i,j)$ and W, the weight vector, or use a similar algorithm to optimise the CPCC.

In the case of the weighted CPCC $x(i,j)$ is dependent on W, so there is a possibility to find a derivative. However, as we have seen, this form of the CPCC might have a (too) strong bias for removing certain variables. In the case of the unweighted CPCC $x(i,j)$ is not dependent on W, since in $x(i,j)$ all weights are set to one when using the unweighted CPCC. Thus, it will not be possible to find an analytical derivate of the unweighted CPCC unless we have a formula for $t(i,j)$.

As seen in Figure 8 to Figure 11, both versions of the CPCC can be heavily influenced in terms of continuity by noise. It is important to look at ways to increase the level of continuity if possible since this gives us better evidence that using the derivative is justified. Since influencing the data in a general way is beyond the scope of this thesis, we first consider the effect of the linkage function on the smoothness.

3.3 THE LINKAGE FUNCTION

As discussed in Sect. 3.2, *t(i,j)* depends on *x(i,j)* and on the linkage method. We are interested to see what the effect of different linkage methods is on the continuity of the CPCC, which can be found in Figure 12. A figure for each of the datasets can be found in appendix 12.1. Note that these are figures for the unweighted CPCC. Since the continuous nature of the unweighted CPCC seems comparable to the continuous nature of the weighted CPCC, we assume that results for the unweighted CPCC in terms of the linkage function are also representative of the weighted CPCC.

- Expand this assumption

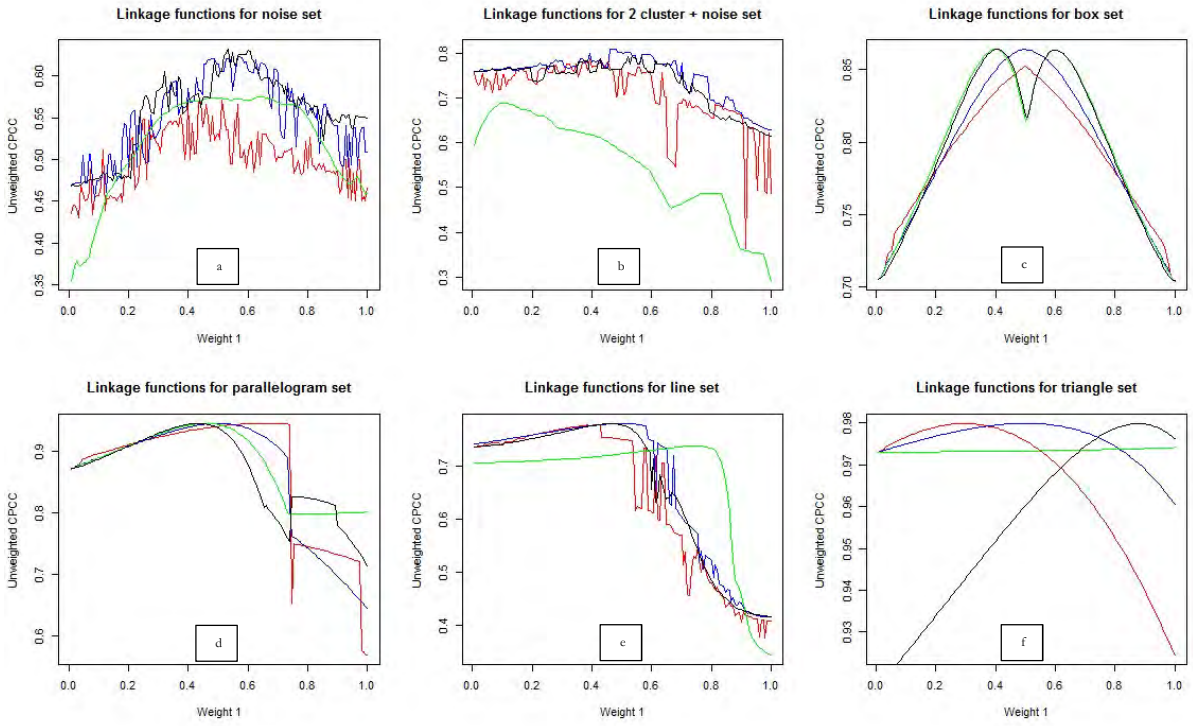


Figure 12a, b, c, d, e, f: Unweighted CPCC on multiple datasets and using different linkage functions. In all figures, the lines represent complete linkage (red), average linkage (blue), single linkage (green), and centroid linkage (black). a): noise set, b): cluster set, c): box set, d): parallelogram set, e): line set, f): triangle set.

The single linkage function (green) almost always produces a relatively smooth curve. However, it also generates lower values of the CPCC than the other linkage functions on most datasets. Furthermore, it does not seem to obtain the same optimal values for the weights when compared to the other linkage functions. Complete linkage (red) produces higher CPCC values than single linkage, but (in general) lower values than average linkage and centroid linkage. It also seems to be the most erratic of the four, with a large number of sudden increases and decreases in four of the six graphs. See for example the datasets *parallelogram*, *line*, *noise* and *2 clusters*. Average linkage (blue) produces high CPCC values and seems to behave better than complete linkage. Finally, centroid linkage (black) produces high CPCC values as well and behaves properly, which is comparable to the average linkage method.

It seems that the performance of the average linkage and centroid linkage functions is comparable and reasonably good. When we have to choose between average linkage and centroid linkage, we choose the average linkage for the remainder of this thesis. The main reason for this is that centroid linkage can, occasionally, produce dendrograms in which the distance at which two clusters are merged decreases when going up the tree. Since this is a property that we do not like, we decided to work with average linkage for now.

3.4 THE SEARCH HEURISTIC

In order to optimise our weights, we use a heuristic that allows us to optimise the weighted or unweighted CPCC. Our solution is to use a quasi-Newton method to optimise the weights. For this purpose, we use the L-BFGS-B algorithm, the Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with simple box constraints. This is an adaptation of a popular quasi-Newton method, the BFGS method, which uses limited memory. The simple box constraints allow us to impose fixed bounds on the weights, allowing us to guarantee that the rule that the weights should be larger than or equal to zero and no larger than one holds.

Like most quasi-Newton methods, the L-BFGS-B method works well on functions that are continuous and thus have properly defined derivatives. The findings discussed in Sect. 3.2 and 3.3, Figure 8 to Figure 12, suggest that this assumption may not always hold, with the CPCC occasionally displaying noisy behaviour. In general, these results indicate that an increase of noise results in a decrease of the continuity of the CPCC, of which Figure 12a, b, and c are an example. Each dataset can be found in appendix 12.1. Early investigation on datasets with more variables (e.g. 8 variables, 15 instances) reveals that it is possible to find a local optimum which is significantly different from the initial solution. However, the quasi-Newton algorithm may occasionally not converge because of problems with the line search algorithm, which is used to determine where to go in the next iteration. In these cases, the algorithm cannot find a weight vector minimising the CPCC in the direction indicated by the derivative. However, if we can find at least a local optimum from an initial solution, then we can find a global optimum as well, or at least an acceptable local optimum, if we use multiple initial solutions.

3.5 FINDING GLOBAL OPTIMA

The quasi-Newton method has a chance of converging to a local optimum. Our proposal for finding a global optimum to the CPCC is to repeat the above-mentioned algorithm with multiple initialisations. Using multiple initialisations gives us a larger chance of obtaining the global optimum since a set of different initialisations will cover a wider range of the search area. This improves the chance of starting near a global optimum, making the chance of converging to that optimum higher. This algorithm would work as follows:

1. Find X sets of initial weights. In the initial version of the algorithm, we choose X sets of weights uniformly between a set of bounds. For now, these bounds are $\left[\frac{1}{3N}, 1-\frac{N-1}{3N}\right]$, where N is the number of variables. These bounds have been chosen in such a way that the variables have enough flexibility to be above and below the default initial value of $\frac{1}{N}$, which corresponds to all weights having equal value.
2. Calculate the CPCC for each set of weights.
3. Sort the weights sets by their corresponding CPCC in descending order and start the L-BFGS-B algorithm at the first set of weights.
4. Stop when a minimum required value for the CPCC has been obtained or when all weights sets have been optimised. This threshold value for the CPCC needs to be set beforehand.

Instead of looking at all possible starting points X, we can look at the first $n \leq X$. This speeds up calculation times but tends to still find good sets of weights. It is important to choose n in such a way that we do not just select sets of weights that are already close to each other. The algorithm can be terminated when n points have been optimised or when a minimum value of the CPCC has been found. For instance, any CPCC higher than 0.9 would probably indicate a good clustering. Further improvements to this algorithm could be made, but for the purpose of testing the difference between local and global optimisation, this algorithm will suffice.

3.6 BOUNDARY ANALYSIS

As mentioned in Sect. 3.2, the optimisation function needs bounds on the weights such that each weight is larger than or equal to zero. We can select an alternative lower bound *LB* as long as this lower bound is between $\left[0, \frac{1}{N}\right]$, where *N* is the number of weights. If we set the *LB* higher than $\frac{1}{N}$, the sum of all weights will be larger than one. We also select $1 - (N-1)LB$ as the upper bound. This value is the maximum possible value of one weight if all the other weights are set to the lower bound and we keep a maximum value for the sum of the weights of one. For optimisation purposes it is interesting to take a look at the influence of *LB* on the CPCC and on the selection of the weights. In Figure 13, Figure 14, and Figure 15 the results of such an analysis are presented. The *cloud* dataset corresponds to the *noise* dataset of Sect. 3.4. The structure of Figure 13 and Figure 15 can be found in Appendix 12.1, the structure of Figure 14 can be found in the documentation for **R**.

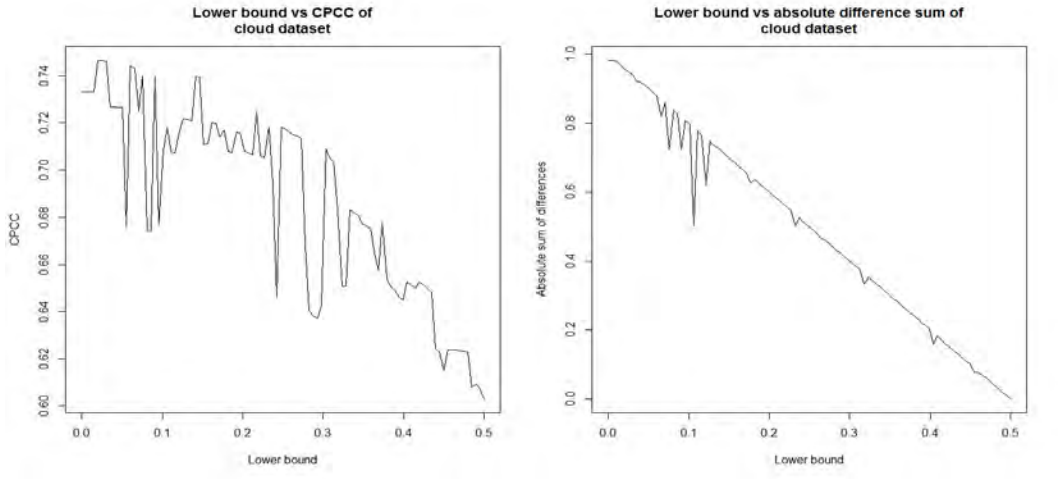


Figure 13a and b: Influence of the lower bound on the weighted CPCC (a) and the sum of absolute differences (b). The sum of absolute differences is the absolute difference between the two weights. The first set of graphs (a) is from the noise / cloud dataset, the second (b) is from and the third dataset (c) is from the box-dataset. The results for the box dataset graph are comparable for the triangle, parallelogram and line datasets.

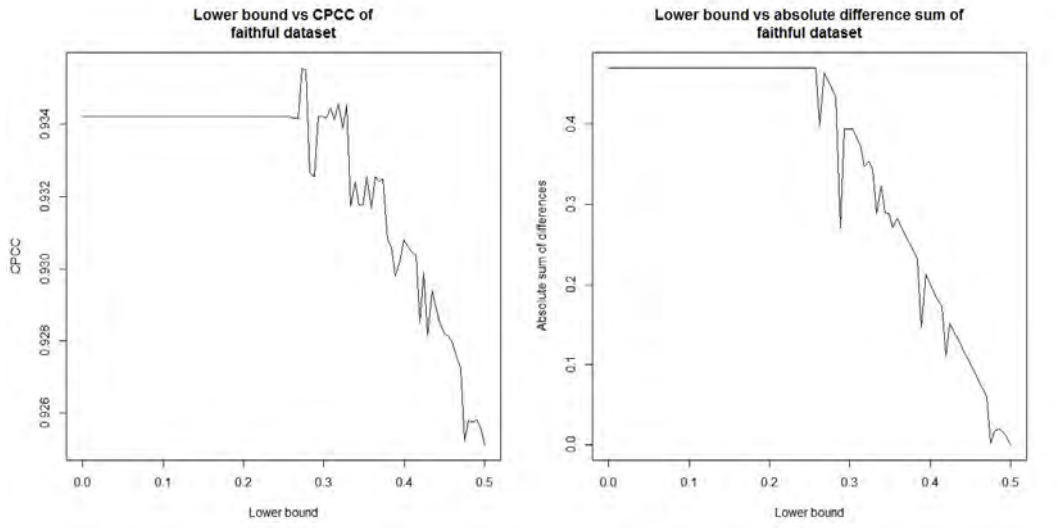


Figure 14a and b: The same as Figure 13 for the old faithful dataset.

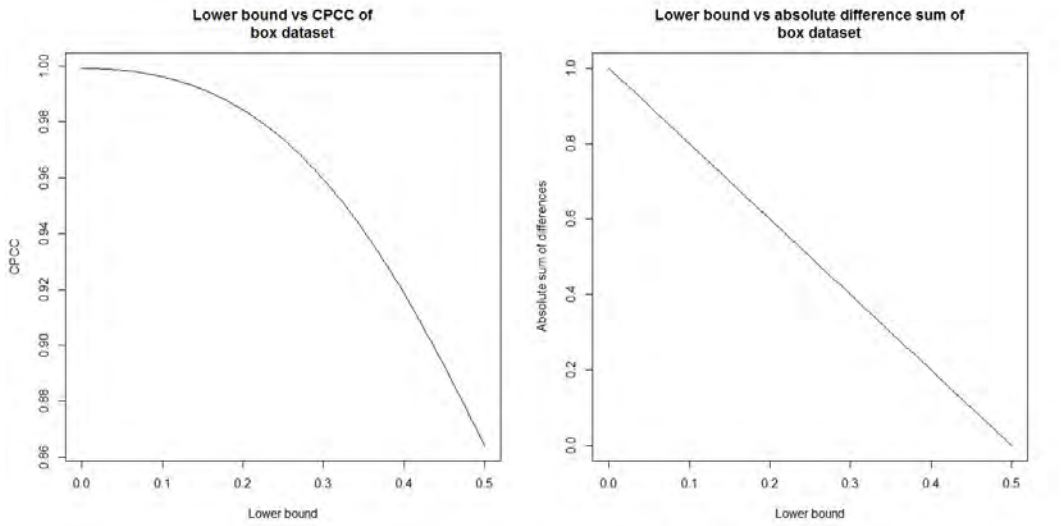


Figure 15a and b: The same as Figure 13 for the box-dataset. The results for the box dataset graph are comparable for the triangle, parallelogram and line datasets.

