ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master's Thesis in Econometrics and Management Science

Business Analytics and Quantitative Marketing

# Convex Clustering of Mixed Numerical and Categorical Data

Author: Thomas Kornelis Gijsbert Brink

Student ID number: 455893

Supervisor:
Dr. C. Cavicchia

Second assessor:
Prof. dr. P.J.F. Groenen

Date final version: July 25, 2021

**Abstract**

Clustering analysis is an unsupervised learning technique widely used for information extraction. Current clustering algorithms often face instabilities due to the non-convex nature of their objective function. The class of convex clustering methods does not suffer from such instabilities and finds a global optimum to the clustering objective. Whereas convex clustering has previously been established for single-type data, real-life data sets usually comprise both numerical and categorical, or mixed, data. In this research, therefore, we introduce the mixed data convex clustering (MIDACC) framework. We implement this framework by developing a dedicated subgradient descent algorithm. Through numerical experiments, we show that, in contrast to baseline methods, MIDACC achieves near perfect recovery of both spherical and non-spherical clusters, is able to capture information from mixed data while distinguishing signal from noise, and has the ability to recover the true number of clusters present in the data. Furthermore, MIDACC outperforms all baseline methods on a real-life data set.

*Keywords*: clustering, mixed data, convex optimization, subgradient descent

# Table of Contents

# 1 Introduction

The goal of data mining is to extract information from data. One of the most common techniques of information extraction is clustering analysis. Clustering analysis is an unsupervised learning technique used to segment observations in clusters, where observations within clusters should be as similar as possible and observations across clusters should be as dissimilar as possible (Jain and Dubes, 1988). As clustering analysis allows users to describe observations and uncover group patterns in data, it has applications in a wide variety of domains (Xu and Wunsch, 2005). Examples include marketing (Punj and Stewart, 1983), biology (Alon et al., 1999), and image segmentation (Pappas and Jayant, 1989). In marketing, for instance, an online supermarket might be interested to cluster (potential) customers in order to design targeted advertisements for specific market segments.

To perform clustering analysis, various algorithms are available. Prominent examples of such algorithms include K-means (Lloyd, 1982; MacQueen, 1967), K-modes (Huang, 1998), Hierarchical Agglomerative Clustering (HAC), see e.g., Ward (1963), and DBSCAN (Ester et al., 1996). Most of such algorithms are confined to clustering data of a single type, i.e., either numerical or categorical data. For instance, K-means is specifically designed for numerical data, while K-modes can be applied to categorical data only. However, in practice, data often comprises both types of data. This is often referred to as *mixed* data. For instance, when clustering customers of an online supermarket, we might have data on customers' number of purchases, the number of products bought, and total spend (numerical features), as well as gender, household type, and city of origin (categorical features). Simply focusing on a single data type incurs information loss. Thus, in order to utilize all available information, being able to cluster observations in mixed data sets is a highly relevant problem.

To solve this problem, several mixed data clustering methods have been developed. For a comprehensive overview, we refer the reader to Ahmad and Khan (2019). A common approach is converting data from one type to the other and applying an existing algorithm to the resulting single-type data set, see e.g., Wei et al. (2015). However, the translation of categorical to numerical data is not straightforward, while discretizing numerical features results in information loss. Therefore, algorithms that directly deal with mixed data, particularly partitional methods, have been of much interest. Examples of such methods include K-prototypes (Huang, 1997, 1998), Partitioning Around Medoids (PAM) with Gower's distance (Gower, 1971), and the feature-weighted K-means algorithm of Modha and Spangler (2003). In addition to partitional methods, hierarchical clustering (Hsu et al., 2007) and model-based methods such as KAMILA (Foss et al., 2016) have been proposed.

Although these types of algorithms are inherently able to deal with mixed data, several complications arise. Namely, due to their non-convex nature, these algorithms fail to guarantee convergence to a global optimum, which limits their applicability and performance. In hierarchical clustering, this issue arises from the heuristic that underlies the formation of clusters. For partitional and model-based clustering methods, such instability originates from the dependence on cluster initializations, which means that different runs of the algorithm can yield completely different results. Furthermore, partitional and model-based methods require the user to preselect the number of clusters that should be formed. However, in practice, users usually do not know what the optimal number of clusters should be. In addition, existing algorithms face issues with combining numerical and categorical features in order to measure the (dis)similarity between observations. Namely, combining features often leads to either favouring a single type of data (Hennig and Liao, 2010) or balancing both types while, given the data, different weights per type are more appropriate (Foss et al., 2016). Furthermore, in common mixed data clustering methods, it can be shown that the shapes of retrieved clusters are limited to spherical and elliptical forms (Foss et al., 2016).

To address the above problems, this research focuses on the technique of convex clustering. This technique, independently introduced by Pelckmans et al. (2005), Hocking et al. (2011), and Lindsten et al. (2011), reformulates the clustering problem as a convex optimization problem. Solving this optimization guarantees convergence to a global optimum, thus removing the issue of cluster instability. In addition, convex clustering allows the user, similar to hierarchical clustering, to assess the stepwise manner in which clusters are formed and select the number of clusters the user is most confident about to represent the 'true' number of clusters. Furthermore, convex clustering is not limited to retrieving spherical or elliptical clusters. Instead, more complex shapes can be retrieved.

Initially, convex clustering was designed for numerical data only. Afterwards, extensions to binary (Choi and Lee, 2019) and exponential-family (Kieskamp, 2019) data were introduced. Only recently, Wang and Allen (2021) proposed an approach to handling different types of data simultaneously. However, they do not demonstrate the workings of their integrative algorithm for mixed numerical and categorical data. In addition, their approach is based on multi-view data, where multiple data sources are integrated to form a single clustering outcome. Per view, the best fitting type of data is chosen. In this paper, however, we focus on directly handling mixed data sets and appropriately combining numerical and categorical features. This leads to the following research question:

- *How can we design a convex clustering method for mixed data that captures the intrinsic structure of numerical and categorical features?*

In answering this question, this paper tries to overcome the issues present in other mixed data clustering algorithms. We do so by introducing the *mixed data convex clustering* (MIDACC) framework. By means of convexifying the clustering objective, MIDACC can establish convergence to a global instead of a local optimum, while being able to recover non-spherical cluster shapes. In addition, MIDACC allows us to select the most likely number of clusters present in the data set. Furthermore, by flexibly combining numerical and categorical data, MIDACC finds a balance between the two data types that is most sensible given the data set at hand. This way, the method is not sensitive to highly noisy data. We implement MIDACC by developing a dedicated subgradient descent algorithm that efficiently clusters mixed data. Applying our methods using synthetic as well as real-life data, we find that MIDACC is able to outperform state-of-the-art and commonly used baseline methods.

This paper is structured as follows. In Section 2, we present an extensive literature review, after which we propose our methods in Section 3. In Section 4, we explain the set-up of our simulation study and real-life data application. Section 5 contains the results of our research, after which we provide conclusions and suggestions for future work in Section 6.

## 2 Literature Review

In clustering literature, a prominent class of methods is that of partitional algorithms such as K-means (Lloyd, 1982; MacQueen, 1967). Although often straightforward to implement, these methods suffer from several drawbacks, most notably that of potentially getting stuck in a local optimum (Berkhin, 2006). This drawback is due to the non-convex optimization problem underlying such methods and leads to different initializations yielding different results (Steinley, 2003). Furthermore, K-means-type algorithms are confined to retrieving spherical clusters. This often leads to poor performance when more complex structures are present in the data (Drineas et al., 2004).

Another group of popular methods includes hierarchical clustering, where observations are merged (agglomerative) or split (divisive), one step at a time, to form clusters (Murtagh and Contreras, 2012). Due to its stepwise nature, hierarchical clustering is inherently greedy. That is, merges or splits made during previous steps of the algorithm cannot be undone. As Zhao and Karypis (2002) note, this leads to hierarchical clustering also not necessarily converging to a global optimum. Thus, just like partitional clustering, hierarchical clustering suffers from instabilities. Furthermore, Fraley (1998) notes that hierarchical clustering, especially in divisive algorithms, may face computational problems due to the combinatorial complexity of the corresponding algorithms.

A family of methods that does not suffer from the problem of instability is that of convex clustering. This type of method formulates the clustering task as a convex optimization problem and finds the 'optimal' clustering by minimizing the corresponding optimization objective. This idea was first proposed by Pelckmans et al. (2005), who minimize the Euclidean loss between observations and cluster centers, where they apply shrinkage with a regularization parameter to merge observations into clusters. By varying the value of the regularization parameter, the number of clusters can be adapted. The idea of convex clustering was, independently, reintroduced by Lindsten et al. (2011) and Hocking et al. (2011). The former applies convex relaxation to K-means in order to obtain an optimization problem, whereas the latter starts from an hierarchical agglomerative clustering set-up and hereafter applies convex relaxation. Hocking et al. (2011) show that convex clustering achieves promising results, being able to outperform methods such as K-means and HAC.

The optimization problem in convex clustering consists of a loss function and a penalty term controlled by a regularization parameter. As Lindsten et al. (2011) state, this regularization parameter controls the trade-off between model fit and the number of clusters. The path of resulting clusters obtained from solving the convex optimization problem for different values of the regularization parameter is often referred to as the *clusterpath* (Hocking et al., 2011). This clusterpath is similar to HAC in that clusters are formed iteratively. However, the important difference is that, in convex clustering, these stepwise merges are based on a new optimization for a different value of the regularization parameter, while merges in hierarchical clustering are based on the previous iteration (Lindsten et al., 2011). This leads to the desirable property that convex clustering is not inherently greedy. Within a specific optimization, the objective function is formulated so as to make sure that clusters merge when their centroids (almost) coincide. This means that, naturally, we only merge clusters when the convex problem draws centroids to each other.

Recently, much attention has been paid to convex clustering and some advantageous properties have been put forward. Most importantly, Hocking et al. (2011) show that convex clustering, in contrast to partitional and hierarchical methods, is able to retrieve non-spherical clusters. Touw et al. (2020) further establish this property by applying convex clustering to a variety of complex cluster shapes. In addition, Kieskamp (2019) uses model selection to demonstrate the ability of convex clustering to retrieve the 'true' number of clusters in the data. This is in contrast to partitional and model-based methods, which require preselecting the number of clusters to be formed.

Furthermore, Tan and Witten (2015) present properties on the prediction error of the method, formalize the link between convex clustering, K-means and hierarchical clustering, and pose a range

of values for the regularization parameter that yields non-trivial cluster solutions. Panahi et al. (2017) use the K-means-based formulation provided by Lindsten et al. (2011) to establish perfect cluster recovery properties. In addition, Sun et al. (2021) extend these perfect recovery properties to a weighted convex clustering model where regularization is only applied to observations with a positive pairwise weight. This weighted convex clustering model is introduced as an extension in both Lindsten et al. (2011), using a local kernel function, and Hocking et al. (2011), using exponentially decaying weights. Furthermore, Radchenko and Mukherjee (2014) prove consistency of convex clustering and derive the convergence rates of this method.

Although the performance of convex clustering methods is promising (Hocking et al., 2011; Chen et al., 2015), its implementation can be computationally cumbersome. Various optimization schemes, such as Newton's method (Kieskamp, 2019) and the off-the-shelf solver CVX (Lindsten et al., 2011) have been applied, while Hocking et al. (2011) propose dedicated algorithms to solve the clusterpath. However, these algorithms do not scale well beyond small data sets. In contrast, Chi and Lange (2015) propose the more efficient Alternating Direction Method of Multipliers (ADMM) and Alternation Minimization Algorithm (AMA) formulations for convex clustering. Both algorithms are widely applicable, though AMA proves to be more efficient. Still, Sun et al. (2021) note that solving large-scale problems is computationally expensive and therefore introduce a semismooth Newton-based augmented Lagrangian method suitable for large-scale problems. Recently, Touw et al. (2020) propose to use diagonal majorization to solve the convex clustering problem, which is, to the best of our knowledge, the fastest technique to date.

Over the past few years, much research has been conducted on applications of convex clustering (Tachikawa et al., 2018; Hiruma et al., 2018; Chi et al., 2020). However, these applications focus mostly on continuous data. Recently, Choi and Lee (2019) proposed to apply convex clustering to binary-valued data by modeling a loss function via the Bernoulli distribution. Kieskamp (2019) extend this idea by presenting a framework to apply convex clustering to distributions from the exponential family. However, these two papers focus solely on a data from a single type.

In contrast, Wang and Allen (2021) proposed the so-called method of Integrative Generalized Convex Clustering Optimization (iGecco) to perform convex clustering on mixed multi-view data. Although this method applies convex clustering to mixed types of data, it does so via integration of multiple data views. This means that multiple data sources (views) are selected, a type of data is selected for each view, and these views are consequently integrated. This approach requires the user to select a single type per data view and computes the corresponding loss function of the multi-view

data in additive fashion. However, single data sets with mixed types of features are not intrinsically analyzed. In addition, no application to combined continuous and categorical data is provided. This paves the way for our research to fill this gap of applying convex clustering to mixed numerical and categorical data in a single data set.

The problem of balancing numerical and categorical data in mixed data clustering is not straightforward. Most partitional methods, e.g., Ahmad and Dey (2007), use dissimilarity measures for both data types, such as Euclidean distance for numerical and Hamming distance for categorical data. Whereas this ensures comparable measurement across features of the same type, given the features are scaled, the relationship between such measures across data types remains unclear (Ahmad and Khan, 2019). For this reason, some methods, such as K-prototypes (Huang, 1997) and the K-means-type clustering of Modha and Spangler (2003), incorporate a type-specific weight. Although this allows the user to control the effect of different data types on the clustering outcome, it does not change the fact that the corresponding similarity measures may be incomparable.

Another option is that of modeling type-specific losses through a likelihood perspective, as in model-based clustering (Hunt and Jorgensen, 2011). This makes sure that the dissimilarity measures for both numerical and categorical data are comparable, or, as Foss et al. (2016) argue, allows us to equitably balance the effect of numerical and categorical data on the clustering outcome. However, we may not always want to equitably balance the influence of both data types, but rather wish to adjust this influence based on the data at hand and thus capture the structure of numerical and categorical data appropriately. To this end, our method combines the partitional and model-based approaches by constructing likelihood-based losses with type-specific weights to control the influence of both data types on the clustering outcome.

# 3 Methodology

In this section, we discuss our approach to convex clustering of mixed numerical and categorical data. First, we provide an introduction into convex clustering. Then, we present MIDACC and its main components, as well as its approach to combining numerical and categorical features. Consequently, we discuss the implementation of MIDACC by providing a subgradient descent algorithm, called MIDACC-SGD, elaborating on parameter selection, and presenting evaluation criteria to assess the clustering performance. Next, we discuss baseline methods and a variable-specific weighted version of MIDACC, or W-MIDACC, after which we briefly mention the software used for this research.

### 3.1 Introduction to Convex Clustering

Let $\mathbf{X}$ be an $(n \times p)$ data matrix, containing $n$ observations $\mathbf{x}_i$, with $i = 1, \ldots, n$, as rows, and $p$ features as columns. Our goal is to cluster the observations such that when $\mathbf{x}_i$ and $\mathbf{x}_s$, $i \neq s$, are similar, they are assigned to the same cluster. Furthermore, let $\mathbf{M}$ be an $(n \times p)$ matrix of centroids, thus containing a $p$-dimensional centroid $\mathbf{m}_i$ for each observation $i = 1, \ldots, n$. We state that observations belong to the same cluster when their centroids merge. In other words, $\mathbf{x}_i$ and $\mathbf{x}_s$ belong to the same cluster iff $\mathbf{m}_i = \mathbf{m}_s$. Convex clustering enforces this type of clustering by means of a convex optimization problem on $\mathbf{M}$. In the (base) case of purely numerical data, as introduced by Hocking et al. (2011) and Lindsten et al. (2011), this problem is defined as follows:

$$\min_{\mathbf{M}} \frac{1}{2} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{m}_i||_2^2 + \lambda \sum_{i<s} ||\mathbf{m}_i - \mathbf{m}_s||_q, \tag{1}$$

where $\sum_{i<s} = \sum_{i=1}^{n-1} \sum_{s=i+1}^{n}$, $\lambda \geq 0$ is the *regularization parameter*, and $||\cdot||_q$ is the penalty norm with order $q$ (Hocking et al., 2011). Common choices for $q$ are 1, 2, and $\infty$, which correspond to the Manhattan, Euclidean, and infinity norm, respectively. In (1), the first term, $\frac{1}{2} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{m}_i||_2^2$, is referred to as the *loss function*, while $\lambda \sum_{i<s} ||\mathbf{m_i} - \mathbf{m_s}||_\mathbf{q}$ is called the *penalty term*.

As previously noted, this formulation works for numerical data only, which is due to the Euclidean loss function. To be able to work with different data types, we can generalize this problem to

$$\min_{\mathbf{M}} \sum_{i=1}^{n} \ell(\mathbf{x}_i, \mathbf{m}_i) + \lambda \sum_{i<s} ||\mathbf{m}_i - \mathbf{m}_s||_q, \tag{2}$$

where $\ell(\mathbf{x}_i, \mathbf{m}_i)$ represents the loss function for observation $i$, which can take multiple forms based on the type of data encountered (Wang and Allen, 2021).

In selecting a loss function, we bear in mind that a convex function should be chosen in order to ensure convexity of (2) in $\mathbf{M}$. This in turn ensures that the objective has a global minimum that we can attain to obtain the 'optimal' clustering. As the name already indicates, this convexity is the main asset of convex clustering over other methods such as K-means and finite mixture models.

Problem (2) bears some similarity with the fused lasso of Tibshirani et al. (2005). Namely, through the regularization parameter, it forces vectors of parameters to merge. When $\lambda = 0$, no regularization takes place and the minimization depends solely on the loss function. This way, we can break down the problem into $n$ piecewise optimization problems: $\min_{\mathbf{m}_i} \ell(\mathbf{x}_i, \mathbf{m}_i)$, $i = 1, \ldots, n$, which, in case we apply the loss function from (1), leads to optimal centroids $\hat{\mathbf{m}}_i = \mathbf{x}_i$.

When we increase $\lambda$, the penalty on unequal centroids increases, such that these centroids are drawn to each other more heavily and may, eventually, merge. Thus, by varying the regularization parameter, we can control the extent to which clusters are formed. Generally speaking, when we increase $\lambda$, more centroids merge, such that the number of clusters decreases. At some point, if $\lambda$ is large enough, all centroids coincide and form a single cluster. This corresponds to the outcome $\hat{\mathbf{m}}_i = \hat{\mathbf{m}}_s$, $\forall i, s$. Note that, in contrast to common clustering methods such as K-means and finite mixture models, we do not specify the number of clusters before optimizing the objective. Instead, we control the number of clusters through $\lambda$.

The way in which fusion occurs over $\lambda$ varies with $q$. Although Hocking et al. (2011) propose algorithms to employ convex clustering for $q = 1, 2$, and $\infty$, the first two choices are most common. Choi and Lee (2019) establish that $q = 1$ is not suitable for clustering of binary data, which arises from the fact that the penalty norm $||\mathbf{m}_i - \mathbf{m}_s||_1$ is separable across dimensions. This leads to only two possible outcomes; either $n^*$ (the number of unique observations) or 1 clusters. As this undesirable property naturally also holds for categorical data, we set $q = 2$.

## 3.2 Mixed Data Convex Clustering (MIDACC)

In this section, we introduce the concept of mixed data convex clustering. Firstly, we provide some general notation, after which we discuss, per component, the MIDACC formulation and elaborate on its approach to balancing numerical and categorical data.

### 3.2.1 Notation

As before, let $\mathbf{X}$ and $\mathbf{M}$ be $(n \times p)$ data and centroid matrices, respectively. $\mathbf{X}$ now consists of both numerical and categorical features, such that we can write: $\mathbf{X} = \begin{bmatrix}\mathbf{X}^{num} & \mathbf{X}^{cat}\end{bmatrix}$. Here, $\mathbf{X}^{num}$ represents the numerical and, similarly, $\mathbf{X}^{cat}$ the categorical part of the data. Note that, by default, we order the data according to the type of features. However, this choice is without loss of generality.

We define the sets $\mathcal{N}$ and $\mathcal{C}$ to contain the numerical and categorical features, respectively. Thus, if $j \in \mathcal{N}$, feature $j$ is said to be numerical, while, if $j \in \mathcal{C}$, feature $j$ is categorical. Zooming in on the categorical variables, we note that different variables may have different numbers of classes. We denote the number of classes for feature $j \in \mathcal{C}$ to be $K_j$.

Since the $K_j$ classes are not naturally ordered, we do not apply clustering directly to the categorical data. Instead, we apply dummy-coding. As each observation belongs to a single class, $K_j - 1$ dummies suffice to describe the entire $j$-th variable. This way, we can denote the entry of observation

$i$ corresponding to the $k$-th class of variable $j$ by $x_{ij}^k$, which takes on value 1 if observation $i$ belongs to class $k$ of feature $j$ and 0 otherwise. In similar fashion, we denote the centroid of observation $i$ corresponding with the $k$-th class of attribute $j$ as $m_{ij}^k$. Note that, as opposed to the data $x_{ij}^k$, these centroids $m_{ij}^k$ are not restricted to values 0 and 1.

We denote the number of numerical and categorical features by $p^{num}$ and $p^{cat}$, respectively, such that $p = p^{num} + \sum_{j=1}^{p^{cat}} (K_j - 1)$. Thus, $\mathbf{X}^{num}$ and $\mathbf{X}^{cat}$ are $(n \times p^{num})$ and $\left(n \times \sum_{j=1}^{p^{cat}} (K_j - 1)\right)$ matrices, respectively. In similar fashion, we define matrices $\mathbf{M}^{num}$ and $\mathbf{M}^{cat}$.