There are some characteristics that are present in all three sets of graphs:

- The CPCC tends to decrease as the lower bound increases. Since a higher *LB* allows the algorithm fewer possibilities for changing the weights, the logical result is that the algorithm has a lower probability of finding a high CPCC.
- The sum of absolute differences decreases when the lower bound increases. This follows the same logic as presented above.

For the *cloud* and *box* datasets, the lower bound has a much larger influence on the CPCC and sum of absolute differences for a longer time than it does on the *faithful* dataset. The changes in the graphs for the *faithful* and *cloud* dataset are probably the result of a different set of final weights due to the changing boundaries. These graphs show that the bounds are quite important for finding an adequate value of the CPCC. When one uses our algorithm, one should focus extra attention on choosing the weights properly.

3.7 SETUP SUMMARY

The summary of our algorithm can be found in Table 1:

Table 1: Setup for the algorithm

Parameters	**Configuration**
Clustering method	Hierarchical clustering.
Linkage function	Average linkage.
Target function	Weighted CPCC or unweighted CPCC, depending on our first test results.
Optimisation method	L-BFGS-B optimisation method.
Initialisation method	Start with all weights equal to each other.
Global optimum method	Select multiple starting weights
Evaluation method	WB-ratio and silhouette to determine the number of clusters.

We did not discuss the evaluation method in this section. These will be discussed in the next section where we evaluate our optimisation method. The general structure of the rest of the thesis is described in Table 2:

Table 2: The general structure of the thesis.

Step #	**Process**
1	Determine the CPCC version to use
2	Validate that the optimisation algorithms improves the clustering using artificial and real data
3	Derive the derivative of the CPCC
4	Prepare our final dataset for testing
5	Validate that the optimisation algorithms improves the clustering using our final dataset

4 MODEL SELECTION / VALIDATION

In this section, we finish our algorithm and compare it to two benchmark algorithms: the standard hierarchical and K-medoids algorithms. We select either the unweighted or weighted CPCC and compare the results to the standard hierarchical algorithm. Next, we compare the silhouette and within / between sum of squares ratio of our chosen method to the standard hierarchical and K-medoids methods. These metrics have been discussed in Sect. 2.6. Finally, we show the derivation of the derivative of the CPCC and perform speed and CPCC comparisons for different implementations of our algorithm.

4.1 COMPARING DIFFERENT HIERARCHICAL SETUPS ON THE CPCC

In Sect. 3.2 we discuss two possible versions of the CPCC which could be used as our target function. These versions use the weighted / unweighted Gower's distance matrix as input for $x(i,j)$. These methods are denoted as W-CPCC and U-CPCC respectively. We need to determine which of these versions is used in our optimisation algorithm. We report the values of both versions of the CPCC for a locally and globally optimising algorithm, generating in total four possible algorithms. The measurements have been generated using an approximation of the derivative, which has been constructed using the finite differences method. The benchmark of a standard hierarchical case uses Gower's distance metric using all weights set to one (see Equation 1). The results are shown in Table 3 and Table 4:

Table 3: Performance of the different algorithms on artificial data. W-CPCC corresponds to the weighted CPCC, U-CPCC to the unweighted CPCC.

Model	**Box-data**	**Cloud-data**	**Cluster & noise data**	**Parallelogram data**	**Line-data**	**Triangle**
Standard Hierarchical	0.86	0.60	0.80	0.95	0.78	0.98
Hierarchical, optimised locally, W-CPCC	0.99	0.72	0.80	1.00	0.88	1.00
Hierarchical, optimised globally, W-CPCC	1.00	0.75	0.85	1.00	0.88	1.00
Hierarchical, optimised locally, U-CPCC	0.86	0.62	0.81	0.95	0.78	0.98
Hierarchical, optimised globally, U-CPCC	0.86	0.62	0.81	0.95	0.78	0.98

Table 4: Performance of the different algorithms on real datasets (Appendix 12.2). The datasets can be found in R *using the same name used to identify them here. Using the notation from Sect. 3.5, we used a threshold value for the CPCC of 1, an X of 100 and an n of 10 for the global settings. W-CPCC corresponds to the weighted CPCC, U-CPCC to the unweighted CPCC. 1: the optimisation algorithm uses n = 5 instead of 10 due to time constraints and using a subset of the dataset.*

Model	**Esoph**	**Faithful**	**Infert**	**Mtcars**	**Quakes[1]**
Standard Hierarchical	0.63	0.93	0.80	0.80	0.76
Hierarchical, optimised locally, W-CPCC	0.90	0.93	1.00	1.00	0.97
Hierarchical, optimised globally, W-CPCC	0.90	0.94	1.00	1.00	1.00
Hierarchical, optimised locally, U-CPCC	0.65	0.93	0.80	0.80	0.82
Hierarchical, optimised globally, U-CPCC	0.65	0.93	0.80	0.82	0.82

We observe from Table 3 and Table 4 that the U-CPCC provides no significant increase (< 0.02) in CPCC compared to the base hierarchical case where all weights are equal to one, both locally and globally. Furthermore, the unweighted CPCC performs worse with regards to the goal of optimising the CPCC than the weighted CPCC algorithm. Given that the unweighted CPCC algorithm does not outperform the standard hierarchical case, we will not use the unweighted CPCC anymore.

For the weighted CPCC, there is a significant increase in the CPCC compared to the base case of all weights set to one in almost every case. The difference between locally and globally optimising in terms of the CPCC is small (< 0.01) in almost all cases except the *cluster & noise*, *cloud*, and *quakes* dataset. In these cases, the difference is between 0.02 and 0.05. This difference is not significant enough to warrant the extra computation time required to perform the globally optimising algorithm. The quasi-Newton method might be better at optimising the weighted CPCC than we first anticipated, independent of the initialisation. We think this is the case because initializing the function with different sets of weights (as done in the global optimisation algorithm) results in similar or the same CPCC values with similar weights. This indicates that the L-BFGS method might be converging to at least a similar optimum.

We conclude that the optimisation function using the weighted CPCC works well. However, since the difference between the local and global algorithm tends to be insignificant, it might be better to use the locally optimising algorithm and not the globally optimising algorithm in most cases. The local method costs significantly less computation time, which is more important than a small increase in the CPCC. For the unweighted CPCC, the optimisation function does not generate an increase in the CPCC for most datasets, compared to the standard hierarchical algorithm.

4.2 COMPARING HIERARCHICAL CLUSTERINGS ON OTHER METRICS

We now know that our algorithm improves the CPCC compared to the standard hierarchical algorithm. We now want to determine if our algorithm outperforms the benchmark algorithms in other quality measures. To be able to compare the hierarchical algorithms with the K-medoids algorithm, we focus on the process of selecting the number of clusters and evaluating the quality of these algorithms. To do so we execute each of the clustering algorithms and report the values we would select if we were to perform the clustering only using that method. These tend to be close to the optimal value (maximum for the silhouette, minimum for the WB-ratio, both explained in Sect. 2.6). For instance, if the metrics we use are comparable for $K = 2$ and $K = 3$, then we would select $K = 3$ since this allows us to examine the third cluster as well. The results for the artificial datasets can be found below in Table 5:

Table 5: Results for our benchmark algorithms and our own algorithm on artificial datasets. Sil: average silhouette. WB: within sum of squares divided by the between sum of squares.

Dataset	**K-medoids**	**Standard Hierarchical**	**Hierarchical optimised**
Line-data	Sil: 0.64, K=3 WB: 0.20, K=3	Sil: 0.64, K=3 WB: 0.20, K=3	Sil: 0.85, K=3 WB: 0.12, K=3
Box-data	Sil: 0.96, K=4 WB: 0.03, K=4	Sil: 0.96, K=4 WB: 0.03, K=4	Sil: 0.89, K=4 WB: 0.03, K=4
Parallelogram-data	Sil: 0.96, K=4 WB: 0.02, K=4	Sil: 0.96, K=4 WB: 0.02, K=4	Sil: 0.92, K=2 WB: 0.03, K=4
Triangle-data	Sil: 0.98, K=3 WB: 0.02, K=3	Sil: 0.98, K=3 WB: 0.02, K=3	Sil: 1.00, K=3 WB: 0.00, K=3
Cluster & noise-data	Sil: 0.59, K=4 WB: 0.25, K=4	Sil: 0.58, K=6 WB: 0.25, K=6	Sil: 0.60, K=6 WB: 0.24, K=6
Cloud-data	Sil: 0.35, K=4 WB: 0.43, K=6	Sil: 0.39, K=2 WB: 0.48, K=7	Sil: 0.47, K=2 WB: 0.35, K=5

It is interesting to note that the optimised hierarchical clustering algorithm does not always generate a higher silhouette than the other methods on artificial data, but the within sum of squares ratio is comparable or better. For the *line*, *triangle*, *cluster & noise*, and *cloud* datasets the silhouette and the WB-ratio are both better than the benchmark algorithms. For the other two datasets, *box* and *parallelogram*, the silhouette is slightly worse than that of the benchmark algorithms and the WB-ratio is comparable. It seems that the estimate for the number of clusters can be derived better from the WB-ratio than from the silhouette when using our algorithm. We can draw this conclusion since we know the true number of

clusters in each of the datasets and the WB-ratio is better at predicting this number than the silhouette. For the *cluster & noise* and *cloud* dataset, all algorithms perform poorly.

We also look at a set of real datasets, namely the same as those introduced in Sect. 4.1. We report the results in Table 6. These have been structured in the same way as in Table 5:

Table 6: Results for our benchmark algorithms and our own method on real data from R. Sil: average silhouette. WB: within sum of squares divided by the between sum of squares.

Dataset	**K-medoids**	**Standard Hierarchical**	**Hierarchical optimised**
Esoph	Sil: 0.240, K=2 WB: 0.5392, K=7	Sil: 0.317, K=4 WB: 0.5513, K=7	Sil: 0.645, K=2 WB: 0.329, K=2
Faithful	Sil: 0.772, K=2 WB: 0.220, K=2	Sil: 0.772, K=2 WB: 0.220, K=2	Sil: 0.7907, K=2 WB: 0.204, K=2
Infert	Sil: 0.435, K=2 WB: 0.494, K=4	Sil: 0.451, K=3 WB: 0.526, K=3	Sil: 0.8999, K=3 WB: 0.099, K=3
Mtcars	Sil: 0.553, K=5 WB: 0.302, K=4	Sil: 0.549, K=5 WB: 0.312, K=4	Sil: 0.9174, K=3 WB: 0.0753, K=3
Quakes	Sil: 0.426, K=3 WB: 0.442, K=5	Sil: 0.307, K=3 WB: 0.455, K=7	Sil: 0.738, K=2 WB: 0.263, K=2

The optimised algorithm performs better on all of these datasets compared to our benchmark algorithms. The silhouette is consistently higher and the WB-ratio is lower. The standard hierarchical algorithm and K-medoids algorithm perform somewhat similar, with the standard hierarchical algorithm performing better on three of the five datasets and the K-medoids algorithm performing better on the other two. It is interesting to see that the expected number of clusters can differ significantly between the three methods on the same dataset. This number can even differ for one algorithm and one dataset based on the two metrics. The most obvious example of this is are the *esoph* and *quakes* datasets, where we see that the conclusions for the number of clusters are different for the three methods.

Something to note is that on the *mtcars* dataset the clustering is the same for the standard hierarchical clustering setup and the optimised hierarchical clustering. The same occurs on the *infert* dataset for $K = 3$, *4* and *7*, but not at $K = 5$. The weights seem to (sometimes) reinforce the normal structure of the hierarchical clustering: if the algorithm would separate the dataset mainly on one variable, it is assigned a high weight. Though this is interesting, in our opinion we would need more datasets to be able to generalise this.

4.2.1 Silhouette difference test

We want to know if our algorithm performs significantly better than the two benchmark algorithms. To do this, we use the Quade test to test if there is a significant difference in silhouettes, taking as rows the different datasets and as columns the different clustering algorithms (Friedman, 1937), as explained in Sect. 2.7. Quade's test is deemed more powerful when we have less than five columns (Quade, 1979). This test produced a P-value of 0.013. With $a = 0.05$ we draw the same conclusions for both tests: there is evidence that there is a significant difference between at least one pair of algorithms.

Doing the posthoc Quade test helps us to find the statistically significant different algorithm(s). Since we are only interested in the difference between the optimised hierarchical algorithm and the two benchmark algorithms, we only conduct tests on these two pairs. The test has been adjusted using the Holm-Bonferroni method of Sect 2.7. The results can be found in Table 7.

Table 7: Results from the posthoc tests.

Methods	**K-medoids**	**Standard Hierarchical**
Hierarchical Optimised, Quade	0.005	0.06

The test generates significant p-values for the K-medoids comparison at $a = 0.05$, thus we draw the conclusion that there is sufficient evidence that there is a significant difference between K-medoids and optimised hierarchical in terms of the silhouette. There is no significant difference between the standard hierarchical algorithm and our optimised algorithm at $a = 0.05$. This causes us to draw the conclusion that there does not seem to be enough evidence to say that optimised hierarchical performs better than the standard hierarchical algorithm in terms of the silhouette.

4.2.2 Within sum of squares ratio difference test

The test for significant differences in WB-ratio uses the same methods as those employed for the silhouette test. We use Quade's test to see if there is a significant difference in within/between sum of squares ratio, taking as rows the different datasets and as columns the different clustering algorithms (Friedman, 1937). This test generated a P-value of 0.0003. With $\alpha = 0.05$ there is enough evidence to say that the WB-ratio differs significantly between the clustering algorithms.

Performing the posthoc Quade test helps us to find the statistically significant different algorithm(s). Since we are only interested in the difference between the optimised hierarchical algorithm and the two benchmark algorithms, we only conduct tests on these two pairs. The test has been adjusted using the Holm-Bonferroni method of Sect 2.7. The results of the posthoc test can be found in Table 8.

Table 8: Results from the posthoc tests.

Methods	**K-medoids**	**Standard Hierarchical**
Hierarchical Optimised, Quade	0.003	0.0002

All tests suggest significant difference at $a = 0.05$ on the comparison of the standard hierarchical algorithm and the optimised hierarchical algorithm. Thus, we conclude that there is a significant difference in WB-ratio between both the K-medoids algorithm and the standard hierarchical algorithm and our optimised hierarchical algorithm.

4.3 THE DERIVATIVE OF THE CPCC

In Sect. 4.1 and 4.2 we have seen that our algorithm performs better than the two benchmark algorithms. In our algorithm, we use a quasi-Newton method, which uses the derivative. Up to now, we have used finite differences to approximate the derivative, but this is very time consuming since we need to calculate multiple hierarchical clusterings to obtain the derivative. An analytical formula would be faster to compute than finite differences. This is the case because an analytical formula would require a maximum of one clustering algorithm to obtain a derivative and the finite differences approach needs multiple. As a reminder, the equation for the CPCC is the following:

Equation 2: CPCC

$$c = \frac{\sum_{i<j}(x(i,j)-\bar{x})(t(i,j)-\bar{t})}{\sqrt{(\sum_{i<j}(x(i,j)-\bar{x})^2)(\sum_{i<j}(t(i,j)-\bar{t})^2)}},$$

where:

- $x(i,j)$: the Euclidean / Gower's distance between instances i and j.
- $t(i,j)$: the distance between instances i and j in the dendrogram, defined as the height in the dendrogram where the two instances are joined for the first time.
- $\bar{x}$: the average Euclidean / Gower's distance between instance.
- $\bar{t}$: the average distance between instances in the dendrogram.

As mentioned in Sect 3.2, we might encounter problems when trying to find a derivative. In this formula, $x(i,j)$ depends on the weights and $t(i,j)$ depends on $x(i,j)$ and the linkage function. A problem arises when we try to find the derivative: $x(i,j)$ can be expressed in terms of the data and the weights, but $t(i,j)$ cannot,

since it is the result of a hierarchical clustering algorithm. The only way we can find the derivative of the CPCC now is by assuming that *t(i,j)* can be considered as independent of the weights during the calculation of the derivative. For this assumption to be reasonable, the change in the CPCC should not be significant when the change in the weights is small. As we have seen in Sect. 3.1 and 3.2, this is not always the case. However, the finite differences approach uses a similar assumption and still works. For this reason, we attempted to find the derivative and we shall compare the results of using the derivate to the results of the finite differences approach. The intention is to determine if these methods differ much. If this is not the case, then we can conclude that our derivation works, or generates results comparable to the finite differences approach.

To improve readability, we define the following variables:

Equation 6: the definitions of a and b.

$$a(i,j) = x(i,j) - \bar{x}$$

$$b(i,j) = t(i,j) - \bar{t}$$

Using Equation 2 we obtainEquation 1:

$$c = \frac{\sum_{i<j} a(i,j)b(i,j)}{\sqrt{\sum_{i<j} a(i,j)^2 \sum_{i<j} b(i,j)^2}} = \frac{1}{\sqrt{\sum_{i<j} b(i,j)^2}} \frac{\sum_{i<j} a(i,j)b(i,j)}{\sqrt{\sum_{i<j} a(i,j)^2}}.$$

The derivative of *c* with respect to w_k, the k^{th} weight of Gower's metric reads:

Equation 7: first version of the derivative

$$\frac{\partial c}{\partial w_k} = \frac{1}{\sqrt{\sum_{i<j} b(i,j)^2}} \frac{\sum_{i<j} \frac{\partial a}{\partial w_k}(i,j)b(i,j) \sqrt{\sum_{i<j} a(i,j)^2} - \sum_{i<j} a(i,j)b(i,j) \frac{\sum_{i<j} \frac{\partial a}{\partial w_k}(i,j)a(i,j)}{\sqrt{\sum_{i<j} a(i,j)^2}}}{\sum_{i<j} a(i,j)^2}.$$

We first look at the derivative of *a(i,j)*, $\frac{\partial a}{\partial w_k}$, so that we can substitute it into Equation 7:

Equation 8: the derivative of a(i,j).

$$\frac{\partial a}{\partial w_k}(i,j) = \frac{\partial x}{\partial w_k}(i,j) - \frac{\partial \bar{x}}{\partial w_k},$$

where *x(i,j)* is the Gower's distance between instance *i* and *j*:

$$x(i,j) = S_{ij} = \frac{\sum_{k=1}^{K} w_k S_{ijk}}{\sum_{k=1}^{K} w_k},$$

where *K* is the number of variables and S_{ijk} is the distance between instances *i* and *j* on variable *k*. For ease of notation we define:

Equation 9: the sum of the weights.

$$W = \sum_{k=1}^{N} w_k,$$

Equation 10: the number of objects in the sum.

$$N_s = \frac{N(N-1)}{2},$$

where N is the number of instances. Recall from Sect. 3.1 that we choose $\sum_{k=1}^{N} w_k = 1$, following the results from Equation 5. Combining this with Equation 8, Equation 9, and Equation 10 gives the following derivative for *a*:

$$\frac{\partial a}{\partial w_k}(i,j) = -\frac{1}{W^2}a(i,j) + \frac{1}{W}\left(S_{ijk} - \frac{1}{N_s}\sum_{e<d} S_{edk}\right).$$

$$\frac{\partial a}{\partial w_k}(i,j) = -a(i,j) + S_{ijk} - \frac{1}{N_s}\sum_{e<d} S_{edk},$$

Equation 11: the final derivative of a(i,j).

$$\frac{\partial a}{\partial w_k}(i,j) = S_{ijk} - \bar{S}_k - a(i,j),$$

where:

$$\bar{S}_k = \frac{1}{N_s}\sum_{e<d} S_{edk}$$

Before we substitute the derivative of *a* into the derivative of *c*, we note that:

$$\sum_{i<j} a(i,j) = 0,$$

$$\sum_{i<j} b(i,j) = 0.$$

From the definitions of *a* and *b* in Equation 6, this should be clear. Combining this information with the derivative of *a* shows us that we do not need to substitute $\bar{S}_k$ into the derivative of *c*. After all, we could simply split up the summations in which the derivative occurs, find that $\bar{S}_k$ appears only with an *a(i,j)* or *b(i,j)* in the summations, allowing us to remove it from the calculation. Combining Equation 7 and Equation 11 gives us the following equation:

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j}\left(S_{ijk} - a(i,j)\right)b(i,j)\sqrt{\sum_{i<j} a(i,j)^2} - \sum_{i<j} a(i,j)b(i,j)\dfrac{\sum_{i<j}\left(S_{ijk} - a(i,j)\right)a(i,j)}{\sqrt{\sum_{i<j} a(i,j)^2}}}{\sum_{i<j} a(i,j)^2\sqrt{\sum_{i<j} b(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j} b(i,j)\left(S_{ijk} - a(i,j)\right) - \sum_{i<j} a(i,j)b(i,j)\dfrac{\sum_{i<j}\left(S_{ijk} - a(i,j)\right)a(i,j)}{\sum_{i<j} a(i,j)^2}}{\sqrt{\sum_{i<j} a(i,j)^2\sum_{i<j} b(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j} b(i,j)\left(S_{ijk} - a(i,j) - a(i,j)\dfrac{\sum_{e<d}\left(S_{edk} - a(e,d)\right)a(e,d)}{\sum_{e<d} a(e,d)^2}\right)}{\sqrt{\sum_{i<j} a(i,j)^2\sum_{i<j} b(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j} b(i,j)\left(S_{ijk} - a(i,j)\left(1 + \frac{\sum_{e<d} S_{edk} a(e,d)}{\sum_{e<d} a(e,d)^2} - \frac{\sum_{e<d} a(e,d)^2}{\sum_{e<d} a(e,d)^2}\right)\right)}{\sqrt{\sum_{i<j} a(i,j)^2 \sum_{i<j} b(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j} b(i,j)\left(S_{ijk} - a(i,j)\left(1 + \frac{\sum_{e<d} S_{edk} a(e,d)}{\sum_{e<d} a(e,d)^2} - 1\right)\right)}{\sqrt{\sum_{i<j} a(i,j)^2 \sum_{i<j} b(i,j)^2}}.$$

Equation 12: the derivative of the CPCC.

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j} b(i,j)\left(S_{ijk} - a(i,j)\frac{\sum_{e<d} S_{edk} a(e,d)}{\sum_{e<d} a(e,d)^2}\right)}{\sqrt{\sum_{i<j} a(i,j)^2 \sum_{i<j} b(i,j)^2}}.$$

Note that many of the variables present in the derivative are constant with respect to k for one set of weights, such as the denominators. This can help improve calculation times by not calculating these for each weight. Furthermore, S_{ijk} is constant for each dataset, so these values can be calculated before the optimisation starts. In our final implementation, we use a different formulation of this derivative, which is optimised to reduce the total number of matrix calculations required:

Equation 13: the final derivative of the CPCC

$$\frac{\partial c}{\partial w_k} = \sum_{i<j} S_{ijk}\left(\frac{b(i,j) - a(i,j)\frac{\sum_{e<d} a(e,d) b(e,d)}{\sum_{e<d} a(e,d)^2}}{\sqrt{\sum_{e<d} a(e,d)^2 \sum_{e<d} b(e,d)^2}}\right).$$

4.4 PERFORMANCE ANALYSIS

In this section, we test the difference between the finite differences method and the derivative method. We focus on the computation times and on the CPCC. , we perform a speed test on the **R** built-in dataset *quakes*. The results are presented below in Table 9.

Table 9: Speed test of different derivative methods. The user time is the CPU time charged for the execution of user instructions of the calling process. The 'system time' is the CPU time charged for execution by the system on behalf of the calling process. This definition comes from the documentation of system.time() in R (R-documentation, n.d.).

Algorithm	**User (s)**	**System (s)**	**Total (s)**
Finite differences	746.95	111.75	864.78
With derivative	253.17	44.81	306.47
With improved derivative implementation	50.05	9.22	63.85

It is clear from Table 9 that the derivative methods are considerably faster than the finite differences method, up to a factor of 13.5 for this dataset. Apart from the speed test it is important to test for consistency between the derivative and the finite differences methods. Below in Table 10 are the CPCC's for the different datasets for the different methods and the maximum percentage difference. This percentage is the maximum difference, expressed as a percentage, between the CPCC of the finite differences method and either of the derivatives.

Table 10: CPCC for the finite differences method and the percentage difference for the derivative methods. The dataset description can be found in Appendix 12.1 and 12.2.

Dataset	**Finite differences**	**With derivative**	**With improved derivative implementation**	**Max percentage difference**
Line-data	0.8781	0.8781	0.8781	0%
Box-data	0.9895	0.9895	0.9895	0%
Parallelogram-data	0.9960	0.9960	0.9960	0%
Triangle-data	0.9983	0.9983	0.9983	0%
Cluster & noise-data	0.8030	0.8103	0.8103	0%
Cloud-data	0.7186	0.7198	0.7198	0%
Esoph	0.8997	0.8997	0.8997	0%
Faithful	0.9342	0.9342	0.9342	0%
Infert	0.9969	0.9964	0.9964	0%
Mtcars	0.9923	0.9954	0.9953	0.3%
Quakes	0.9716	0.9462	0.9462	2.6%

Table 10 indicates that the derivative methods are consistent when compared to the finite differences method. Furthermore, there is no difference between the different implementations of the derivatives, except a 0.0001 difference at *mtcars*. Based on this evidence, we conclude that the difference between the derivative methods and the finite differences method is insignificant and that our derivative implementations works.

The improved derivative implements two possible settings: minimal storage requirements or faster calculation. This setting influences whether S_{ijk} from Gower's metric (see Equation 1) is computed online or before starting the optimisation function respectively. The faster version runs approximately twice as fast (and is the improved implementation method in Table 9 and *Table 10*. The disadvantage of this is that it requires all combinations of S_{ijk} to be stored during the entire process. For large datasets, it might not be possible to use the faster version. The largest set we used this on so far is a dataset of 1505 rows and 35 columns, which presented no problems on a laptop with 4 GB of RAM running Windows.

Relation between computation time and number of instances / observations

It is interesting to see what the influence of the input size is on the computation time. For this purpose, we use the *quakes* dataset since it allows us to select subsets of up to 1000 instances without replacement. We select the first *K* rows and perform the algorithm 10 times on that dataset. *K* starts at 20 instances and

increases with a step size of 20 to 1000 instances, the complete dataset. We report the average computation time for each number of instances below in Figure 16:

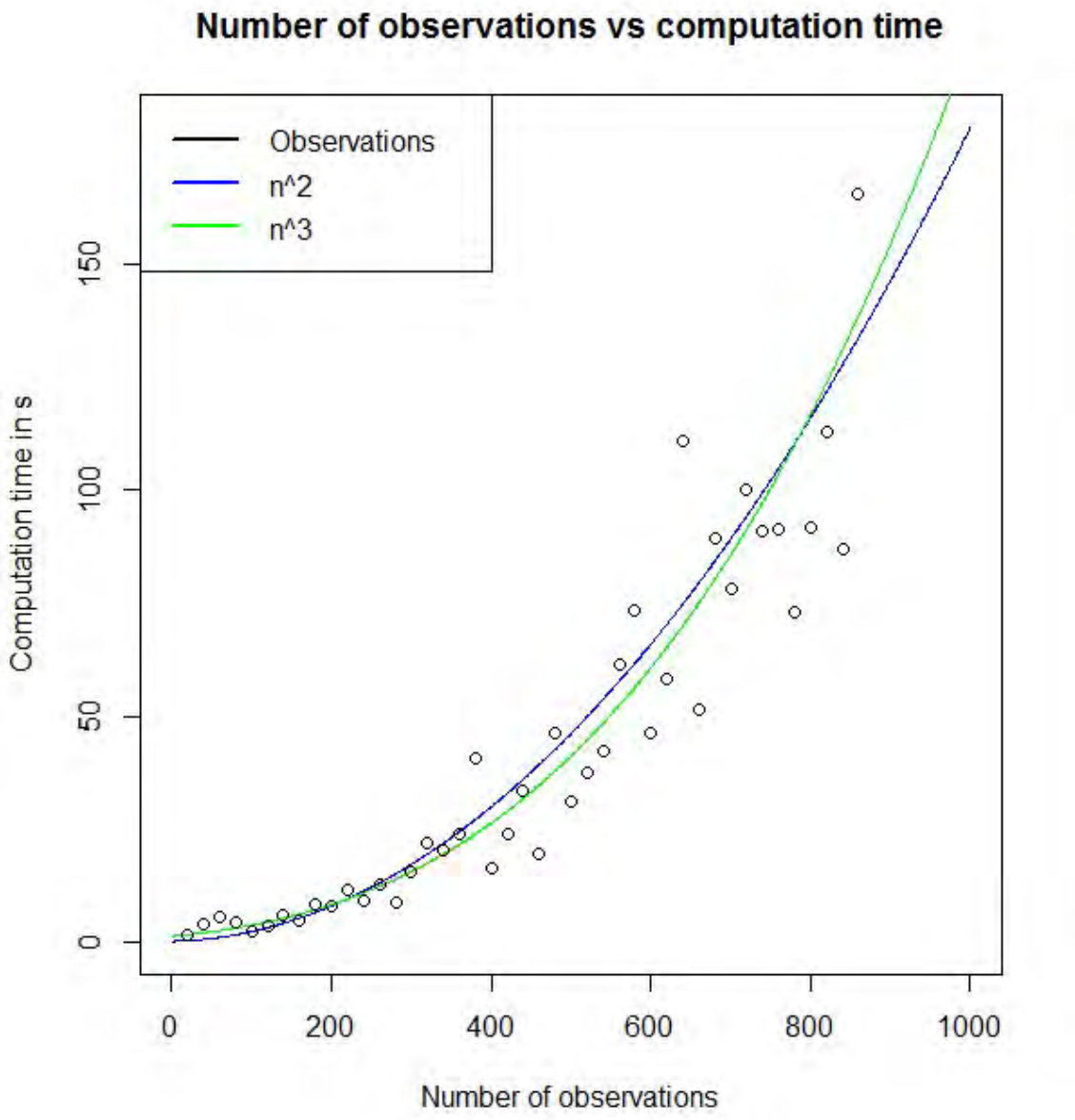


Figure 16: Number of instances / observations versus the average required computation time. One outlier at n = 920 with an average computation time of more than 400 has been removed.

From Figure 16 we can see an increase in computation times when the number of instances increases. To test the complexity we plotted two regression lines: n^2 and n^3, where n is the number of observations. The n^3 regression line seems to fit this data better than n^2 since it seems to capture the trend from 400 observations onward better than the n^2 regression line. This tells us that there is a good chance the algorithm runs in $O(n^3)$. A Shapiro-Wilk test on the residuals of a regression model of the n^3 line generates a P-value of 0.0002, indicating that with $a = 0.05$ that there is strong evidence that the residuals are normally distributed. Since we use hierarchical clustering, which runs in $O(n^3)$, the $O(n^3)$ complexity is in accordance with our expectations.