### 3.2.2 Objective Function

To construct the MIDACC objective function, we combine the ideas of model-based clustering and K-prototypes. Namely, as in mixture models (Hunt and Jorgensen, 2011) and the integrative model of Wang and Allen (2021), we add (log-)likelihood-based loss functions to balance the data types. Similar to K-prototypes (Huang, 1997), we control the influence of both data types using a type-specific weight factor. This yields the following formulation:

$$\min_{\mathbf{M}} (1-\gamma) \sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2, \tag{3}$$

where $\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})$ and $\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})$ represent convex loss functions for numerical and categorical data, respectively, and $0 \leq \gamma \leq 1$ is a parameter controlling the relative contribution of numerical and categorical attributes on the clustering task. Furthermore, $\lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2$ is the penalty term with regularization parameter $\lambda \geq 0$ and pairwise weights $\omega_{is} \geq 0$ between observations $i$ and $s$. Provided the pairwise weights are independent of $\mathbf{M}$, this objective function is (additively) convex in $\mathbf{M}$ and can thus be minimized to ensure a global optimum is found.

### 3.2.3 Loss Functions

In the MIDACC objective function (3), we use two loss functions for the two data types. The loss function for numerical data, or $\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})$, is as in (1) and thus given by

$$\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) = \frac{1}{2} ||\mathbf{x}_i^{num} - \mathbf{m}_i^{num}||_2^2, \qquad i = 1, \ldots, n. \tag{4}$$

It is worth observing that this loss function is proportional to the negative log-likelihood of a multivariate normally distributed variable with location parameter $\mathbf{m}_i^{num}$ and identity covariance matrix

($\mathbf{\Sigma} = \mathbf{I}_{p^{num}}$). We prove this property in Appendix A.1.

Following this idea, we take the loss function for categorical data to follow from a multinomial distribution for each, independent, feature. To ensure that the categorical centroids are continuous and unconstrained, which, optimization-wise, is a desirable property, we perform a logit transform on the parameters (i.e., probabilities) of these distributions. Similar to Kieskamp (2019), we can then define the loss function for categorical data, or $\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})$, as

$$\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) = \sum_{j \in \mathcal{C}} \left( \sum_{k=1}^{K_j - 1} -x_{ij}^k m_{ij}^k + \log \left( 1 + \sum_{k=1}^{K_j - 1} \exp\left( m_{ij}^k \right) \right) \right), \qquad i = 1, \ldots, n, \tag{5}$$

for which we provide a derivation in Appendix A.2. It is interesting to draw similarities between our loss functions and model-based clustering methods. Namely, by combining Gaussian mixtures with latent class analysis, these methods typically use similar normal and multinomial likelihood-based loss functions (Hunt and Jorgensen, 2011, Foss et al., 2019).

#### 3.2.4 Penalty Term

Whereas model-based clustering rests upon the selection of a predefined number of clusters, convex clustering uses a regularized penalty term to control the fusion of centroids. In MIDACC, the penalty term is $\lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2$. This term consists of three parts; i) the Euclidean distance between centroids, or $||\mathbf{m}_i - \mathbf{m}_s||_2$, ii) the pairwise weights $\omega_{is}$, and iii) the regularization parameter $\lambda$.

Firstly, as previously discussed, the penalty term functions as a regularization term that penalizes differences between centroids, which we compute using the Euclidean distance. Although the Euclidean distance is not suitable as a loss function for categorical data, the categorical centroids are unconstrained and continuous due to the previously discussed logit transform. This makes the Euclidean distance suitable to compute differences between centroids.

Secondly, instead of putting a similar penalty on each pair of observations, we apply pairwise weights $\omega_{is} \geq 0$ based on the distance between $\mathbf{x}_i$ and $\mathbf{x}_s$ that make sure we cluster based on local density. The intuition behind this is that observations that are highly dissimilar should, any way, not be clustered together. Therefore, their centroid difference should not affect the clustering objective function and possibly distort the clustering outcome. To provide larger weights to 'closer' observations, we use exponentially decaying weights, which can drastically improve performance (Hocking et al., 2011). In addition, we enforce sparsity in the weights, therewith reducing runtime and potentially improving performance, by applying K-Nearest Neighbor (KNN) weights (Lindsten

et al., 2011). In combination, this leads to the following weight specification (Chi and Lange, 2015):

$$\omega_{is} = I_{k,is} \exp\left(-\phi d\left(\mathbf{x}_i, \mathbf{x}_s\right)\right), \qquad i, s = 1, \ldots, n, \tag{6}$$

where $\phi \geq 0$ represents the order of exponential decay and $I_{k,is}$ is an indicator function that takes on value 1 when $i$ belongs to the K-nearest neighbors of $s$ or vice versa, and 0 otherwise. This makes sure our pairwise weights are symmetric in $i$ and $s$, i.e., $I_{k,is} = I_{k,si}$, which means that reshuffling the data does not lead to different weights. This attributes to our method being insensitive to initialization. In other words,

$$I_{k,is} = \begin{cases} 1, & \text{if } i \in \text{KNN}(s) \text{ or } s \in \text{KNN}(i) \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The K-nearest neighbors of observation $i$, or KNN($i$), are defined as the $K$ observations $\mathbf{x}_s$ with the smallest distance $d(\mathbf{x}_i, \mathbf{x}_s)$, where tied observations are all included in KNN($i$). As in (6), $d(\mathbf{x}_i, \mathbf{x}_s)$ stands for the distance between observations $i$ and $s$, for which Gower's distance is suitable (Wang and Allen, 2021) as this measure makes sure distances for all features are of similar magnitude. In our case, we propose to use a weighted version of Gower's distance metric (Gower, 1971), given by

$$d(\mathbf{x}_i, \mathbf{x}_s) = \frac{1-\gamma}{p^{num}} \sum_{j \in \mathcal{N}} \frac{|x_{ij} - x_{sj}|}{\max_r x_{rj} - \min_r x_{rj}} + \frac{\gamma}{p^{cat}} \sum_{j \in \mathcal{C}} I(x_{ij} \neq x_{sj}), \qquad i, s = 1, \ldots, n, \tag{8}$$

where we apply scaling to numerical and categorical distances through $\gamma$. Note that the categorical part (Hamming distance) in the above expression is not calculated on dummy-coded data, but rather on the original data. Although seemingly a minor feat, this is important.

Namely, were we to compute the Hamming distance on dummy-coded data, the range of categorical distances would be $0 - 2$ instead of the $0 - 1$ range on the numerical distance part of (8). This would lead to, naturally, incomparable weights being put on the two data types. In addition, since our dummy-coded data omits the $K_j$-th category of feature $j$, we would enforce an asymmetry in the distances across observations. That is, if feature $j$ of observation $i$ belongs to the $K_j$-th class and feature $j$ of observation $s$ does not, this difference would *not* be fully captured using the dummy-coded data. In contrast, if observations $i$ and $s$ belong to different classes but not to the $K_j$-th class of feature $j$, these differences *would* be fully captured. By working with the original categorical data, we do not encounter this issue.

Note that the asymmetry issue also does not occur in the categorical loss function and the penalty term. Due to the restrictions on multinomial parameters, the loss function requires only $K_j - 1$ parameters in order to capture all information in feature $j$. As for the penalty term, the logit transform $m_{ij}^k = \log\left(\frac{p_{ij}^k}{1-\sum_{l=1}^{K_j-1} p_{ij}^l}\right) = \log\left(\frac{p_{ij}^k}{p_{ij}^{K_j}}\right)$ theoretically assures that $m_{ij}^{K_j} = 0$ for all observations $i = 1, \ldots, n$, such that the penalty contribution of the $K_j$-th class is, by definition, zero. Here, $p_{ij}^k$ is the multinomial probability of class $k$ of feature $j$ for observation $i$ and $p_{ij}^{K_j}$ denotes the probability of the $K_j$-th class of feature $j$ in observation $i$.

Thirdly, in the penalty term, $\lambda$ functions as regularization parameter. When $\lambda = 0$, no regularization on centroids takes place. If we increase $\lambda$, locally different centroids are penalized and forced to merge. At some point, if $\lambda$ is large enough, the penalty term dominates the clustering, which may lead to all centroids fusing and, therewith, the formation of a single cluster. This does not always occur, however, as the set of feasible outcomes may be restricted by the weights $\omega_{is}$. That is, due to the sparse (KNN) weights, the graph with observations as nodes and weights as edges may not be connected. Generally speaking, such restrictions are undesirable, especially in light of *clusterpath* comparisons with hierarchical clustering (Hocking et al., 2011). However, as this research is not concerned with plotting out the clusterpath but rather focuses on finding the 'optimal' clustering given a data set, this phenomenon is outside the scope of this work. Furthermore, we find that sensible choices for $K$ almost always lead to connected weight graphs.

For the no-clustering ($\lambda = 0$) and single-cluster (large $\lambda$) cases, analytic MIDACC solutions of (3) can be derived. We provide derivations of these extreme cases in Appendix B.

#### 3.2.5 Balancing Numerical and Categorical Data

An important problem in mixed data clustering is combining numerical and categorical data in such a way that the influence of both types can be controlled. For reference, see e.g., Foss et al. (2016) and Hennig and Liao (2010). The first step in doing so is making sure that the clustering contributions from features of the same type are on the same scale.

Within categorical features, the feature-specific clustering contribution in MIDACC, see (3), consists of a multinomial loss function, the difference between centroids captured via the penalty term, and pairwise weights. We note that the centroids are constructed based on the probability mass function of a multinomial distribution with a single trial. This ensures that the loss contributions of all categorical variables behave in similar fashion; like a distributional likelihood. In addition, the magnitude of and difference between centroids is controlled by a logit transform of probabilities (i.e.,

the parameters in the multinomial distribution). Again, this type of calculation makes sure that the centroids of different categorical variables are on the same scale. Furthermore, the Hamming distance used for the categorical distance in the pairwise weights ensures that all categorical variables have distance 0 or 1. Of course, the actual contribution of the categorical variables is dependent on the rate of occurrence of variable-specific classes and the number of classes $K_j$ per variable, but these are natural factors that contain valuable information and *should*, therewith, affect the clustering outcome in variable- or class-specific fashion.

As for numerical data, the feature-specific clustering contribution from MIDACC problem (3) consists of the Euclidean loss function, the difference between centroids in the penalty term, and pairwise weights. Firstly, we have shown that the Euclidean loss corresponds with the likelihood contribution from a multivariate normal distribution with identity covariance. However, this assumption implies that the loss is dependent on the scale of measurement and therefore does not equitably balance numerical features. Thus, we choose to scale the numerical features prior to performing the clustering task. Various methods to do so exist, such as min-max normalization and standardization. Since min-max normalization relies on the two most extreme observations (Hennig and Liao, 2010), this method is extremely sensitive to outliers. Therefore, we choose to apply standardization to unit variance, which is given by

$$\tilde{x}_{ij} = \frac{x_{ij}}{s(\mathbf{x}_j)}, \qquad i = 1, \dots, n, \quad j \in \mathcal{N}, \tag{9}$$

where $s(\cdot)$ is the sample standard deviation and $\mathbf{x}_j$ is the vector containing $x_{ij}$, $\forall i = 1, \dots, n$. This standardization scheme ensures that all numerical features are on the same scale and, thus, that their Euclidean loss is comparable. Note that we do not apply mean-standardization; the mean of numerical features should, namely, already be captured through $\mathbf{m}_i$. As we apply scaling throughout the entirety of this research, we do not adopt the $\tilde{x}_{ij}$ notation but rather employ standardization directly when referring to $x_{ij}$. Secondly, as for the penalty term, standardization ensures that the centroid differences are on the same scale. Thirdly, just as with the categorical data, the Gower distance used in (8) ensures that all numerical features have range $0-1$ in the distance metric. This way, all numerical features contribute similarly to the pairwise weights.