Relation between computation time and number of variables

The number of instances is not the only variable that can influence the computation times. Another possible influence is the number of variables we use for our clustering. As such, this relation should also be investigated. To test this we used 20 instances and a varying number of variables, ranging from two to 200. For each variable and each instance, we drew random samples from a standard normal distribution to generate a dataset. This result was then clustered. We repeated this experiment 190 times for each number of variables. The results can be found below in Figure 17.

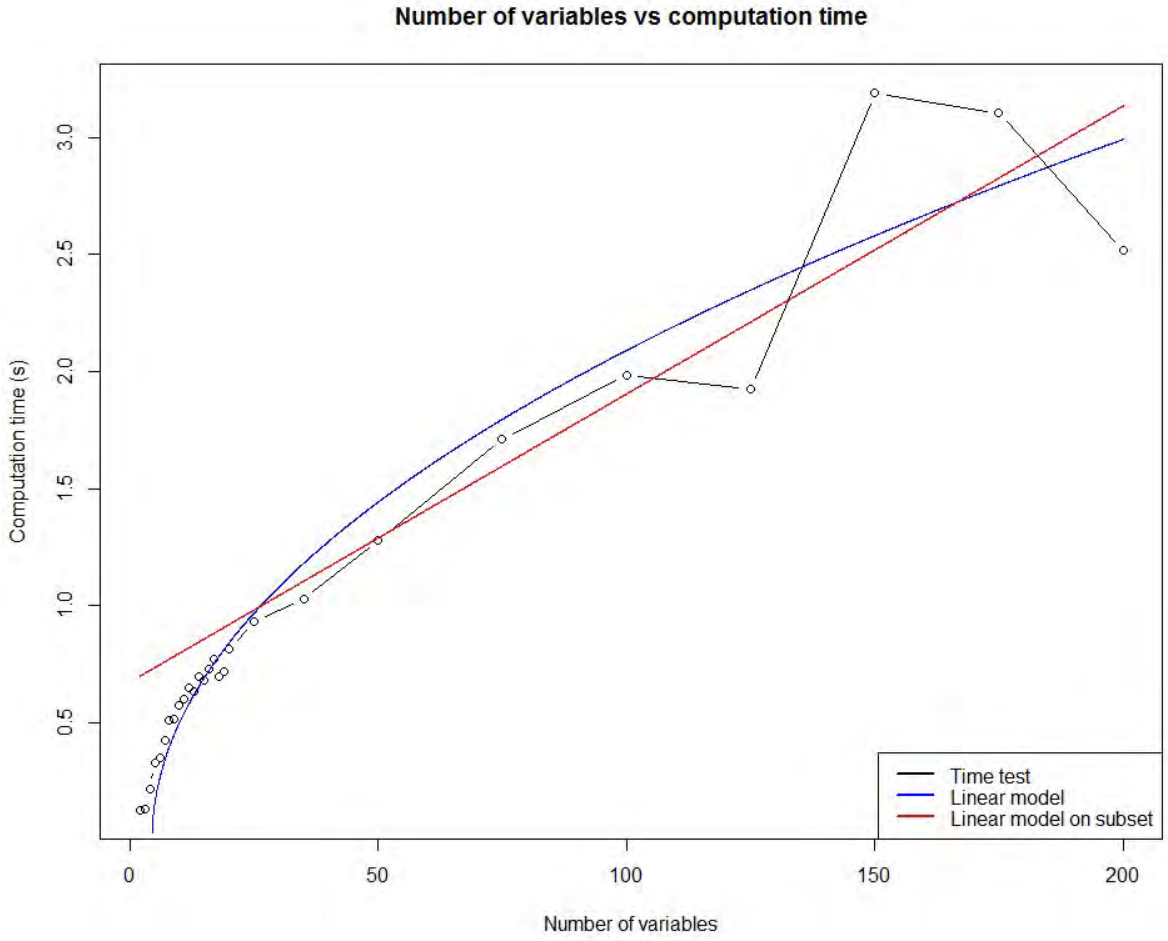


Figure 17: Number of variables versus the average required computation time. We included a blue regression line $\left(\sqrt{time} = Number\ of\ variables * \propto + \beta\right)$ *and a red linear regression line, performed on the subset of 'number of variables > 20'*

Figure 17 indicates that there is a relationship between the number of variables and the computation time, though it is less clear which one. Even for a sample size of 190 per experiment, the computation time varies significantly from 100 variables onward. There seems to be a sublinear relation between the *number of variables* and the *computation time*, but we are sceptical to conclude that this is indeed the case. We would not expect such a relation when working with complex optimisation algorithms, which tend to have time complexities above linear. The blue regression line represents a linear model of the relation between the *number of variables* and the square root of the *computation time*, which seems to fit the data well. However, from 20 variables onwards the relation between the two variables appears to be linear. The larger number of observations at low numbers of *number of variables* may have skewed the results. To test this, we include the red linear regression line, which has been constructed using just the experiments with more than 20 variables. In both cases, the coefficient for the *number of variables* was considered significant with P-values at 0.00015 and $4.8*10^{-13}$ for the red and blue model respectively. This indicates that both models are possible and that further research is required to distinguish the best fitting model.

5 MOBIQUITY DATASET EXPLORATION

5.1 INTRODUCTION

In the previous chapters of this thesis, we first discussed the methods we use for our algorithm and motivated the choices we made. In addition to that, we also showed that our algorithm outperforms the standard hierarchical and K-medoids clustering algorithms. In this section, we use a dataset of new, real dataset to validate our algorithm. This dataset describes interactions of users with a mobile application focused on increasing the awareness and prevalence of recycling. Our intention is to find clusters of users, which we use to gain insights into the types of users we have. The process of variable creation and data organisation was already performed (see Acknowledgements). In this section, we discuss the data pre-processing and exploration phase and in Sect. 6 we present the results of the clustering. We present a general overview of the dataset in Table 11:

Table 11: General overview of the dataset. It is important to note that in this dataset each row corresponds to one interaction of or with the user. This includes the user doing certain actions, but also the system sending messages to the user.

Number of rows	238661
Number of columns (/variables)	191
Number of unique users	2225
Number of factor variables	24
Number of logical variables	2
Number of integer variables	36
Number of numeric variables	129
Total percentage NA's (Not Available)	17.03%

Note in Table 11 the high percentage of NA's in the dataset. As we will show in the next section, the NA's tend to be concentrated in a small subset of the variables. Since we have a large number of variables (191), we select a subset of the variables to ensure that we do not supply the algorithm with unnecessary information, slowing down the computations significantly. Furthermore, since this dataset describes events, we have to convert this event-focused dataset to a user-focused dataset to ensure that we can perform our clustering on users.

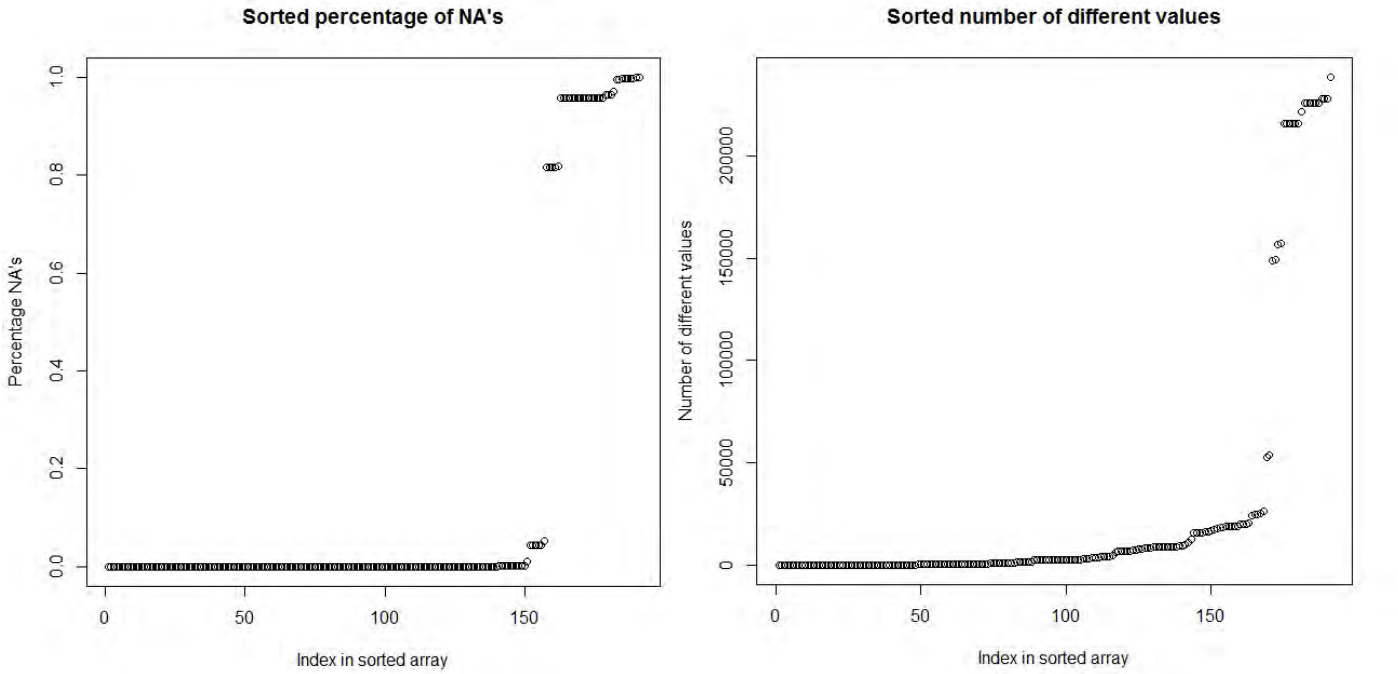
5.2 FEATURE SELECTION

We first look at variables that we want to remove before we start to look into specific features to keep. The variables that are removed in this way fulfil at least one of the following criteria:

- Categorical variables with too many levels are removed. The only exception to this is user-id, which we keep until we perform the clustering. Variables with too many levels are those with more levels than 30% of the number of instances.
- Variables with more than 90% NA's are removed immediately. Variables with more than 40% NA's are inspected more closely before we make a decision to remove them.
- Variables that have only one possible value, excluding NA, are removed.
- Variables which are a duplicate of another variable. Variables which hold similar information as others (such as mean time online and total time online) also fall into this category. We remove all but one of these variables for each set of duplicates.
- Variables that are not user-specific. If they do not describe the behaviour of the user, then we cannot use them to distinguish users with.

5.2.1 Preliminary analysis: NA's and number of different values.

We first consider the number of NA's and the number of different values for each variable to make a decision on which variables to remove. In Figure 18 the sorted percentage of NA's (a) and the sorted



number of different values (b) are shown.

Figure 18 a): sorted percentage of NA's for each variable. b): sorted number of different values for each variable.

Figure 18a shows that approximately 75% of variables have a NA percentage lower than 0.05. Only a small selection of variables have almost 100% NA values. As stated in Sect. 5.2, any variable with more than 90% NA values is removed, which results in 28 variables being removed.

Figure 18b shows that approximately 80% of all variables have less than ±30.000 values. These are not considered for removal yet since they do not have too many different values. Of the group with more than 30.000 values, none is a categorical variable. However, in this group, there exists one variable which is the sequence 1 to 238661 and which is removed. Furthermore, we find 17 variables with only 1 value (NA's excluded). These variables are removed as well.

All these variables amount to 36 unique variables that we have removed. This means that there are 10 overlapping variables, which probably originates from an overlap between the variables with too few different values and variables with too many NA's.

5.2.2 Removing duplicate and similar variables.

There is a large number of variables containing similar data. For instance, for most per-day analyses we have the minimum, maximum, median, mean, sum and standard deviation. Of these variables we only keep the mean and the standard deviation, since in our opinion, they represent these variables adequately. We continue by looking at what each variable describes and remove those that describe non-user-related information, such as total standard deviation for the *number of active days* measurement. Variables of which we do not know what they do and what they describe are also removed since we would not learn anything from including these variables.

Based on the criteria from Sect. 5.2 we removed 118 distinct variables. We now have 36 variables left. A number of variables are removed once we start clustering: *maxId* (user ID), *lastEvent*, *lastEventInput*,

lastEventOutput and *direction*. These are used during the conversion to a user-focused dataset and will be removed afterwards.

5.3 CONVERSION TO A USER-FOCUSED DATASET

The current dataset describes events, not users. We have to convert the dataset to a user-focused dataset to do clustering analysis on the users, since we are interested in which users are similar, not which events are similar. We want to cluster on the last information that we have from the user. However, a number of variables like *eventType* are effectively distributions. The application allows us to interact with it in five different ways, which are recorded as *eventType*. To be able to use this information, we have to convert this variable to a distribution. This means that we look at all the events the user generated and create the distribution of events for each user. This describes the behaviour of users as the fraction of all interactions that a user spent on each event.

History values describes the number of times the user generated input, but in a non-numerical way. We transform this to the number of times the user got input. The values for this variable consists of character sequences. They start and end with brackets ([]) with either nothing in between or sequences of ones separated by commas. Each 1 represents a previous event. An example of a valid value would be "[1,1,1]" or "[]". This means that if we want to count the total number of events, we can use the following equation:

Equation 14: history values

$$N = \begin{cases} 1 \; for \; [] \\ \dfrac{length + 1}{2} \; else \end{cases},$$

where:

- N is the total number of events.
- length is the length of the character sequence.

When we converted the dataset to a user-focused dataset, we noticed that this dataset contains 1505 users, even though we had 2225 users in the original dataset. This means that 720 users never interacted with the app. This should be investigated.

We create the above-mentioned set of six variables, five for the fraction of events associated with each type and one for the number of events, and add them to this dataset. We remove the five variables from Sect. 5.2.2. The general overview of the final dataset is presented in Table 12:

Table 12: Final overview of the clustering dataset.

Number of rows	1505
Number of columns (/variables)	35
Number of unique users	1505
Number of factor variables	0
Number of logical variables	0
Number of integer variables	2
Number of numeric variables	33
Percentage NA's (total)	0.23%

6 RESULTS

6.1 OVERVIEW

We apply our optimised hierarchical algorithm on the final dataset described in Sect. 5.3. We do this to show that the algorithm also works for larger datasets. The final results of the clustering can be found in Table 13. We use the same metrics as in Sect. 4.2, with the addition of the more: Dunn's index and the Pearson-gamma correlation.

Table 13: Clustering results. The reported statistics for the silhouette, WB-ratio, Pearson-gamma correlation and Dunn's index have been calculated using all 13 clusters.

Variable	**Value**
Total runtime	20.8 minutes
User time	17.0 minutes
System time	3.2 minutes
CPCC, standard hierarchical value	0.84
CPCC, optimised hierarchical value	0.97
Suspected number of clusters	4 main clusters, 9 small clusters
Corresponding silhouette	0.56
Corresponding WB-ratio	0.15
Corresponding Pearson-gamma correlation	0.92
Corresponding Dunn's index	1.38

Note that the algorithm caused a 15% increase in the CPCC compared to the base case using standard hierarchical clustering. This seems to be further validation that our algorithm is capable of finding optimal weight vectors for Gower's metric. The algorithm indicates four main clusters and nine small clusters. The small clusters contain a maximum of four users per cluster and together hold only 19 users in total. Further argumentation on our selection of the number of clusters will be presented in Sect. 6.4.

When we remove the nine small clusters from the calculation of the final statistics, they improve significantly. These statistics can be found in Table 14:

Table 14: Clustering results when excluding the 9 small clusters compared to using all clusters.