Besides controlling the influence of features of the same type, balancing numerical and categorical data is an important task. In MIDACC, we control the interplay between these two types of data by means of $\gamma$. Remember that the MIDACC objective function is given by (3) where $\omega_{is}$ is formed using

(6-8). Also, by setting $0 \leq \gamma \leq 1$ and using weights $(1-\gamma)$ and $\gamma$ for numerical and categorical data, respectively, which is similar to the approach taken by Modha and Spangler (2003), the influence of numerical and categorical features on the clustering process becomes simple and intuitive.

Through $\gamma$, we can control the effect the two types of data have on the clustering outcome. If $\gamma$ is large, i.e., close to 1, categorical features are given a large influence on the clustering outcome as the losses and pairwise weights are scaled towards the categorical data. On the other hand, if $\gamma$ is close to 0, numerical features dominate the clustering process. The two extremes $\gamma = 0$ and $\gamma = 1$ clearly represent the cases in which the clustering depends solely on numerical and categorical data, respectively. In terms of interpretability, this is simpler than the K-prototypes formulation of Huang (1997), which uses weight $\gamma$ for categorical and weight 1 for numerical attributes. Note that, besides through the loss function, $\gamma$ is also present in the distance function (8) used to compute pairwise weights. This ensures that the clustering outcome is not unknowingly largely influenced by the categorical c.q. numerical data through large weight contributions even if $\gamma$ c.q. $(1-\gamma)$ is low.

In case $\gamma = 0.5$, the current formulation assigns equal weights to numerical and categorical features. However, we note that, in contrast to what Foss et al. (2016) pose, simply adding normal and multinomial likelihoods with equal weights does not necessarily equitably balance the influence of numerical and categorical attributes. Instead, by using weights $(1-\gamma)$ and $\gamma$ in conjunction with normal and multinomial likelihoods, we get an interpretable and flexible objective function that is able to equitably balance numerical and categorical attributes. Besides our current MIDACC formulation, we considered a different form where type-specific weights were added in the penalty term. We provide both formulations and a comparison of their workings in Appendix C.

### 3.3 Implementation

In previous literature, several algorithms have been proposed for convex clustering, the most notable methods including ADMM (Chi and Lange, 2015) and gradient-based methods such as *clusterpath* (Hocking et al., 2011). Other methods exist, among which the semismooth Newton-based augmented Lagrangian (SSNAL) method of Sun et al. (2021), but these methods tend to purely focus on reducing computation time, at the expense of flexibility and interpretability. As this research is not so much concerned with designing computationally efficient algorithms but rather focuses on interpretably implementing mixed data convex clustering, we focus on ADMM and gradient-based methods.

The main advantage of gradient-based methods is their simplicity and interpretability, while ADMM is the most commonly used type of algorithm for convex clustering and has been shown

to perform well (Chi and Lange, 2015). In spite of their convergence generally being slower than ADMM, we decide to focus on gradient-based methods and their favorable interpretability. Therefore, we present a subgradient descent algorithm based on that of Hocking et al. (2011) in order to solve the MIDACC problem. As this method is relatively simple and easily interpretable, subgradient descent is perfectly suitable for this research. Still, due to its surge in popularity and computational benefits, we provide an ADMM-type method for MIDACC in Appendix D.

This section provides details on the implementation of MIDACC. We first provide and explain our mixed data convex clustering using subgradient descent, or MIDACC-SGD, algorithm. Next, we discuss parameter selection in MIDACC. Last, we present evaluation criteria to analyze the performance of MIDACC.

### 3.3.1 MIDACC-SGD

The idea of subgradient descent is similar to that of gradient descent; iteratively, we update the to-be-optimized variables based on the gradient of the objective function with respect to those variables. Whereas gradient descent relies on the objective function being differentiable on the complete domain of the variables, subgradient descent can be applied to non-differentiable functions as well.

In MIDACC, we optimize the cluster centroids captured in $\mathbf{m}_i$, $i = 1, \ldots, n$, by minimizing objective function (3), which we denote by $f(\mathbf{M})$. In order to apply gradient-based methods, we require the derivative of this objective function to $\mathbf{m}_i$, or $\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i}$. Since $f(\mathbf{M})$ is the sum of three functions, this derivative can be computed as the sum of three separate derivatives, or,

$$\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i} = (1-\gamma)\frac{\partial \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i} + \gamma\frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial \mathbf{m}_i} + \lambda\frac{\partial \left(\sum_{r<s} \omega_{rs}||\mathbf{m}_r - \mathbf{m}_s||_2\right)}{\partial \mathbf{m}_i}, \quad (10)$$

where we use the fact that loss functions $\ell^{num}(\cdot)$ and $\ell^{cat}(\cdot)$ are separable across observations, and adopt the notation of observations $r$ and $s$ in the penalty so as to prevent confusing these observations with $i$. To further simplify this expression, we note that $\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i} = \left[\left(\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i^{num}}\right)\left(\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i^{cat}}\right)\right]$, where

$$\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i^{num}} = (1-\gamma)\frac{\partial \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i^{num}} + \lambda\frac{\partial \left(\sum_{r<s} \omega_{rs}||\mathbf{m}_r - \mathbf{m}_s||_2\right)}{\partial \mathbf{m}_i^{num}}, \quad (11a)$$

$$\frac{\partial f(\mathbf{M})}{\partial \mathbf{m}_i^{cat}} = \gamma\frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial \mathbf{m}_i^{cat}} + \lambda\frac{\partial \left(\sum_{r<s} \omega_{rs}||\mathbf{m}_r - \mathbf{m}_s||_2\right)}{\partial \mathbf{m}_i^{cat}}. \quad (11b)$$

Computing these gradients, we obtain the following expressions:

$$\frac{\partial \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i^{num}} = -(\mathbf{x}_i^{num} - \mathbf{m}_i^{num}), \tag{12a}$$

$$\frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial m_{ij}^k} = -x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1 + \sum_{l=1}^{K_j - 1} \exp\left(m_{ij}^l\right)}, \qquad j \in \mathcal{C}, \quad k = 1, \ldots, K_j - 1, \tag{12b}$$

$$\frac{\partial \left(\sum_{r<s} \omega_{rs} ||\mathbf{m}_r - \mathbf{m}_s||_2\right)}{\partial \mathbf{m}_i} = \sum_{s \neq i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2}, \tag{12c}$$

for $i = 1, \ldots, n$, where we represent the gradient of the categorical loss function in elementwise fashion. We provide full derivations of these expressions in Appendix E.

From (12c), it is straighforward to notice that the partial derivative of the penalty function is undefined in case $\mathbf{m}_i = \mathbf{m}_s$ for some pair of centroids $i$ and $s$ (with $i \neq s$). This prohibits us from using gradient-based methods such as simple gradient descent and Newton's method. Therefore, we adopt a different approach and rather work with the subgradient of the penalty function, which, for observation $i = 1, \ldots, n$, is given by $g_i\left(\sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2\right)$ and has the form

$$g_i\left(\sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2\right) = \sum_{s \neq i,\ \mathbf{m}_s \neq \mathbf{m}_i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2} + \sum_{s \neq i,\ \mathbf{m}_s = \mathbf{m}_i} \omega_{is} \boldsymbol{\beta}_{is}, \tag{13}$$

where $\boldsymbol{\beta}_{is} = -\boldsymbol{\beta}_{si}$ and $||\boldsymbol{\beta}_{is}|| \leq 1$. This is similar to the approach of Hocking et al. (2011). We provide a derivation of this subgradient in Appendix F. In order to implement the above subgradient, we combine all subgradients for observations belonging to the same cluster. Although Hocking et al. (2011) note that they compute this subgradient per cluster $c$ by summing up the subgradients for all $i \in c$, they in fact calculate the average subgradient. This is the same approach we take, such that the subgradient of the penalty for cluster $c$ is

$$g_c\left(\sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2\right) = \frac{1}{|c|} \sum_{i \in c} \left( \sum_{s \neq i,\ \mathbf{m}_s \neq \mathbf{m}_i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2} \right), \tag{14}$$

as the $\boldsymbol{\beta}_{is}$ terms cancel out due to their symmetry. Here, $|c|$ denotes the cardinality of $c$, i.e., the number of observations in cluster $c$. We can now write the complete subgradient for cluster $c$,

**Algorithm 1:** MIDACC-SGD

**Data:** Data matrix $\underset{(n\times p)}{\mathbf{X}}$; $\phi$; number of neighbors $K$; $\gamma$; $\lambda$; minimum fusion distance $\delta$

**Result:** Optimal centroids $\underset{(n\times p)}{\mathbf{M}^*}$; optimal set of clusters $C^*$

Compute weights $\omega_{is} = I_{k,is}\exp(-\phi d(\mathbf{x}_i, \mathbf{x}_s)), \qquad \forall i, s = 1, \ldots, n$

Initialize t $\leftarrow 1$; $\mathbf{M}^0 \leftarrow \mathbf{X}$; $\mathbf{G}^0 \leftarrow SubGradient(\mathbf{X}, \mathbf{M}^0)$; $C \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}$

**while** *convergence criteria not met* **do**

  $\alpha \leftarrow LineSearch(\mathbf{M}^{t-1}, \mathbf{G}^{t-1}, t)$

  $\mathbf{M}^t \leftarrow \mathbf{M}^{t-1} - \alpha\mathbf{G}^{t-1}$

  $d^* \leftarrow \min_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  $i^*, s^* \leftarrow \underset{i,s}{\text{argmin}} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  **while** $d^* < \delta$ **do**

    $\mathbf{M}^t, C \leftarrow Fuse(\mathbf{m}_{i^*}, \mathbf{m}_{s^*})$

    **if** $|C| = 1$ **then**

      **break**

    $d^* \leftarrow \min_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

    $i^*, s^* \leftarrow \underset{i,s}{\text{argmin}} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  $\mathbf{G}^t \leftarrow SubGradient(\mathbf{X}, \mathbf{M}^t)$

  $t := t + 1$

$\mathbf{M}^* \leftarrow \mathbf{M}^{t-1}$

$C^* \leftarrow C$

denoted by $g_c(f(\mathbf{M}))$, as

$$g_c(f(\mathbf{M})) = \frac{1}{|c|}\sum_{i\in c}\left((1-\gamma)\frac{\partial \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i} + \gamma\frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial \mathbf{m}_i}\right) + \lambda g_c\left(\sum_{i<s}\omega_{is}||\mathbf{m}_i - \mathbf{m}_s||_2\right), \tag{15}$$

where we can fill in the expressions from (12a-12b) and (14) for the loss function gradients and penalty subgradient, respectively. Using the subgradients $g_c(\cdot)$ for all clusters $c \in C$, where $|C|$ denotes the number of clusters, we can perform subgradient descent in MIDACC. The corresponding MIDACC-SGD algorithm, which is similar to that in Hocking et al. (2011), is given in Algorithm 1.