Variable	**All clusters**	**Four main clusters**
CPCC, optimised hierarchical value	0.9695	0.9694
Silhouette	0.560	0.674
WB-ratio	0.148	0.139
Pearson-gamma correlation	0.922	0.933
Dunn's index	1.382	2.035

Note that all statistics except the CPCC improve. The influence of the nine small clusters is most apparent in the silhouette and Dunn's index. The silhouette increases by 0.114. Dunn's index improves by 0.653. The Pearson-gamma correlation was already near its maximum value of one, which explains why it does not increase significantly. The WB-ratio remains nearly constant.

6.2 COMPARING STATISTICS

Since we have only numerical data in this dataset resulting from the dimensionality reduction in Sect. 5.2, we compare our algorithm to the K-means clustering algorithm as well. To be able to get a better comparison between the different algorithms, we perform K-means on a normalised version of the dataset, with all variables in the range [0, 1]. Each variable *j* is scaled using the following formula:

$$x(i,j)^* = \frac{x(i,j) - \min_{k=1:N} x(k,j)}{\max_{k=1:N} x(k,j) - \min_{k=1:N} x(k,j)},$$

where:

- *x(i, j)* : the value of variable *j* for instance *i*.
- *N* : the number of instances in the dataset.

We compare the same statistics that we mentioned in Sect. 6.1: the silhouette, WB-ratio, Dunn's (second) index and the Pearson-gamma correlation. The clustering algorithms that we compare are our optimised hierarchical algorithm, standard hierarchical, K-means, and K-medoids. The results can be found in Figure 19 below:

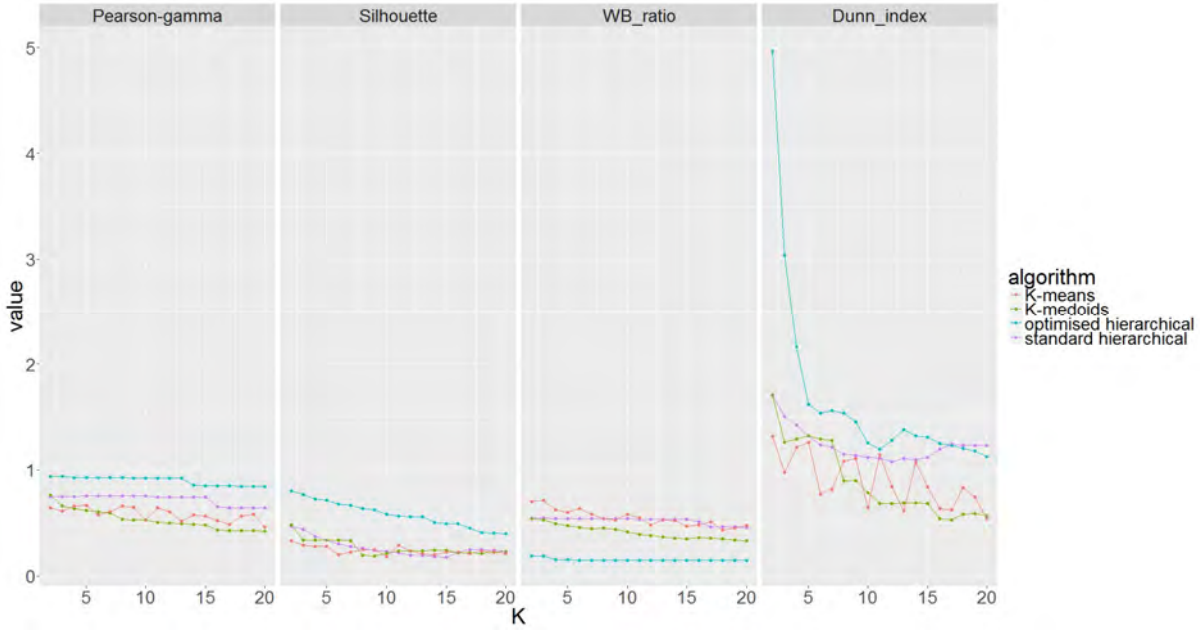


Figure 19: Quality metrics for all clustering algorithms, on 2 to 20 clusters. Of these statistics, Dunn's index, Pearson's correlation, and the silhouette should be as large as possible. The within / between sum of squares ratio should be as low as possible.

Figure 19 indicates that our algorithm performs consistently better on all four metrics and on all number of clusters compared to all other algorithms. The only exception is Dunn's index when compared to the standard hierarchical clustering algorithm when we select more than 16 clusters. In short, we conclude that our algorithm works better than the three benchmark algorithms in general. Of the other algorithms, standard hierarchical clustering performs better on Dunn's index and the Pearson-gamma correlation, K-medoids performs best on the WB-ratio.

6.3 SIGNIFICANCE OF RESULTS

In order to further extend the validation of our algorithm, we perform the same tests as before in Sect. 4.2.1 and 4.2.2 to test the significance of our results. This data has been collected for all four cluster algorithms and is based on 2 to 20 clusters. The results can be found in Table 15:

Table 15: P-values for the Quade test (Sect. 2.7) on the four selected statistics.

Algorithm	**Silhouette**	**WB-ratio**	**Dunn's index**	**Pearson-gamma**
Quade	2.306e-08	1.492e-14	6.511e-10	5.002e-16

These results indicate that there is a significant difference according to Quade's test between at least one pair of algorithms for each of the quality metrics. To determine if there is a significant difference between

our optimised hierarchical algorithm and the benchmark algorithms, we perform the posthoc Quade test on each set of metrics, similarly to our methodology in Sect. 4.2.1 and 4.2.2. As before, we use the Holm p-value correction method for this test (Sect. 2.7). The results are presented in Table 16:

Table 16: P-values for the silhouette, WB-ratio, Dunn's index and Pearson-gamma for the posthoc Quade tests.

Quality metric	**Base hierarchical**	**K-medoids**	**K-means**
Silhouette	< 0.00001	< 0.00001	< 0.00001
WB-ratio	< 0.00001	0.0004	< 0.00001
Dunn's index	0.03	< 0.00001	< 0.00001
Pearson-gamma	0.0002	< 0.00001	< 0.00001

All tests reject H_0 at $\alpha = 0.05$. Based on the results shown in Table 16 we can statistically confirm that our optimised hierarchical clustering algorithm performs better than all the benchmark algorithms tested in this thesis.

6.4 ANALYSIS OF CLUSTERING

Before we can analyse the clustering, we need to determine the number of clusters *K*. Results from the Pearson-gamma correlation from Figure 19 indicate that a significant difference occurs between *K = 13* and *K = 14*. Dunn's index also decreases after *K = 13,* but from Figure 19 it is clear that Dunn's index and the silhouette suggest as few clusters as possible. The WB-ratio changes very little after *K = 4*, indicating that *K = 4* is also an option to investigate.

We, therefore, conclude that *K = 4, K = 13*, and *K = 14* are possible candidates for the true number of clusters. The dendrogram (Figure 20) and the number of users in each cluster reveal that at *K = 4* we have one outlier group containing one instance and three main clusters. At *K = 13* we have nine small outlier clusters with a maximum number of users of four and at *K = 14* we split one of the larger clusters into two new clusters, of which the smallest one contains 67 users. Because of this, we set *K = 13,* remove the small clusters from the initial analysis, and study the large clusters and outlier clusters separately. This gives us four main clusters and nine small 'clusters', where the latter contain 19 users in total.

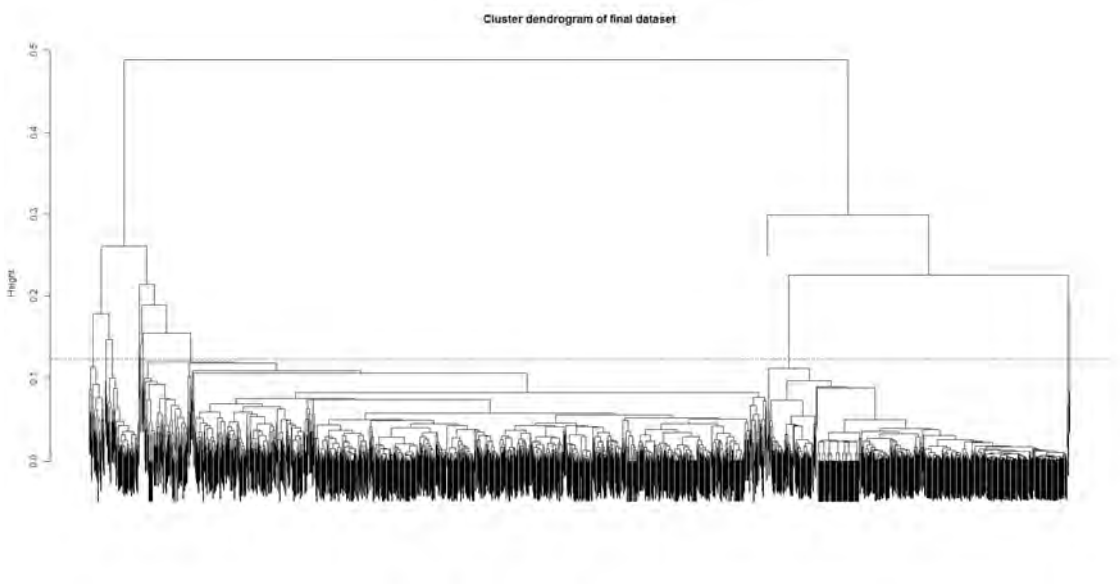


Figure 20: Dendrogram of the final dataset. The division between active / inactive users from Table 17 is clear in the large group on the left and slightly smaller group on the right. The red line marks the height where we split the dendrogram to create 13 clusters.

We continue by investigating what separates these 19 users from the rest of the data. Instead, we look at the minimum distance between a user and the other users. In this selection of 'other users', the first 19

users are excluded to understand how these 19 users relate to the others. The results are presented in Figure 21. The first 19 points represent the 19 outlier users, the others represent the other users.

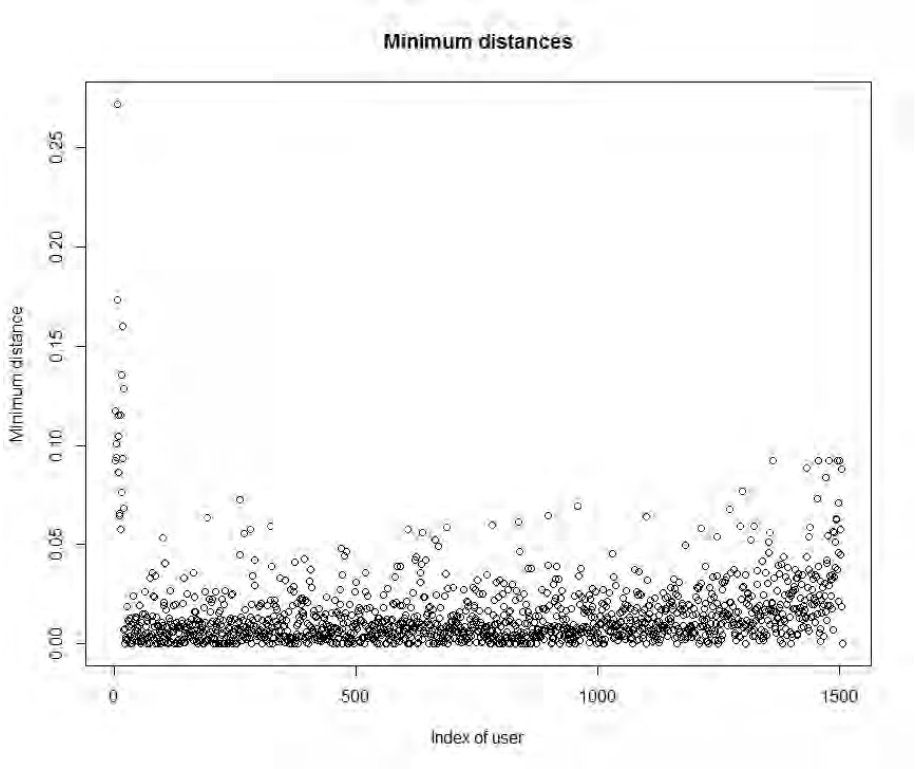


Figure 21: Minimum distance to another user, per user. The first 19 points represent the 19 outlier users, the other points represent the rest of the users.

It seems that the first 19 users have a relatively high minimum distance compared to the clustered group. However, some of the other users in this group also have a high minimum distance. This minimum distance can, occasionally, be even larger than the lowest minimum distance of the outlier group.

We now inspect some of the characteristics of the four main clusters in Table 17:

*Table 17: Characteristics of the final clustering. Errors in the fractions occur due to rounding errors. *: excluding one possible outlier with 154 uses, compared to the second most active user with 55 uses and the third with 26.*

Characteristic	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
Number of users	956	459	47	24
Mean days engaged	0	44.9	0.8	0
Mean Observed time	0	71.6	1.62	0
Mean number of uses	2.64	9.79	8.43	8.56*
Mean education fraction	0.93	0.80	0.95	0.97
Mean invitation fraction	0.03	0.03	0.00	0.00
Mean redemption fraction	0.00	0.01	0.01	0.00
Mean order fraction	0.02	0.04	0.03	0.02
Mean pickup fraction	0.02	0.13	0.02	0.01
Mean of average days since this event	145.6	136.7	132.8	34.7
Summary	Large group that used the app a few times then stopped	Group of users with varying levels of activity. More active than the first group	Users who were active first two days, then they stopped.	Very active first day users, recent as well.

The most obvious distinction is between cluster 1 and 2. Cluster 1 contains users who used the app a few times on the first day and then disengaged from the app. We can conclude this because the *mean observed time* is equal to zero and the *number of uses* is relatively low. Cluster 2, on the other hand, contains fewer users than cluster 1 but contains more active users. These users have used the app on average for 72 days and have used it almost ten times. Users in cluster 1 also seem to favour the *education* part of the app more than those in cluster 2, while users in cluster 2 use the *pickup* function more.

Clusters 3 and 4 look rather similar: they contain users that quit soon after starting to use the app but were very active on this day with 8.5 uses on average. Apart from this fact, they are similar to the users in cluster 1. The main difference between cluster 3 and 4 seems to originate from the *mean of average days since this event*, indicating that cluster 4 might contain more recent users than cluster 3. Table 17 indicates that there is no significant difference in the usage of the *invitation, redemption,* and *order* functions between the four clusters. Furthermore, the *mean of average days since this event* is comparable for clusters 1, 2, and 3. Only cluster 4 differs significantly from the others on this statistic.

7 ALTERNATIVES TO HIERARCHICAL CLUSTERING

7.1 BACKGROUND

The algorithm that we have created in this thesis is modular. To be able to optimise Gower's weights for a certain clustering algorithm, we need to know its target function and preferably that function's derivative. We can exploit this to try to implement different clustering algorithms using our optimisation method. One of the most important reasons to do this is that hierarchical clustering is quite slow with a time complexity of $O(n^3)$.

Since K-medoids is a method we understand well, we create a trial K-medoids implementation and test if we see similar improvements as we observed with the hierarchical algorithm. Another possibility would be to use DB-scan.

7.2 DEFINING THE TARGET FUNCTION AND DERIVATIVE

For our K-medoids implementation we use the sum of distances between the centroid of each cluster and its associated instances as the target function, which we want to minimise:

Equation 15: the target function of K-medoids.

$$f(w, X) = \sum_{k=1}^{K} \sum_{x \in X_k} d_w(x, C_k),$$

where:

- w = the weight vector.
- X = the dataset.
- K = the number of clusters
- X_k = all instances belonging to cluster k.
- $d_w(x,y)$ = the distance between instances x and y, using weight vector w.
- C_k = the centroid of the k-th cluster.

Apart from the target function, we would also like to be able to use its derivative. This allows us to speed up the computation times of the BFGS method. The derivative of this function is as follows:

$$\frac{\partial f(w, X)}{\partial w_i} = \frac{\partial f}{\partial w_i} = \sum_{k=1}^{K} \sum_{x \in X_k} \frac{\partial d_w(x, C_k)}{\partial w_i}$$

Equation 16: the derivative of Equation 15.

$$\frac{\partial f}{\partial w_i} = \sum_{k=1}^{K} \sum_{x \in X_k} \frac{\partial}{\partial w_i} \left(\frac{1}{W} \sum_{z=1}^{Z} w_z d_z(x, C_k) \right),$$

where:

- $W = \sum_{k=1}^{N} w_k = 1$
- Z = the number of variables
- $d_i(x, C_k)$ = The distance between x and C_k on variable i.

From this follows:

$$\frac{\partial f}{\partial w_i} = \sum_{k=1}^{K} \sum_{x \in X_k} d_i(x, C_k) - \sum_{z=1}^{Z} w_z d_z(x, C_k)$$

Equation 17: the final derivative of K-medoids

$$\frac{\partial f}{\partial w_i} = \sum_{k=1}^{K} \sum_{x \in X_k} d_i(x, C_k) - d_w(x, C_k),$$

7.3 RESULTS

We look at the relative difference in the target function for the standard and optimised K-medoids algorithms to test for improvements. The reason we look at the relative difference and not the absolute difference is that, unlike the CPCC, our target function for the K-medoids algorithm is not fixed on a finite scale, making the comparison between different datasets difficult. We report the mean value of the target function for the optimised case divided by the mean value of the target function for the standard case below in Table 18. The datasets are the same as those used for testing the optimised hierarchical algorithm and a description can be found in Appendix 12.1 and 12.2.

Table 18: Fraction of the mean value for the target function. This equals the mean of the optimised algorithm's value for the target function divided by the mean of the standard algorithm's value. These have been averaged for K = 2 to 7.

Dataset	**Fraction of the mean**
Line-data	0.58
Box-data	0.50
Parallelogram-data	0.49
Triangle-data	0.76
Cluster & noise-data	0.83
Cloud-data	0.78
Esoph	0.74
Faithful	0.86
Infert	0.96
Mtcars	1.00
Quakes	0.87

From Table 18 we see that in all cases except *mtcars* the optimised K-medoids provides an improvement compared to the standard K-medoids algorithm. The exception at *mtcars* is caused by a failed convergence on the first optimisation step. Studying the other datasets shows that such a convergence problem occurs in approximately 50% of the values for *K*. This shows that a good initial estimate of *K* is crucial. In view of this, we recommend using different initial starting vectors for the weights might be helpful to attain proper convergence.

For the artificial datasets, we see that the optimised K-medoids algorithm reduces the target function by 17 to 50%. For the real datasets, excluding mtcars, this target function gets reduced by 4% to 26%. This indicates that the algorithm works as intended. We need to test the silhouette and WB-ratio of both our benchmark algorithm and our optimised K-medoids algorithm to see if the same effect applies to the actual clustering. The results of this test can be found in Table 19. The *mtcars* dataset has been excluded since it would generate the same results as the standard case unless we were to select a different initial weight vector:

Table 19: Silhouette and WB-ratio for the standard and optimised hierarchical algorithms. Mtcars has not been included because of the results from the target function in Table 18.

Dataset	**Standard K-medoids**	**Optimised K-medoids**
Line-data	Sil: 0.64, K=3 WB: 0.20, K=3	Sil: 0.85, K=3 WB: 0.12, K=3
Box-data	Sil: 0.96, K=4 WB: 0.03, K=4	Sil: 0.96, K=4 WB: 0.03, K=4
Parallelogram-data	Sil: 0.96, K=4 WB: 0.02, K=4	Sil: 0.97, K=4 WB: 0.02, K=4
Triangle-data	Sil: 0.98, K=3 WB: 0.02, K=3	Sil: 1.00, K=3 WB: 0.01, K=3
Cluster & noise-data	Sil: 0.59, K=4 WB: 0.25, K=4	Sil: 0.69, K=3 WB: 0.17, K=4
Cloud-data	Sil: 0.35, K=4 WB: 0.43, K=6	Sil: 0.50, K=2 WB: 0.31, K=6
Esoph	Sil: 0.24, K=2 WB: 0.54, K=7	Sil: 0.41, K=2 WB: 0.38, K=8
Faithful	Sil: 0.77, K=2 WB: 0.22, K=2	Sil: 0.74, K=2 WB: 0.13, K=6
Infert	Sil: 0.44, K=2 WB: 0.50, K=4	Sil: 0.44, K=4 WB: 0.44, K=4
Quakes	Sil: 0.43, K=3 WB: 0.44, K=5	Sil: 0.54, K=3 WB: 0.26, K=3

We can see that the silhouette is consistently higher for all datasets except the *faithful* dataset, increasing the silhouette by approximately 0.00 to 0.2. The WB-ratio is consistently lower for all datasets, decreasing by approximately 0.00 to 0.16. We conclude that the optimised K-medoids algorithm works.

7.4 RUNTIME COMPARISONS

We compare our hierarchical and K-medoids algorithms to study the effect of the different clustering methods on the final clustering. The results can be found in

:

*Table 20: Runtime comparison between optimised hierarchical and K-medoids. Both have been run on the same Windows 10 laptop, using the average of 20 runs. *: 5 runs used because of time constraints. The swap phase has been turned off for the K-medoids algorithm. The resulting weight vector is not influenced significantly by this change.*

Dataset	**Optimised Hierarchical**	**Optimised K-medoids, K=2**	**Optimised K-medoids, K=7**	**Optimised K-medoids, K=20**
Esoph	1.53	1.18	0.65	1.15
Faithful	9.5	0.35	0.26	7.25
Infert	2.15	4.56	3.98	3.4
Mtcars	1.95	0.62	0.68	0.3
*Quakes**	146.8	73.7	134.9	86.2

In general, the optimised K-medoids algorithm take less time to compute than the optimised hierarchical algorithm. The only exception to this trend seems to be the *infert* dataset, where the optimised hierarchical algorithm takes about half the time to complete. In the other cases, the K-medoids algorithm provides a significant decrease in computation time. However, note that K-medoids needs to be computed for multiple *K* to get a reasonable estimate for a good *K*. If the optimised K-medoids algorithm needs to be run multiple times, the optimised hierarchical algorithm will be faster.

One possibility to compensate for this is to run the standard K-medoids algorithm to get an estimate of *K*, then use that estimate to run the optimised K-medoids algorithm. Another possibility is to use the estimate of the weight vector from a previous run of the optimisation algorithm as the initial guess for the optimum for the next iteration. This should reduce the number of updates required to attain a local optimum if the optimal weight vectors tend to be similar.

Finally, it is interesting to note the large and inconsistent influence of *K* on the computation time of the K-medoids algorithm. We observe no clear pattern that holds for all datasets. For the *esoph* and *faithful* datasets it is the lowest at $K = 7$, for $K = 20$ this occurs with the *infert* and *mtcars* dataset, and finally the *quakes* dataset finishes the quickest when $K = 2$. Further inspection of the *faithful* dataset and the relation between *K*, the average computation time and the target function gives us Figure 22:

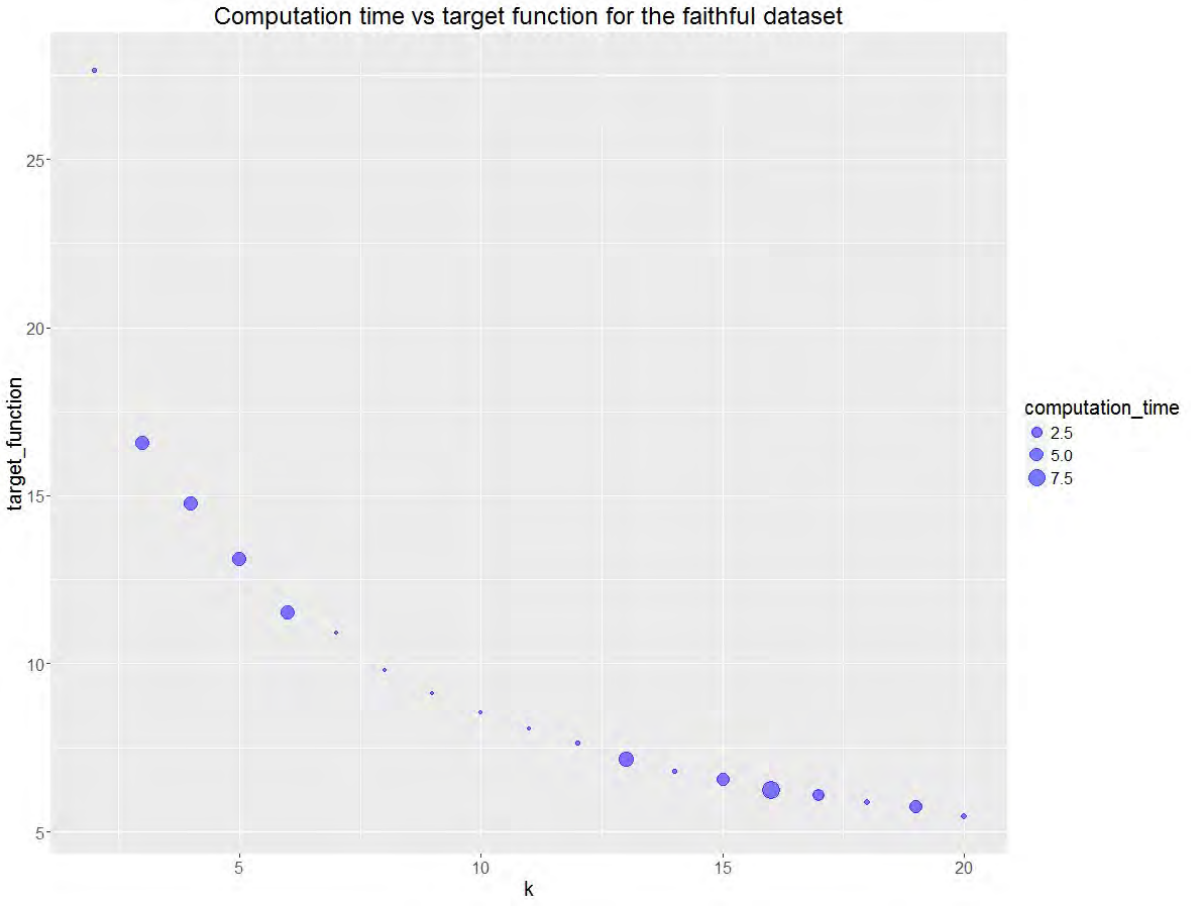


Figure 22: Influence of K and the computation time on the target function. The results are generated on a different PC than those from Table 20, resulting in a different computation time. Why the result for K = 20 is significantly different is unclear.

The relation between *K* and the target function is clear: as *K* increases, the target function decreases. The relation between the computation time and either one of the two other variables is unclear. Further investigation will be required to find the cause of the high variability in computation times.

8 WEIGHT SETS

We want to investigate the possibility of setting the weights of a selection of variables to the same value whilst still performing our optimisation algorithm. This approach could be used when we prefer two or more variables to have the same impact on the distances relative to each other. An example of such a set of variables could be the height and width of an object. It would also require the algorithm to perform fewer computations to update the weight vector since it will contain fewer unique weights. This method could also help to prevent one weight from dominating the rest. The downside would be that this method would probably decrease the value of the CPCC since the BFGS method will have less freedom to manoeuvre the weight vector.

The target function we use is the standard CPCC, with the exception that variables that are grouped use the same weight. We have to derive the formula for the derivative in the case of combined weights. Before we show the derivation, we define the following variables alongside those already defined in Sect. 4.3:

- V_k : the set of variables for which we will be using weight w_k.
- $|V_k|$: the size of set V_k.
- $\sigma_{ijk} = \sum_{z \in V_k} S_{ijz}$

The derivation of this derivative is similar to the derivation explained in Sect. 4.3. The complete derivation is presented in Appendix 12.3. The final CPCC derivative is the following, from Equation 19:

$$\frac{\partial c}{\partial w_k} = \sum_{i<j} \left(\sum_{z \in I_k} \sigma_{ijk} \right) \left(\frac{b(i,j) - a(i,j) \frac{\sum_{e<d} a(e,d) b(e,d)}{\sum_{e<d} a(e,d)^2}}{\sqrt{\sum_{e<d} a(e,d)^2 \sum_{e<d} b(e,d)^2}} \right).$$

The only difference between Equation 13 and Equation 12 is that S_{ijk} is replaced by the sum of S_{ijk} over all variables that use weight w_k for each pair i,j.

9 CONCLUSION

The primary conclusion of this thesis is that the optimization of the weights for the Gower's metric led to better clustering results if compared to the standard Gower's metric. We concluded that by systematically studying multiple artificial and real datasets. In short, our algorithm uses the limited memory Broyden–Fletcher–Goldfarb–Shanno with bounds to select the optimal weight vector for Gower's metric such that it maximises the cophenetic correlation coefficient (CPCC). Our methodology also improved the silhouette, within-between ratio (WB-ratio), Dunn's index, and Pearson's gamma correlation, if compared to the standard implementation.

We compared our algorithm with two benchmark algorithms: the standard hierarchical and K-medoids algorithms. In all twelve datasets used here we observed an increase in the CPCC. For artificial and real test datasets, we noted a significant increase/decrease in the silhouette and WB-ratio respectively when our optimised hierarchical clustering algorithm is compared with the standard hierarchical and K-medoids algorithms. We also further confirmed these good results by using posthoc statistical tests, such as Quade's test. Lastly, we statistically proved that our optimised hierarchical clustering algorithm performs better than all the benchmark algorithms tested in this thesis. We successfully extended the optimised hierarchical clustering setup to construct an optimised K-medoids clustering algorithm. Although more investigation is needed for the optimised K-medoids algorithm, we obtained an improvement of its target function of approximately 25%. The silhouette and WB-ratio also improved for the K-medoids algorithm. Finally, we showed that it is possible to analytically obtain a derivative of the CPCC even when some weights are set constant relative to each other.

Regarding our final dataset, obtained from a recycling mobile app, we found four main clusters and nine smaller clusters. The main clusters are organised as follows:

1. One large cluster consisting of 956 users who used the app on the first day and then stopped using it. They favour the *education* feature of the app.
2. One cluster about half the size of the first cluster of active users who on average used the app multiple times over the course of a few months. Besides favouring the *education* part they have a distinct higher usage of the *pickup* feature.
3. Two small clusters of 47 and 24 users that only used the app very intensely on the first day. They are similar to the first cluster of inactive users in most other aspects.

Finally, the results discussed in this thesis demonstrate the possibilities of choosing optimal weights for the quality of the clustering and the understanding of the importance of each weight regarding the clustering results.

10 DISCUSSION & RECOMMENDATIONS

In this section, we focus on the following topics:

- The influence of dominating weights on the quality of the clustering.
- The speed and convergence of our algorithm.
- Alternatives to hierarchical clustering and other ways to optimise the weights for Gower's metric.

We end with a number of possible research questions that can be studied in the future.