The brief idea is as follows: we initialize the centroid matrix $\mathbf{M}$ to be equal to data matrix $\mathbf{X}$ and $C$ to be the set of $n$ clusters, one for each observation. Then, until a convergence criterion is

**Algorithm 2:** *LineSearch*

**Data:** $\underset{(n\times p)}{\mathbf{M}}$ ; $\underset{(n\times p)}{\mathbf{G}}$ ; $t$

**Result:** Stepsize $\alpha$

Initialize $\alpha \leftarrow 1$; set $\eta \leftarrow 0.5$; calculate $cost^{pre} = f(\mathbf{M})$; initialize $cost^{post} \leftarrow cost^{pre}$

**if** $t(mod\ 10) = 0$ **then**

  $\alpha = \frac{1}{t}$

**else**

  **while** $cost^{post} \geq cost^{pre}$ **do**

    $\alpha = \alpha \times \eta$

    $\mathbf{M}_\alpha \leftarrow \mathbf{M} - \alpha\mathbf{G}$

    $cost^{post} := f(\mathbf{M}_\alpha)$

met, we iteratively update $\mathbf{M}$ in order to find the optimal centroids $\mathbf{M}^*$ and the corresponding set of clusters $C^*$. Note that any other initialization of $\mathbf{M}$ could be used; convex clustering should be independent of initialization and always converge to the global optimum. As input to the algorithm, besides data matrix $\mathbf{X}$, we provide parameters $\phi$, $K$, $\gamma$, $\lambda$, and $\delta$. Thus, Algorithm 1 is run for *fixed* values of these parameters. This is in contrast to Hocking et al. (2011), who perform subgradient descent for a path of values for $\lambda$ in order to retrieve an entire clusterpath.

Within the algorithm, we use the function $SubGradient(\mathbf{X}, \mathbf{M})$ to obtain subgradients $g_c\left(f(\mathbf{M})\right)$ calculated using (15) for each $c \in C$ and stack these subgradients in the $(n \times p)$ matrix $\mathbf{G}$. Here, we fill in the $i$-th row of $\mathbf{G}$, or $\mathbf{g}_i$, by using $\mathbf{g}_i = g_c(f(\mathbf{M})),\ \forall i \in c$.

The function $LineSearch(\mathbf{M}, \mathbf{G}, t)$ calculates the stepsize $\alpha > 0$ of the centroid update along $\mathbf{G}$. We provide pseudocode for this function in Algorithm 2. In short, we use simple backtracking line search alongside random stepsizes each 10th iteration, which we find improves flexibility and speed of the centroid updates. In backtracking line search, we multiply the stepsize by a constant until the corresponding centroid update leads to a decrease in the MIDACC objective function (3).

The fusion process makes sure that we merge similar centroids into the same cluster. In order to fuse observations, we check whether the minimum distance between unequal centroids, or $d^*$, lies below a certain threshold $\delta$. We define $\delta$ to be a fraction, in our case $\frac{1}{2}$, of the smallest distance between data points, such that

$$\delta = \frac{1}{2} \min_{i,s} \left( (1-\gamma)||\mathbf{x}_i^{num} - \mathbf{x}_s^{num}||_2 + \gamma \sum_{j \in \mathcal{C}} I(x_{ij} \neq x_{sj}) \right). \tag{16}$$

Note that this equation is, as opposed to (8), *not* scaled to range $0-1$. This is because $\delta$ should be in the order of magnitude of the centroid differences themselves. In addition, note that we calculate the distance part of categorical features using the original, non-dummy-coded, data $x_{ij}$. Were we to use dummy-coded data, this would enforce asymmetry in distances.

In case $d^* < \delta$, we fuse the centroids $i^*$ and $s^*$ with distance $d^*$. We do so by means of the $Fuse(\mathbf{m}_{i^*}, \mathbf{m}_{s^*})$ function, which performs two actions. First of all, this function merges $\mathbf{m}_{i^*}$ and $\mathbf{m}_{s^*}$. Secondly, it updates $C$. Let $i^* \in c_1$ and $s^* \in c_2$, state that $\mathbf{m}_{i^*}$ and $\mathbf{m}_{s^*}$ fuse to form centroid $\mathbf{m}_r$, and clusters $c_1$ and $c_2$ join to form cluster $c$. The two actions $Fuse(\mathbf{m}_{i^*}, \mathbf{m}_{s^*})$ performs are then represented by

$$\mathbf{m}_r = \frac{|c_1|\mathbf{m}_{i^*} + |c_2|\mathbf{m}_{s^*}}{|c_1| + |c_2|}, \qquad \forall r \in c_1, c_2, \tag{17a}$$

$$C = (C \backslash (c_1 \cup c_2)) \cup c, \tag{17b}$$

such that we take a weighted average of merging centroids to find the fused centroid. This is sensible and possible due to the continuity we ensure in both the numerical and categorical parts of the centroids. As for the cluster set $C$, we simply remove $c_1$ and $c_2$ while adding $c$.

Note that, in contrast to Hocking et al. (2011), MIDACC-SGD allows multiple centroids to be merged in a single iteration. We find this barely affects the outcomes of the algorithm and comes with stark computational benefits. Also, logically, we do not run the $Fuse(\cdot)$ function in case the number of clusters is 1. Furthermore, it is apparent that we run MIDACC-SGD for a single value of $\lambda$. This is in contrast to Hocking et al. (2011), who are interested of the behavior of the clustering outcome with varying values of $\lambda$. As their algorithm uses the solutions for past values of $\lambda$ for a new run with a larger value of $\lambda$, cluster splits are not taken into account, such that there is no security a global optimum is reached. In our case, we do not face this issue, since we start from scratch with each value of $\lambda$ we put into MIDACC-SGD.

As for convergence, we use two criteria to determine whether to terminate MIDACC-SGD. Firstly, we use a criterion on the number of iterations. Specifically, we terminate MIDACC-SGD when $t = 1000$. As, in this case, termination does not ensure convergence, this criterion is undesirable. In none of our experiments, we have observed this criterion to be reached. Secondly, we use a criterion on the squared Frobenius norm of the subgradient matrix, or, $\frac{||\mathbf{G}||_F^2}{np} < \epsilon$. Since $\mathbf{G}$ has $n \times p$ elements, we scale the norm by means of this factor to ensure scalability to any data set. We define $\epsilon$ to be a user-specific threshold. Hocking et al. (2011) use an absolute threshold, e.g.,

$\epsilon = 0.001$. However, we find that scale is quite influential on the algorithm and rather recommend using a relative threshold. As taking a percentage of the initial gradient would make convergence still highly dependent on initializations and on the data set at hand, we use a different approach to making MIDACC's convergence relative to the data. Namely, we scale our loss functions by a *loss factor*. More specifically, we calculate the loss for both data types in case we would end up with a single cluster, take the mean of those losses, and divide both loss functions by this loss factor.

Note that the above choice does not affect the workings of MIDACC at all. To be precise, applying such a loss factor only affects MIDACC through the magnitude of the losses and gradients. The effect on the magnitude of the loss function affects the interplay with the penalty term, but this effect disappears due to our freedom in choosing $\lambda$. As for the gradient, the loss factor has effect on convergence and on the line search parameter $\alpha$. Since we use backtracking line search, which is not dependent on scale ($\alpha$ can take on any kind of value), the line search is unaffected. As for convergence, we find that applying the loss factor yields desirable convergence properties with $\epsilon = 0.00001$. Therefore, we use this setting for the remainder of this research.

### 3.3.2 Parameter Selection

To apply the proposed methods, we need to define several parameters. The previous section already elaborated on setting $\delta$ and $\epsilon$. This section discusses the parameters $\phi$, $K$, $\gamma$, and $\lambda$.

**Choosing $\boldsymbol{\phi}$ and $\boldsymbol{K}$.** The parameters $\phi$ and $K$ form the exponential decay and sparsity parameters in the pairwise penalty weights $\omega_{is}$, $\forall i, s = 1, \ldots, n$. For $\phi = 0$, we obtain uniform weights (over all $K$ nearest neighbors), while increasing $\phi$ leads to increasingly smaller weights between distant pairs of observations. In this research, we use $\phi = 2$, which means that weights decrease quadratically with increasing distance. Note that there is no 'true' guideline on which value of $\phi$ to choose; across literature, many different values are used. We find changing the value of $\phi$ does not affect the clustering outcome much. The number of nearest neighbors $K$ controls the degree of sparsity in the weights. Since a small value of $K$ can drastically improve computation time and clustering performance, and since this research does not focus on retrieving a full clusterpath, we choose $K$ relatively low. More specifically, we opt for using $K = \frac{n}{10}$, such that we apply 90% sparsity. From our experiments, we do not observe this choice to ever lead to an unconnected weight graph.

**Selecting $\boldsymbol{\gamma}$.** As previously discussed, the parameter $\gamma$ controls the relative impact of numerical and categorical features on the clustering outcome. When $\gamma$ is small (close to 0), numerical features dominate, while categorical features are given more importance in case $\gamma$ is high (close to 1). By

varying $\gamma$, we can obtain largely different clustering results. Therefore, it is important to sensibly set this parameter. Below, we provide various choices, for which we provide results in Section 5.1.4 and, more extensively, in Appendix G.

An easy start would be to use $\gamma = 0.5$, which leads to equal weights on the two data types. However, we find that equal weights do not necessarily imply equal contributions to the clustering outcome, as opposed to what Foss et al. (2016) state. Even so, such a property would not be desirable as, depending on the data, either one of the data types could be more important than the other.

Therefore, we wish to adaptively set $\gamma$ based on the data set at hand. To do so, we follow the line of reasoning of Szepannek (2018), who implements several heuristics to compute a similar parameter in K-prototypes. Such heuristics compute this parameter using $\gamma = \frac{h^{num}}{h^{num}+h^{cat}}$, where $h^{num}$ and $h^{cat}$ are measures of dispersion for the numerical and categorical attributes, respectively. Within K-prototypes, Szepannek (2018) uses

$$h_{num} = \frac{1}{p^{num}} \sum_{j \in \mathcal{N}} s(\mathbf{x}_j), \qquad \text{or} \qquad h_{num} = \frac{1}{p^{num}} \sum_{j \in \mathcal{N}} s^2(\mathbf{x}_j), \tag{18a}$$

$$h_{cat} = \frac{1}{p^{cat}} \sum_{j \in \mathcal{C}} \left( 1 - \sum_{k=1}^{K_j} \left( \hat{p}_j^k \right)^2 \right), \qquad \text{or} \qquad h_{cat} = \frac{1}{p^{cat}} \sum_{j \in \mathcal{C}} \left( 1 - \max_k \hat{p}_j^k \right). \tag{18b}$$

However, since our concern with $\gamma$ evolves around the loss functions and the Gower distance that governs the pairwise penalty weights instead of dissimilarity measures as in K-prototypes, the above settings may not be appropriate. Still, we include results for these settings in Appendix G. Please note that, since we scale numerical data to unit variance, we have $s(x_j) = s^2(x_j) = 1$, $\forall j \in \mathcal{N}$, such that we consider two (instead of four) combinations of dispersion measures.