Our hypothesis is that a balanced weight vector leads to an adequate clustering. The results regarding improving the CPCC and other metrics is shown in Sect. 6.2 and the weight vector can be found in Appendix 12.4. However, in the case where one weight is significantly larger, we advise to employ other approaches to determine the quality of the clustering. For instance, a simple approach is to raise the lower bound for the CPCC from 0.7 to 0.8. Another solution would be to run a clustering using equal weights for Gower's metric to get a first-order estimation of the difference in CPCC between the standard clustering algorithm and our optimised algorithm. If the difference is too large, we advise to inspect the cluster to ensure representability.

Another possibility to reduce the influence of dominant weights is to set stricter bounds on the weights themselves. Setting a stricter upper bound allows us to control the maximum value of the weights. More interesting to discuss is if we want to set the lower bound to zero. The advantage would be that we give the weights the maximum amount of freedom to manoeuvre, getting the highest chance of obtaining a clustering with a high CPCC. Furthermore, if some weights were set to zero, we could potentially exclude them from a future clustering attempt once we obtain new data, speeding up the calculations. When we tried this on our final dataset, we found that only 15 of the 35 variables got a weight higher than zero whilst not obtaining dominating weights (see Appendix 12.4). This allows us to study the zero weight variables as if they have not been included in the clustering, analysing whether or not there exist differences in distribution across the clusters.

One possible downside of setting the bounds to [0, 1] is that we might exclude variables that are important in identifying differences between existing clusters and we would not be able to detect this. As an example, see the *box* dataset in Appendix 12.1. In this dataset, the removal of one of the variables could result in the merging of clusters, though this did not happen for this example. The value of the smallest weight was set almost equal to zero, around 10^{-8}, although the option to set it to zero was available. We advise to always inspect the weight vector for dominating weights when using this algorithm. Furthermore, we recommend to confirm that the resulting clusters are actual clusters, especially in the case of dominating weights.

In our opinion, two important points that require further study are speed and convergence of our algorithm. The $O(n^3)$ complexity of hierarchical clustering is most likely the main cause of the long computation time. One possible way this might be solved is by filtering out some of the instances before running our optimised hierarchical. This procedure leads to a decrease on the required computation time. One way to achieve this is by using the DB-scan algorithm (Ester, et al., 1996) and filtering out the points that the DB-scan algorithm classifies as noise. The dataset without these noise instances could then be clustered faster. One could also extend our algorithm to work with the DB-scan method, similar to how we extended the algorithm to K-medoids.

Another way to reduce computation times would be to use parallelisation techniques to perform the hierarchical clustering faster. Furthermore, parallelisation can also be integrated into the quasi-Newton approach by calculating multiple evaluations of the CPCC in parallel. We have experimented with parallelisation for the calculation of S_{ijk} in the derivative of the CPCC. Our preliminary results suggest that parallelisation may allow for a significant performance increase, but it would require significant knowledge on the subject. Further optimisation of the parameters of the line search method employed by the quasi-

Newton method could also help reduce the number of CPCC evaluations needed to achieve the optimal value of the CPCC. GPU computation could also be employed to do the hierarchical clustering. **R** already has packages that are capable of doing this, such as the *rpud* package. Tests using this package indicate a significant decrease in computation times for the hierarchical clustering (Yau, 2009-2016).

Decreasing the computation requirements is a necessity if this algorithm is to be applied to large datasets. In its current form the optimised hierarchical clustering algorithm can handle 1000-2000 instances on a laptop with 4 MB RAM and an i3 processor. However, for larger applications more specialised knowledge is needed to provide optimisations. For instance, integrating the clustering algorithm more efficiently into the BFGS method may help reduce the computation time. From the total runtime, we can see that doing more optimisation on the algorithm would be a requirement since a 20-minutes runtime is relatively long compared to other clustering algorithms. On the other hand, we also perform some variable analysis in the process of the clustering through the weight selection. Note that, in this case, our algorithm has become an optimisation algorithm instead of an evaluation algorithm, which tends to require more computation time.

Since the hierarchical clustering is most likely the main cause of the long computation time, finding a way to ensure we do not have to recalculate the clustering might also improve the computation time. One way to accomplish this could be to generate an approximation of *t(i,j)* (Sect. 3.2) from an initial clustering, updating it similarly to how the Hessian matrix is updated in quasi-Newton methods. Not recalculating the clustering at each step would allow a significant speed increase. The clustering could be recalculated after a certain number of steps to ensure that we do not deviate significantly from the actual *t(i,j)* with our approximation.

One interesting approach to creating an optimal weight vector for Gower's metric would be to use an evolutionary computing approach (Eiben & Smith, 2007). In evolutionary computing, we use the principles of evolution to optimise a certain target function. Individuals (the weight vector for Gower's metric and its fitness) compete with each other and are ranked according to some fitness function (the CPCC). These individuals then produce new solutions by combining their genomes (the weight vector). An advantage of this method would be that we do not require the target function to be continuous, which is a requirement for quasi-Newton methods. A downside of this approach would be that the calculation of the fitness function would take $O(n^3)$ steps, which is far from optimal. Heuristics could help us to overcome this issue.

The global optimisation algorithm that we have proposed can be further optimised. More advanced algorithms could help to improve its results, for instance by implementing a version of taboo search to search for a global optimum (Glover, 1986). One could add a previously selected optimal weight vector to the taboo list and create an area around that vector where we will not allow the optimisation algorithm to go to in a next iteration.

Further investigation into why certain variables get a high weight is also needed. If we can find a pattern for variables that get a high (or low) weight, then we could exploit that pattern to provide the algorithm with a better initial solution, speeding up the computation process. It could also give us an interesting insight into what makes a variable important or unimportant for clustering.

Finally, we would like to present some propositions for interesting future research questions:

- What is the influence of dominating weights on the validity of the clustering?
- Can we speed up the algorithm through the use of GPU computing, parallel programming, and/or integrating the different methods?
- Is it possible to find either an analytical formula or an approximation for *t(i,j)*?

11 REFERENCES

Alcalá-Fdez, J. et al., 2011. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, Volume 17.

Alpaydin, E., 2009. Introduction to Machine Learning. In: *Introduction to Machine Learning*. London: The MIT Press, p. 147.

Alpaydin, E., 2009. Introduction to Machine Learning. In: *Introduction to Machine Learning*. London: TheMIT Press, p. 157.

Azzalini, A. & Bowman, A. W., 1990. A look at some data on the Old Faithful geyser. *Applied Statistics,* Volume 39, p. 357–365.

Breslow, N. E. & Day, N. E., 1980. *Statistical Methods in Cancer Research. Volume 1: The Analysis of Case-Control Studies,* IARC Lyon: Oxford University Press.

Chae, S. S., Kim, J.-M. & Yang, W. Y., 2006. Cluster analysis with balancing weights on mixed-type data. *The Korean communications in statistics*, 13(3).

Cuadras, C. M., Fortiana, J. & Arenas, C., 1998. Some computational aspects of a Distance-Based model for Prediction. *Communication in Statistics- Simulation and Computation*, 25(3).

Eiben, A. & Smith, J., 2007. *Introduction to Evolutionary Computing*. 2 ed. s.l.:Springer.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X., 1996. *A density-based algorithm for discovering clusters in large spatial databases with noise*. Munich, AAAI, pp. 226--231.

Fisher, R., 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics,* 7(2), pp. 179-188.

Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Volume 32.

Gavin, D. G., Oswald, W. W., Wahl, E. R. & Williams, J. W., 2003. A statistical approach to evaluating distance metrics. *Quaternary Research*, Volume 60.

Glover, F., 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research,* 13(5), p. 533–549.

Gonçalves, L. et al., 2008. Comparison of multivariate statistical algorithms to cluster tomato heirloom accessions. *GMR*, 7(4).

Gower, J. C., 1971. A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 27(4), p. 859.

Henderson & Velleman, 1981. Building multiple regression models interactively. *Biometrics,* Volume 37, pp. 391-411.

Holm, S., 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, Volume 6.

Iman, R. L. & Conover, W. J., 1979. On multiple-comparisons procedures.

KEEL, 2011. *KEEL*. [Online]
Available at: http://sci2s.ugr.es/keel/datasets.php
[Accessed 08 03 2016].

Legendre, P. & Legendre, L., 1998. *Numerical Ecology*. 2nd ed. s.l.:Elsevier Science.

Montanari, A. & Mignani, S., 1994. Notes on the bias of dissimilarity indices for incomplete data sets: the case of archaeological classification. *Qüestiió*, 18(1).

Mouchet, M. et al., 2008. Towards a consensus for calculating dendrogram-based functional diversity indices. *Oikos*, Volume 117.

Nemenyi, P., 1963. Distribution-free Multiple Comparisons. *Australian Journal of Statistics*, 33(3).

Nocedal, J. & Wright, S. J., 1999. Numerical Optimization. In: *Numerical Optimization, second edition.* s.l.:Springer.

Pavoine, S. et al., 2009. On the challenge of treating various types of variables: application for improving the measurement of functional diversity. *Oikos,* 118(3), pp. 391-402.

Petchey, O. L. & Gaston, K. J., 2009. Dendrograms and measures of functional diversity: A second instalment. *Oikos*, 118(7).

Podani, J., 1999. Extending Gower's General Coefficient of Similarity to Ordinal Characters. 48(2), pp. 331-340.

Pohlert, T., 2016. *The Pairwise Multiple Comparison of Mean.* [Online]
Available at: https://cran.r-project.org/web/packages/PMCMR/vignettes/PMCMR.pdf
[Accessed 01 03 2016].

Quade, D., 1979. Using weighted rankings in the analysis of complete blocks with additive block effects. *Journal of the American Statistical Association.*

Ramos, H. et al., 2012. Multivariate analysis to determine the genetic distance among backcross papaya (Carica papaya) progenies. *GMR*, 11(2).

R-documentation, n.d. *Locations of Earthquakes off Fiji.* [Online]
Available at: http://127.0.0.1:15008/library/datasets/html/quakes.html
[Accessed 3 6 2016].

R-documentation, n.d. *R: CPU Time Used.* [Online]
Available at: https://stat.ethz.ch/R-manual/R-devel/library/base/html/system.time.html
[Accessed 08 06 2016].

Rocha, M. R., Gaedke, U. & Vasseur, D. A., 2011. Functionally similar species have similar dynamics. *Journal of Ecology*, 99(6).

Rohlf, F. J., 1988. *Numerical taxonomy and multivariate analysis system.* 1.5 ed. New York: Exeter Publishing.

Rohlf, R. R. S. a. F. J., 1962. The Comparison of Dendrograms by Objective Methods. *Taxon*, 11(2).

Rousseeuw, P. J., 1987. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*, Issue 20.

Schaffelke, B. et al., 2011. Water quality in the inshore Great Barrier Reef lagoon: Implications for long-term monitoring and management. *Marine Pollution Bulletin*, Volume 65.

Sokal, P. S. a. R. R., 1973. *Numerical Taxonomy: The Principles and Practice of Numerical Classification.* San Francisco: Freeman.

Sokal, R. & Michener, C., 1958. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin,* Issue 38, pp. 1409-1438.

Sorensen, T., 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biologiske Skrifter,* Issue 5, p. 1–34.

Theodoridis, S. & Koutroumbas, K., 2006. *Pattern Recognition.* Orlando: Academic Press.

Thompson, K. et al., 2009. Little evidence for limiting similarity in a long-term study of a roadside plant community. *Journal of Ecology*, 98(2).

TRICHOPOULOS, D. et al., 1976. Induced arbortions and secondary infertility. *British Journal of Obstetrics and Gynaecology ,* Volume 83, pp. 645-650 .

van den Hoven, J., 2015. *Analyzing Spotify data, exploring the possibilites of user data from a scientific and business perspective,* Amsterdam: VU University.

Wickham, H., 2014. Tidy Data. *Journal of Statistical Software,* August.59(10).

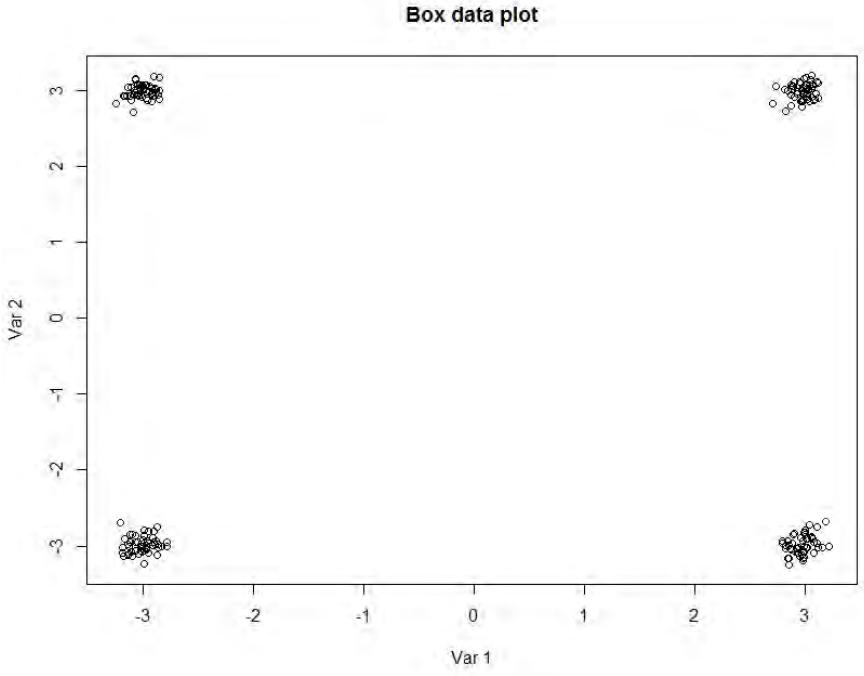
Willmot, C. J., 1981. On the validation of models. *Physical Geography*, 2(2).

Yau, C., 2009-2016. *R-tutor.* [Online]
Available at: http://www.r-tutor.com/gpu-computing/clustering/hierarchical-cluster-analysis
[Accessed 26 April 2016].