Instead of using K-prototypes-type measures, we could use dispersion measures linked to our loss functions. For this cause, we apply the set-up of Wang and Allen (2021), who propose to scale data types based on the loss achieved in case a single cluster would be formed. We can find the corresponding $h^{num}$ and $h^{cat}$ by calculating (4) and (5) using centroids $\mathbf{m}^{num}$ and $\mathbf{m}^{cat}$, respectively, from Appendix B.2. We note that our loss functions scale linearly in the number of features, such that a for-variable-adjusted dispersion measure would be more appropriate. Therefore, we compute the corrected $h^{(num)*}$ and $h^{(cat)*}$ as $\frac{h^{num}}{p^{num}}$ and $\frac{h^{cat}}{p^{cat}}$, respectively, and compute $\gamma = \frac{h^{num}}{h^{num}+h^{cat}}$. As this seems to generally work well, we implement this method as our standard approach. Similarly, we can calculate $h^{num}$ and $h^{cat}$ as the total Gower distance per data type across all observations, again correct these measures for the number of features per data type, and calculate $\gamma = \frac{h^{(num)*}}{h^{(num)*}+h^{(cat)*}}$.

The advantage of the above heuristic methods is that we compute $\gamma$ prior to running MIDACC, which makes their implementation efficient. However, due to their heuristic nature, these options may not lead to an optimal or even desirable value of $\gamma$. Therefore, we also consider optimizing $\gamma$. Note that we cannot simply minimize the MIDACC objective function across $\gamma$, as the interplay between loss and penalty is different for every value of $\gamma$. However, we do propose two other methods.

Firstly, we could use a model selection approach inspired by the Bayesian Information Criterion (BIC) of Schwarz et al. (1978). The BIC originally is given by

$$BIC = -2\ell(\mathbf{X}, \boldsymbol{\theta}) + k \log(n), \tag{19}$$

where $\boldsymbol{\theta}$ is a vector (or matrix) of parameters, $N$ is the total number of observations, $\ell(\cdot)$ is a log-likelihood function, and $k$ denotes the number of parameters the model estimates. Minimizing this criterion yields the 'best' model. In our case, we apply model selection with $\gamma$ using data sets for which the number of clusters is fixed beforehand, such that the second term in the above equation is constant. Thus, we simply consider minimizing the model selection criterion $mod_\gamma$ given by

$$mod_\gamma = -\ell(\mathbf{X}, \boldsymbol{\theta}) = \ell^{num}(\mathbf{X}^{num}, \mathbf{M}^{num}) + \ell^{cat}(\mathbf{X}^{cat}, \mathbf{M}^{cat}), \tag{20}$$

where we use the fact that our loss functions represent a negative log-likelihood. Please note that we omit $(1-\gamma)$ and $\gamma$ as factors for the loss functions, which we do so as to make the criterion comparable across $\gamma$. We observe this criterion to generally work relatively well. Still, we note that this approach requires multiple evaluations for various values of $\gamma$ on a single data set in order to select $\gamma$. In addition, we would need to further establish the properties of this model selection criterion in order to formally justify using this procedure. Therefore, we decide against using this method. However, we do provide results for this method in Section 5.1.4 and Appendix G.

Secondly, we could try to optimize $\gamma$ together with $\mathbf{M}$, either simultaneously or sequentially. Were we to perform simultaneous optimization, then we would perform MIDACC-SGD with parameter vector $\underset{(1\times(np+1))}{\boldsymbol{\theta}} = [\text{vec}(\mathbf{M}) \quad \gamma]$ instead of the original set-up where $\boldsymbol{\theta} = \mathbf{M}$. Further research is needed in order to establish whether the objective function (3) is convex in the combined parameter vector $\boldsymbol{\theta}$. However, were we to use dense pairwise weights $\omega_{is} = \exp(-\phi d(\mathbf{x}_i, \mathbf{x}_s))$, we know that the objective function is separately convex in both $\mathbf{M}$ and $\gamma$, the latter of which we prove in Appendix H. This means we could sequentially optimize both parameter groups. In short, after each optimization step of $\mathbf{M}$, we update $\gamma$ such that the objective function is decreased most. Since we have restricted

$0 \leq \gamma \leq 1$, we could sequentially optimize $\gamma$ for instance by applying a Golden Section search on this fixed interval. However, more research is needed to inspect the properties of such a two-step method; namely, this method should converge in both $\mathbf{M}$ and $\gamma$, which again depends on convexity of the total objective function $f(\mathbf{M}, \gamma)$. Furthermore, we do not use dense weights but rather apply sparsity in the weights, which both increases performance and reduces runtime (Chi and Lange, 2015). Therefore, for now, we decide against using the above optimization methods but rather stick with the heuristic methods and leave optimization of $\gamma$ for future research.

**Selecting $\boldsymbol{\lambda}$**. Lastly, we need to specify the parameter $\lambda$, which is the regularization parameter in MIDACC. This parameter controls the trade-off between model fit and the number of clusters in the convex clustering outcome. Some papers use this parameter to form a clusterpath, i.e., the solution path for different values of $\lambda$ is presented as an altogether hierarchically structured outcome of the clustering model. Other papers select $\lambda$ based on some criterion. In this research, we are not so much interested in investigating an entire solution path, but rather wish to evaluate the performance of the 'best' model. We do so in two settings.

Firstly, in most experiments presented in this paper, we compare MIDACC with baseline methods on data sets for which we know the true number of clusters before running the clustering. To ensure fair comparison between MIDACC and the other methods, the latter of which require selecting $|C|$ prior to running the clustering algorithm, we set $\lambda$ such that we obtain this true number of clusters. In this case, we use a 'smart' searching heuristic that uses the structure in $\lambda$ to run MIDACC as efficiently as possible. That is, for larger values of $\lambda$, more centroids will merge, such that the number of clusters will be smaller. We make use of this property to start our search with a certain value of $\lambda$ and increase c.q. decrease this value in case the retrieved number of clusters is larger c.q. smaller than the true number of clusters. Based on previous iterations, we can start our search using a value of $\lambda$ that is in the approximate range of values of $\lambda$ that yield the desired outcome. Please note that multiple values of $\lambda$ can correspond with retrieving the same number of clusters. We have experienced only minuscule differences between the models corresponding with these values, in terms of the merged observations, to exist, which confirms the suitability of our simple search for a single value of $\lambda$ yielding the desired number of clusters.

Secondly, we consider the case in which we let MIDACC adaptively select the 'optimal' number of clusters given a data set. That is, we perform model selection to choose a value of $\lambda$ that corresponds with the 'optimal' trade-off between model fit and the number of clusters retrieved. For such a model selection problem, multiple methods exists, such as stability selection (Meinshausen

and Bühlmann, 2010) and information criterion approaches (Tan and Witten, 2015). As stability selection approaches are known to be computationally expensive, we decide to focus on information criteria. Following Choi and Lee (2019) and Tan and Witten (2015), we employ the BIC in order to find the optimal $\lambda$ (Schwarz et al., 1978). Originally, we can write this criterion as (19). We note that, in our case, the log-likelihood, where $\boldsymbol{\theta} = \begin{bmatrix}\mathbf{M}^{num} & \mathbf{M}^{cat}\end{bmatrix}$, is given by

$$\ell(\mathbf{X}, \boldsymbol{\theta}) = -\left((1-\gamma)\ell^{num}(\mathbf{X}^{num}, \mathbf{M}^{num}) + \gamma\ell^{cat}(\mathbf{X}^{cat}, \mathbf{M}^{cat})\right). \tag{21}$$

Since the sum of parameters $(1-\gamma)$ and $\gamma$ equals 1 instead of 2 (which is the case in regular likelihood-based models), we multiply the above loss by a factor 2 in order to retrieve a likelihood that is on the appropriate scale. Furthermore, the number of observations is $n$, while the number of parameters that we estimate equals the number of clusters times the number of parameters, or, equivalently, $|C| \times p$. Thus, we calculate the BIC using

$$BIC = 4\left((1-\gamma)\ell^{num}\left(\mathbf{X}^{num}, \mathbf{M}^{num}\right) + \gamma\ell^{cat}\left(\mathbf{X}^{cat}, \mathbf{M}^{cat}\right)\right) + |C| \times p \times \log(n). \tag{22}$$

This equation resembles that of Kieskamp (2019), who uses $BIC = -2\ell(\mathbf{X}, \mathbf{M}) + |C| \times p \times \log(n)$. Our criterion differs through the weights based on $\gamma$. Please note that other approaches with different expressions exist. Wang and Allen (2021) use $BIC = -n \times \ell(\mathbf{X}, \mathbf{M}) + |C| \times \log(np)$, and Park et al. (2019) use $BIC = -\ell(\mathbf{X}, \mathbf{M}) + |C| \times p \times \frac{\log(n)}{n}$. By running MIDACC for various values of $\lambda$ and selecting that model yielding the lowest BIC, we select our 'best' model given a data set.

### 3.3.3 Evaluation

To evaluate the performance of MIDACC against that of baseline approaches, we consider various evaluation criteria. As real-life data usually does not come with a 'true' clustering outcome, evaluating the results of clustering algorithms is not as straightforward a task as supervised learning techniques. However, to validate results, two types of criteria exist: internal and external validation criteria (Ahmad and Khan, 2019). Internal validation criteria are based solely on the clustering itself, while external validation makes use of previous knowledge on the data.

In our research, we apply our methods to simulated clusters and data sets for which a classification is known a priori. Therefore, we can apply external validation criteria and test the outcomes of our methods against the 'ground truth'. In order to do so, we evaluate two metrics: clustering accuracy and adjusted Rand index (Hubert and Arabie, 1985).

The clustering accuracy $Acc$ is given by

$$Acc = \frac{\sum_{i=1}^{n} I(c_i = y_i)}{n}, \tag{23}$$

where $c_i$ is the predicted class (the obtained cluster) of observation $i$, $y_i$ is its true class, and $I(c_i = y_i) = 1$ if the predicted class equals the true class (with value 0 otherwise). Note that there is no predefined agreement between the found clusters and true classes. Instead, we artificially match classes by choosing the 'best' matches between prediction and truth.

The adjusted Rand index (ARI), which is the corrected-for-chance version of the Rand index (Rand, 1971), compares two partitions of observations $i = 1, \ldots, n$ in terms of a contingency analysis. In our case, the predicted partition is $C$ with $|C|$ classes $C_1, \ldots, C_{|C|}$, while the true partition is given by $Y$, with $|Y|$ classes $Y_1, \ldots, Y_{|Y|}$. We denote the number of elements in a specific class $C_v$ or $Y_z$ by $|C_v|$ and $|Y_z|$, for $v = 1, \ldots, |C|$ and $z = 1, \ldots, |Y|$, respectively. Also, let $n_{vz} = |C_v \cap Y_z|$. Then the ARI is given by

$$ARI = \frac{\sum_{v,z} \binom{n_{v,z}}{2} - \left(\sum_v \binom{|C_v|}{2} \sum_z \binom{|Y_z|}{2}\right) / \binom{n}{2}}{\frac{1}{2}\left(\sum_v \binom{|C_v|}{2} + \sum_z \binom{|Y_z|}{2}\right) - \left(\sum_v \binom{|C_v|}{2} \sum_z \binom{|Y_z|}{2}\right) / \binom{n}{2}}. \tag{24}$$

In case $ARI = 1$, the predicted classes are exactly in line with the true classes. In other words, the clustering is 'perfect' with respect to the classification. In case $ARI = 0$, the clustering performs as well as one would expect it to do purely based on chance. Note that, as opposed to the regular Rand index, the ARI can be lower than 0. This case corresponds with a clustering that performs worse than one would expect it to do purely based on chance.

### 3.4 Baseline Methods

As in other clustering papers, we compare the workings of our method with that of prominent and popular other approaches. In this research, we focus on two commonly used mixed data clustering methods and a third method that has recently gained popularity. The first two methods include K-prototypes (Huang, 1997) and HAC (Murtagh and Contreras, 2012) with the Gower coefficient as distance metric (Gower, 1971). For HAC, we apply Ward linkage (Ward, 1963), which, from observing results of various linkage methods, we find yields the best results. The third method is a model-based clustering method called KAMILA (Foss et al., 2016). This more recent approach for clustering mixed data has gained popularity and is interesting to compare our results with.

Firstly, K-prototypes (Huang, 1997) combines distance metrics for numerical and categorical data in the objective function, where a parameter, which we set based on the heuristics of Szepannek (2018), is used to control the relative weight of categorical attributes. In this objective function, the number of clusters is predefined. Secondly, in contrast, HAC (Murtagh and Contreras, 2012) builds up the cluster structure in tree-wise fashion. That is, in every HAC-step, two clusters are merged. Cutting the tree of clusters at a certain point gives us a partition of a certain number of clusters. Lastly, KAMILA (Foss et al., 2016) combines a kernel-based loss for numerical and a multinomial loss for categorical attributes in a mixture model with a pre-specified number of clusters (or mixture components). Using the idea of latent variables governing the cluster assignment of observations, KAMILA finds a partition of clusters using an EM algorithm. Through the combination of loss functions, Foss et al. (2016) aim to equitably balance the influence of numerical and categorical features on the clustering outcome.

### 3.5 W-MIDACC

Although MIDACC allows for combining numerical and categorical features, different features may prove to be more informative than others. To capture this difference, Huang et al. (2005) apply variable-specific weights to the K-means formulation, which they find can drastically improve clustering results. This concept has, however, not yet been applied to convex clustering. Therefore, we also provide a variable-specific weighted version, called W-MIDACC, of mixed data convex clustering. This version contributes to MIDACC's flexibility by allowing the clustering outcome to be constructed based on specific feature weights. We note that we require the user to input weights prior to running W-MIDACC. Adaptive optimization of the weights given the data set at hand is outside the scope of this paper. We provide details on W-MIDACC as well as some illustrations of its potential in Appendix I.

### 3.6 Software

This research makes use of the R software, version 4.0.5 (R Core Team, 2021). Our code is available on GitHub. We make use of the packages *clustMixType* for K-prototypes (Szepannek, 2018), *kamila* for KAMILA (Foss and Markatou, 2016), and the *hclust* and *cutree* functions for HAC (R Core Team, 2021). To generate non-spherical data, we apply the *halfmoons* data generating function (van den Berg et al., 2018).

# 4 Data

We apply the proposed methods using a simulation study and an analysis of a real-life data set. In this section, we explain the set-up of both applications.

## 4.1 Simulation Study

As previously discussed, besides finding a global optimum to the clustering objective, MIDACC's main assets should be the abilities to recover both spherical and non-spherical cluster shapes, handle mixed data by capturing and combining the information in both numerical and categorical data, and retrieve the true number of clusters present in the data. To this end, we divide our numerical experiments in three parts that cover these assets.

Furthermore, as the manner in which we combine numerical and categorical data depends on the choice of $\gamma$, we perform a sensitivity analysis on this control parameter. Unless mentioned otherwise, we generate $r = 200$ replications of data sets containing $n = 100$ observations. Furthermore, as previously discussed, we have $\delta = 0.5$, $\phi = 2$, $K = 10$, and $\gamma$ loss-based.

Note that, except from the part covering the retrieval of the true number of clusters, we assume that the number of clusters is known a priori. We do so as to ensure fair comparison between MIDACC and the baseline methods, the latter of which require selecting the number of clusters prior to the clustering. As previously discussed, we search across values of $\lambda$ to retrieve MIDACC models yielding the desired number of clusters.

### 4.1.1 Recovering Cluster Shapes

In this case, we investigate the ability to recover both spherical and non-spherical cluster shapes in mixed data. In doing so, we compare the performance of MIDACC with that of the baseline methods in two experiments; one with spherical and one with non-spherical clusters.

**Spherical clusters.** We generate numerical data from multivariate normal distributions and implement multinomial distributions for each categorical feature. More specifically, we create $|C| = 2$ clusters with $n_1 = n_2 = 50$ observations each and $p^{num} = p^{cat} = 4$. For the numerical data, we apply two multivariate normal distributions with covariance $2 \cdot \mathbf{I}_4$ and location parameters $\mathbf{1}_4$ and $-\mathbf{1}_4$ for the two clusters. We generate categorical data, where $K_j = 4$, $\forall j \in \mathcal{C}$, from two feature-specific multinomial distributions for the two clusters. Both clusters have a different majority class per feature, which we assign probability 0.75. We set the probabilities of the other three classes to

be equal and sum up to 0.25. Since clear cluster structure is present in both data types, we expect all clustering methods to be able to detect this information and partition the data appropriately.

**Non-spherical clusters.** Again, we have $|C| = 2$ clusters with $n_1 = n_2 = 50$ observations each, where $p^{num} = p^{cat} = 4$. In this case, however, we generate non-spherical shapes for numerical attributes. We do so by creating two pairwise *interlocking halfmoons*, with 30% noise, within the first and last two numerical features. As for the categorical data, we use $K_j = 4$, $\forall j \in \mathcal{C}$, and implement multinomial distributions with a different majority class per cluster. We assign probability 0.5 to this class and equal probabilities, summing up to 0.5, to the other three classes. Thus, each categorical feature contains 50% noise. This set-up tests the ability to detect non-spherical signal in the data, while there is some structure in both data types. We expect spherical baseline methods, such as KAMILA and K-prototypes, to obtain reasonable results, using the signal in categorical data and spherically clustering the numerical data. However, we expect MIDACC to be able to outperform these methods through its ability to capture the non-spherical part of the data.

### 4.1.2 Capturing Information in Mixed Data Sets

Next, we investigate MIDACC's ability to detect information in mixed data sets where different numerical and categorical structures occur. More specifically, we consider three experiments; i) both data types consist of features which exhibit cluster structure (signal) and features that do not (noise), ii) the categorical data has signal, while the numerical data contains noise, and iii) the numerical data contains signal, but the categorical data is noisy.

**Signal and noise in both data types.** We create data from $|C| = 2$ clusters with $n_1 = n_2 = 50$ observations each, where $p^{num} = p^{cat} = 8$. For both data types, half of the features contain cluster signal, while the other half consists of noise features. For numerical data, we generate informative features from two multivariate normal distributions with covariance $\mathbf{I}_4$ and location parameters $\mathbf{1}_4$ and $-\mathbf{1}_4$ for the two clusters. We generate noisy features from a single multivariate normal distribution with $\mathbf{I}_4$ covariance and location $\mathbf{0}_4$. We generate categorical data, where $K_j = 4$, $\forall j \in \mathcal{C}$, in similar fashion. The informative features follow multinomial distributions with majority class probability 0.7 and probability 0.1 for the other three classes, while the noise features follow multinomial distributions with equal probabilities, i.e., 0.25, assigned to all classes. This set-up tests the ability of detecting signal in mixed data when such signal is not present in all features.

**Signal in categorical, noise in numerical data.** We create data from $|C| = 2$ clusters with $n_1 = n_2 = 50$ observations each and $p^{num} = p^{cat} = 4$, where the categorical data exhibits

cluster structure but the numerical data does not. That is, we generate numerical data from a single multivariate normal distribution with location $\mathbf{1}_4$ and covariance $\mathbf{I}_4$. For the categorical data, we have $K_j = 4$, $\forall j \in \mathcal{C}$, and use multinomial distributions with a different majority class per cluster. We assign probability 0.75 to this class and equal probabilities summing up to 0.25 for the other three classes. A clustering algorithm that can capture the structure of mixed data appropriately should be able to detect signal and distinguish it from noise, which makes this set-up interesting.

**Signal in numerical, noise in categorical data.** Again, we have $|C| = 2$ clusters with $n_1 = n_2 = 50$ observations each and $p^{num} = p^{cat} = 4$. In this case, the numerical data contains signal, whereas the categorical data does not. That is, we generate numerical data from two multivariate normal distributions with $\mathbf{I}_4$ covariance and location parameters $\mathbf{1}_4$ and $-\mathbf{1}_4$ for the two clusters. For the categorical data, where $K_j = 4$, $\forall j \in \mathcal{C}$, all classes have probability 0.25 across all features. Again, this set-up tests clustering algorithms' ability to capture information in mixed data.

### 4.1.3 Retrieving the Number of Clusters

The third important property that we test is MIDACC's ability to recover the true number of clusters. To do so, we apply the BIC criterion (22) to select the best fitting number of clusters given a data set. As before, we use $n = 100$ observations, while we now perform $r = 100$ replications. We perform two experiments; one where the true number of clusters $|C| = 2$, and one where $|C| = 4$. For both experiments, we consider the number of clusters to range between 1 and 10.

**2 clusters.** We have two clusters with $n_1 = n_2 = 50$ observations each, where $p^{num} = p^{cat} = 4$. Furthermore, we generate numerical data from two multivariate normal distributions with $\mathbf{I}_4$ covariance and location $\mathbf{1}_4$ and $-\mathbf{1}_4$ for the two clusters, respectively. We set $K_j = 4$, $\forall j \in \mathcal{C}$ and implement multinomial distributions with majority class probability 0.7 and probability 0.1 assigned to the other classes for all features. The majority class differs across the two clusters.