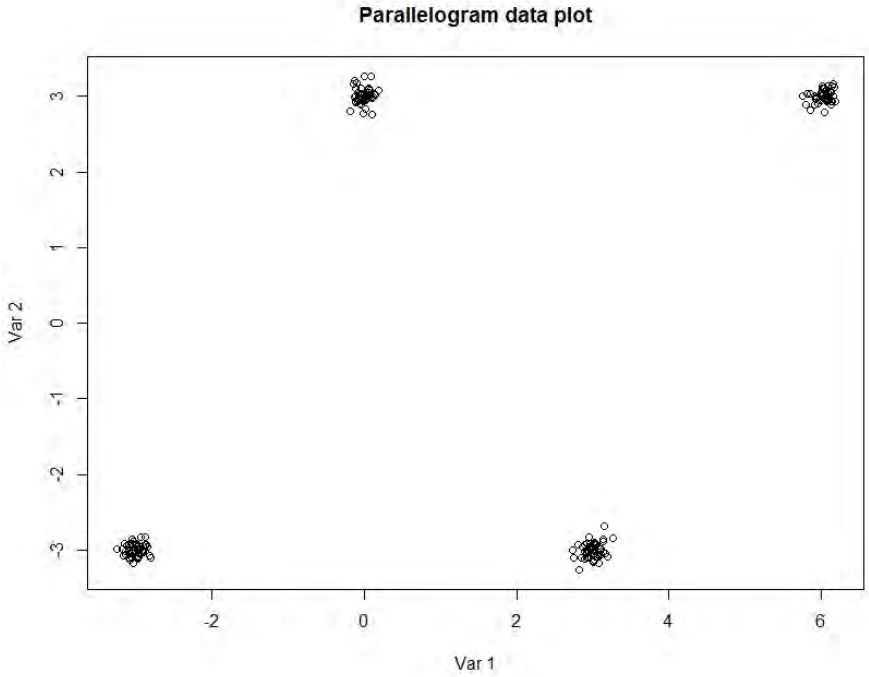
12 APPENDIX

12.1 CUSTOM DATA PLOTS

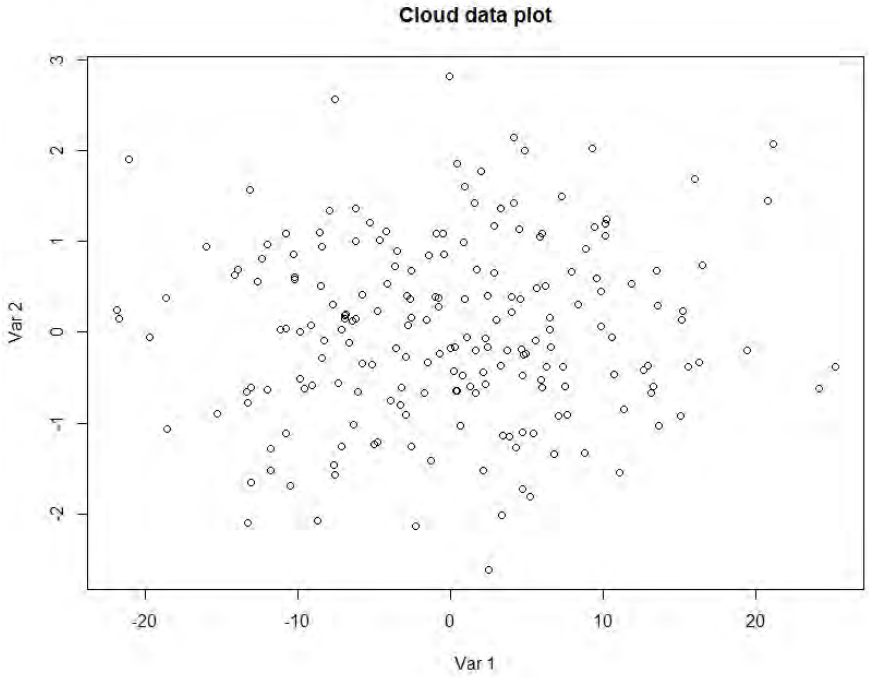
Box dataset



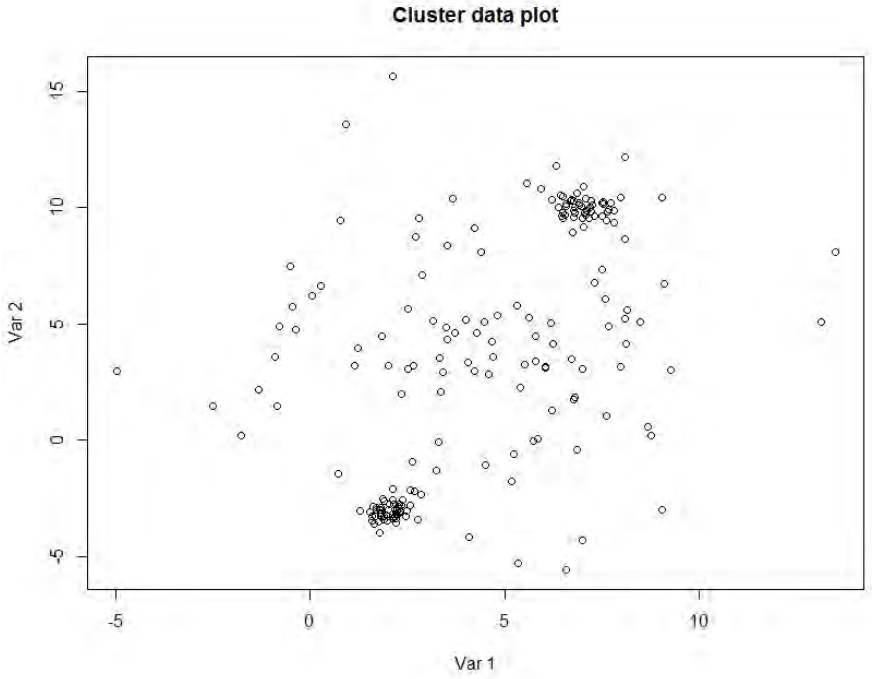
Parallelogram dataset



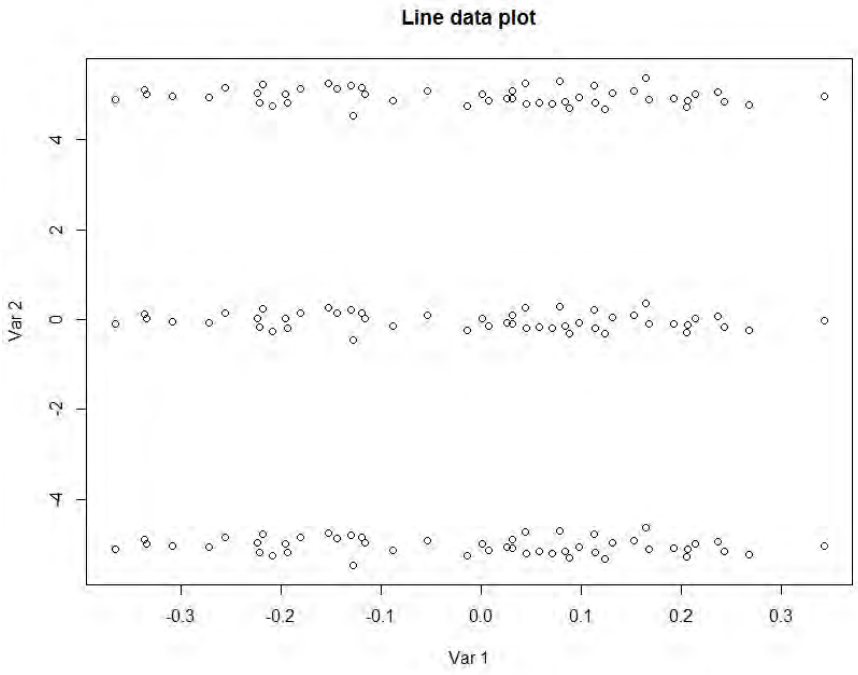
Cloud / noise dataset



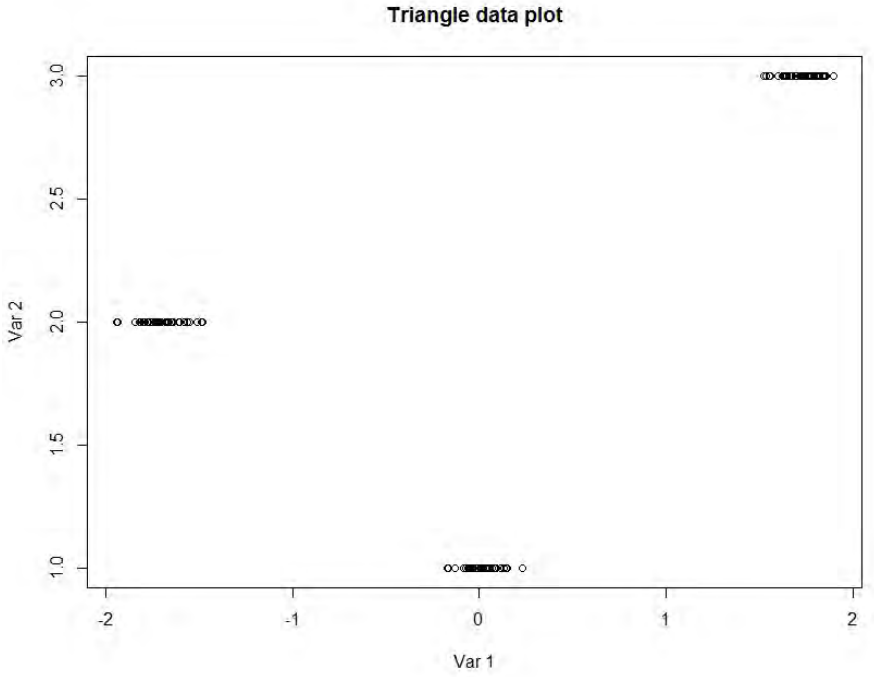
Cluster / 2 cluster + noise dataset



Line dataset



Triangle dataset



Variable 2 is a factor in this plot!

12.2 REAL DATASETS

Esoph

Data from a case-control study of oesophageal cancer in Ille-et-Vilaine, France. (Breslow & Day, 1980)

Format

A data frame with records for 88 age/alcohol/tobacco combinations.

"agegp"	Age group
"alcgp"	Alcohol consumption
"tobgp"	Tobacco consumption
"ncases"	Number of cases
"ncontrols"	Number of controls

Faithful

Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA. (Azzalini & Bowman, 1990)

Format

A data frame with 272 observations on 2 variables.

eruptions	numeric	Eruption time in mins

waiting	numeric Waiting time to next eruption (in mins)

Infert

This is a matched case-control study dating from before the availability of conditional logistic regression. (TRICHOPOULOS, et al., 1976)

Format

Education	years of education
age	age in years of case
parity	count
induced	number of prior induced abortions
case	case status
spontaneous	number of prior spontaneous abortions
stratum	matched set number
pooled.stratum	stratum number

Mtars

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). (Henderson & Velleman, 1981)

Format

A data frame with 32 observations on 11 variables.

mpg	Miles/(US) gallon
cyl	Number of cylinders
disp	Displacement (cu.in.)
hp	Gross horsepower
drat	Rear axle ratio
wt	Weight (1000 lbs)
qsec	1/4 mile time
vs	V/S
am	Transmission (0 = automatic, 1 = manual)
gear	Number of forward gears
carb	Number of carburettors

Quakes

The data set gives the locations of 1000 seismic events of MB > 4.0. The events occurred in a cube near Fiji since 1964. (R-documentation, n.d.)

Format

A data frame with 1000 observations on 5 variables.

lat	Latitude of event
long	Longitude
depth	Depth (km)
mag	Richter Magnitude
stations	Number of stations reporting

12.3 DERIVATIVE OF GROUPED WEIGHTS CPCC

- V_k : the set of variables for which we will be using weight w_k.
- $|V_k|$: the size of set V_k.
- $\sigma_{ijk} = \sum_{z \in V_k} S_{ijz}$

The derivation of this derivative is similar to the derivation explained in Sect. 4.3. We start with the derivative for *a* (the mean-shifted Gower's distance metric) and then continue to the one for *c* (CPCC):

$$\frac{\partial a}{\partial w_k}(i,j) = \frac{\partial x}{\partial w_k}(i,j) - \frac{\partial \bar{x}}{\partial w_k},$$

$$\frac{\partial a}{\partial w_k}(i,j) = -\frac{1}{W^2}|V_k|a(i,j) + \frac{1}{W}\left(\sigma_{ijk} - \frac{|V_k|}{N_s}\sum_{e<d}\sigma_{edk}\right),$$

Equation 18: derivative of a for weight sets.

$$\frac{\partial a}{\partial w_k}(i,j) = \sigma_{ijk} - |V_k|a(i,j) - \bar{\sigma}_k.$$

Substituting Equation 18 into Equation 7 from Sect. 4.3 results in the following derivative:

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j}\frac{\partial a}{\partial w_k}(i,j)b(i,j) - \sum_{i<j}a(i,j)b(i,j)\frac{\sum_{i<j}\frac{\partial a}{\partial w_k}(i,j)a(i,j)}{\sum_{i<j}a(i,j)^2}}{\sqrt{\sum_{i<j}b(i,j)^2\sum_{i<j}a(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j}b(i,j)\left(\sigma_{ijk} - |V_k|a(i,j) - a(i,j)\frac{\sum_{e<d}\left(\sigma_{edk} - |V_k|a(e,d)\right)a(e,d)}{\sum_{e<d}a(e,d)^2}\right)}{\sqrt{\sum_{i<j}b(i,j)^2\sum_{i<j}a(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j}b(i,j)\left(\sigma_{ijk} - a(i,j)\left(|V_k| + \frac{\sum_{e<d}\sigma_{edk}a(e,d)}{\sum_{e<d}a(e,d)^2} - |V_k|\right)\right)}{\sqrt{\sum_{i<j}b(i,j)^2\sum_{i<j}a(i,j)^2}},$$

$$\frac{\partial c}{\partial w_k} = \frac{\sum_{i<j}b(i,j)\left(\sigma_{ij} - a(i,j)\frac{\sum_{e<d}\sigma_{edk}a(e,d)}{\sum_{e<d}a(e,d)^2}\right)}{\sqrt{\sum_{i<j}b(i,j)^2\sum_{i<j}a(i,j)^2}},$$

Equation 19: CPCC derivative for weight sets.

$$\frac{\partial c}{\partial w_k} = \sum_{i<j}\left(\sum_{z \in I_k}\sigma_{ijk}\right)\left(\frac{b(i,j) - a(i,j)\frac{\sum_{e<d}a(e,d)b(e,d)}{\sum_{e<d}a(e,d)^2}}{\sqrt{\sum_{e<d}a(e,d)^2\sum_{e<d}b(e,d)^2}}\right).$$

12.4 WEIGHTS

	Old version, default bounds. CPCC = 0.969	New version, default bounds. CPCC = 0.966	New version, [0, 1] bounds. CPCC = 0.972
daysEngaged	0.00952381	0.00952381	
meanImpDaysSinceThisEventGrpByMaxId	0.00952381	0.00952381	
meanImpEventTypeEducationGrpByMaxIdDivByObsTime	0.00991507	0.04582650	0.05250735
-Invitation-	0.05190488	0.06962005	0.08921243
-Order-	0.03181044	0.05328719	0.07506600
-Pickup-	0.03439012	0.05518380	0.08024161
-Redemption-	0.06123015	0.10221643	0.10204438
meanImpMinutesSincePreviousInput	0.00952381	0.00952381	
meanImpWeekdayFridayGrpByMaxIdDivByObsTime	0.00952381	0.00952381	
-Monday-	0.00952381	0.00952381	
-Satudary-	0.00952381	0.00952381	
-Sunday-	0.00952381	0.00952381	
-Thursday-	0.00952381	0.00952381	
-Tuesday-	0.00952381	0.00952381	
-Wednesday-	0.00952381	0.01048760	0.00321917
obsTime	0.00952381	0.00952381	
sdImpDaysSinceThisEventGrpByMaxId	0.03549686	0.03301265	0.06806624
sdImpEventTypeEducationGrpByMaxIdDivByObsTime	0.08796042	0.06000148	0.04773527
-Invitation-	0.12486736	0.09716310	0.08822677
-Order-	0.08855940	0.06390903	0.06531887
-Pickup-	0.11998497	0.11998497	0.07943582
-Redemption-	0.11544635	0.00952381	0.07817802
sdImpWeekdayFridayGrpByMaxIdDivByObsTime	0.00952381	0.00952381	
-Monday-	0.00952381	0.00952381	
-Satudary-	0.00952381	0.00952381	
-Sunday-	0.00952381	0.00952381	
-Thursday-	0.00952381	0.00952381	
-Tuesday-	0.00952381	0.00952381	
-Wednesday-	0.00952381	0.00952381	
educationPercentage	0.00952381	0.00952381	
-Invitation-	0.00952381	0.00952381	
-Order-	0.00952386	0.01461919	0.03206244
-Pickup-	0.00952381	0.00952381	
-Redemption-	0.02057616	0.02019950	0.05833533
NumberOfUses	0.00833396	0.01108512	0.08035031

Missing values indicate a weight of zero.