**4 clusters.** Here, we have four clusters, now with $n_1 = n_2 = n_3 = n_4 = 25$ observations each. Furthermore, we set $p^{num} = p^{cat} = 4$. We generate numerical data from four multivariate normal distributions with identity covariance and location parameters $\mathbf{3}_4$, $\mathbf{1}_4$, $-\mathbf{1}_4$, and $-\mathbf{3}_4$. For categorical data, we use the same settings as in the previous experiment, such that we obtain four multinomial distributions for each feature, each having a different majority class with probability 0.7 and three other classes with probability 0.1. Although naturally subject to variability, being able to select the true number of clusters would be a strong asset of MIDACC.

### 4.1.4 Sensitivity Analysis on $\gamma$

Besides assessing three of the main potential assets of MIDACC, we investigate the influence of the parameter $\gamma$ on MIDACC's clustering outcome. Recall that this parameter controls the relative importance of the two data types in the clustering task. Different settings of this parameter may, for the very same data set, yield completely different outcomes. Therefore, it is relevant to investigate the behavior of this parameter. In this case, we use $r = 50$ replications, as evaluating MIDACC for a wide range of values for $\gamma$ is computationally costly.

We start off by plotting out the performance of MIDACC for a grid of values of $\gamma$. More specifically, we let $\gamma$ range between 0 and 1 with stepsize 0.1. We plot the corresponding behavior using the same experiments we applied to assess MIDACC's ability to capture the information in mixed data sets. Thus, we consider three data sets; i) signal and noise in both data types, ii) signal in categorical, noise in numerical attributes, and iii) signal in numerical, noise in categorical attributes.

After plotting the behavior of MIDACC over $\gamma$, we assess its performance using the various specifications of $\gamma$ from Section 3.3.2. We do so using the same data sets as those used for evaluating MIDACC's assets of recovering cluster shapes and capturing information in mixed data sets.

## 4.2 Real-Life Data

In order to demonstrate the applicability of our methods in real-life, we apply our methods to a mixed data set retrieved from the UCI Machine Learning Repository (Blake and Merz, 1998). More specifically, we use the Cleveland Heart Disease data set. This is a frequently used data set in mixed data clustering research, see e.g., Huang et al. (2005) and Modha and Spangler (2003).

The Cleveland Heart Disease data set contains 303 observations, of which 6 observations have missing values in some of the attributes. We remove these observations and thus end up with $n = 297$ observations. The data set contains 13 features, of which $p^{num} = 5$ features are numerical and $p^{cat} = 8$ features are categorical, with varying numbers of classes $K_j$, $j \in \mathcal{C}$. Furthermore, a 'true' classification exists that defines whether or not an observation belongs to a person with a heart disease. We use this classification to evaluate MIDACC and the baseline methods. That is, we search for a value of $\lambda$ that yields $|C| = 2$ clusters to ensure fair comparison with the baseline methods, which require preselecting $|C|$. As in the simulation study, we set $\delta = 0.5$, $\phi = 2$, and choose $\gamma$ loss-based. Furthermore, we set $K = 30$, which, similar to the simulation study, implies that we apply around 90% sparsity in pairwise weights.

# 5 Results

In this section, we present the results of our methods by discussing the simulation study and real-life data application that we previously introduced.

## 5.1 Simulation Study

We start off by assessing MIDACC's recovery of both spherical and non-spherical cluster shapes, after which we test its capability to capture information in mixed data sets. In doing so, we compare MIDACC with the previously discussed baseline methods. Then, we investigate MIDACC's ability to retrieve the true number of clusters and, last, analyze its sensitivity over $\gamma$. As previously noted, we perform $r = 200$ replications with $n = 100$ observations, unless mentioned otherwise. For KAMILA and K-prototypes, baseline methods dependent on initialization, we use 100 random starts per replication and select the partition yielding the lowest objective value.

### 5.1.1 Recovering Cluster Shapes

Here, we assess the performance of MIDACC and the baseline methods in the cases of spherical and non-spherical clusters in mixed data sets using the two experiments discussed in Section 4.1.1. In Table 1, we provide results for the spherical case, where we present the mean and standard deviation of the accuracy and ARI for all methods. We compute the standard deviation across replications, such that this measure provides an indication of the stability of the applied methods but does not allow for comparing whether two methods have significantly different mean performances. Also, we note that we use multiple random starts per replication for KAMILA and K-prototypes, such that the standard deviation shows the stability of the 'best' partitions and thus does not indicate the sensitivity of these methods to different initializations.

*Table 1: Mean and standard deviation of the performance of MIDACC and the baseline methods in the case of spherical cluster shapes.*

| | *Acc* | | *ARI* | |
|---|---|---|---|---|
| | Mean | St. Dev. | Mean | St. Dev. |
| MIDACC | 0.988 | 0.011 | 0.951 | 0.044 |
| KAMILA | 0.986 | 0.013 | 0.947 | 0.051 |
| K-prototypes | 0.988 | 0.012 | 0.951 | 0.045 |
| HAC | 0.965 | 0.022 | 0.867 | 0.081 |

As expected, we find that both MIDACC and the spherical baselines, i.e., KAMILA and K-prototypes, are well able to achieve near perfect recovery of spherical clusters. Namely, we find that these three methods all reach an average accuracy and ARI of around 0.990 and 0.950, respectively. In contrast, HAC performs slightly worse, obtaining an accuracy and ARI of 0.965 and 0.867, respectively. This also shows in the standard deviation of HAC's accuracy, which, at a value of 0.022, is around twice as high as that of MIDACC. Although K-prototypes exhibits similar standard deviations to MIDACC, KAMILA seems to be slightly less stable.

Turning to the non-spherical case, where we generate numerical data from interlocking halfmoons, we obtain the results as given in Table 2. From this table, we observe that MIDACC outperforms all baseline methods. More specifically, MIDACC's performance is near perfect with mean accuracies and ARIs of 0.983 and 0.932, respectively. KAMILA and K-prototypes perform slightly worse, reaching accuracies and ARIs of around 0.970 and 0.890, respectively. The HAC baseline performs poorest, achieving a mean accuracy and ARI of 0.886 and 0.600, respectively.

*Table 2: Mean and standard deviation of the performance of MIDACC and the baseline methods in the case of non-spherical cluster shapes.*

| | *Acc* | | *ARI* | |
|---|---|---|---|---|
| | Mean | St. Dev. | Mean | St. Dev. |
| MIDACC | 0.983 | 0.014 | 0.932 | 0.055 |
| KAMILA | 0.971 | 0.018 | 0.887 | 0.068 |
| K-prototypes | 0.974 | 0.015 | 0.899 | 0.058 |
| HAC | 0.886 | 0.046 | 0.600 | 0.141 |

Compared to the spherical case, we find that, as expected, all baseline methods have decreasing performance, as their ability to recover non-spherical cluster shapes is limited. In contrast, MIDACC is able to capture complex cluster shapes in the data. Furthermore, MIDACC and K-prototypes seem to be most stable, while KAMILA and especially HAC possess larger standard deviations.

### 5.1.2 Capturing Information in Mixed Data Sets

In this section, we investigate the manner in which MIDACC and the baseline methods detect signal in mixed data sets and are able to distinguish such signal from noise. We do so by conducting the three experiments presented in Section 4.1.2.

First, Table 3 contains results for the experiment where we have $p^{num} = p^{cat} = 8$ features per data type, half of which contain signal and half of which do not. We find that MIDACC, KAMILA, and K-

prototypes all adequately distinguish signal from noise within data types, obtaining mean accuracies and ARIs of around 0.990 and 0.960, respectively. Although the differences are small, KAMILA seems to perform best, which may be due to the fact that the generated data sets are symmetric; both data types contain an equal number of informative and non-informative features. This is in accordance with KAMILA's aim of being able to equitably balance numerical and categorical data (Foss et al., 2016). In turn, MIDACC has a slightly higher average accuracy and ARI than K-prototypes. Again, HAC performs worst, reaching an average accuracy and ARI of 0.944 and 0.791, respectively. This indicates that, as opposed to MIDACC and the spherical baselines, HAC may not be well-suited for balancing signal and noise in mixed data sets. In terms of stability, we find that MIDACC and KAMILA have similarly low standard deviations, while the variability in K-prototypes and, especially, HAC is larger.

*Table 3: Mean and standard deviation of the performance of MIDACC and the baseline methods in the case of 50%-50% signal-noise features in both data types.*

| | *Acc* | | *ARI* | |
|---|---|---|---|---|
| | Mean | St. Dev. | Mean | St. Dev. |
| MIDACC | 0.989 | 0.010 | 0.956 | 0.038 |
| KAMILA | 0.992 | 0.009 | 0.968 | 0.035 |
| K-prototypes | 0.987 | 0.012 | 0.950 | 0.046 |
| HAC | 0.944 | 0.026 | 0.791 | 0.092 |

Second, we conduct an experiment in which the categorical data contains signal but the numerical data does not. Whereas the previous experiment tests the capacity of clustering algorithms to distinguish signal from noise *within* data types, this experiment tests the ability to do so *across* data types. We report results for MIDACC and the three baseline methods in Table 4.

*Table 4: Mean and standard deviation of the performance of MIDACC and the baseline methods in the case of signal in categorical and noise in numerical data.*

| | *Acc* | | *ARI* | |
|---|---|---|---|---|
| | Mean | St. Dev. | Mean | St. Dev. |
| MIDACC | 0.961 | 0.022 | 0.849 | 0.079 |
| KAMILA | 0.624 | 0.141 | 0.158 | 0.272 |
| K-prototypes | 0.953 | 0.024 | 0.820 | 0.088 |
| HAC | 0.942 | 0.027 | 0.781 | 0.097 |

From Table 4, we find that MIDACC is the best performing method, reaching an average accuracy and ARI of 0.961 and 0.849, respectively. K-prototypes and HAC perform slightly worse with average accuracies of 0.953 and 0.942 and ARIs of 0.820 and 0.781, respectively. However, most striking is the poor performance of KAMILA, which achieves a mean accuracy and ARI of only 0.624 and 0.158, respectively. This indicates that, while MIDACC is the top method in detecting signal in categorical and noise in numerical data, KAMILA is not suitable for such data sets. This may be due to KAMILA's focus on equitably balancing numerical and categorical data, which is not appropriate in this setting. Considering standard deviations, we find that MIDACC is the most stable method, while KAMILA has the highest variability in performance across replications.

Third, we consider the case in which the numerical data exhibits signal but the categorical data does not. In this case, we obtain performances as given in Table 5. From this table, we find that KAMILA is the top performer with an average accuracy and ARI of 0.974 and 0.899, respectively. Thus, apparently, KAMILA puts relatively large weights on numerical data in this case, which may also be the reason why its performance in the previous experiment is poor. Still, MIDACC and K-prototypes achieve strong results, reaching accuracies of 0.957 and 0.965 and ARIs of 0.834 and 0.864, respectively. In contrast, HAC performs poorly, reaching similarly low values as KAMILA did in the previous experiment. Considering standard deviations, we observe a similar pattern, where KAMILA is most stable and HAC is subject to the highest variability.

*Table 5: Mean and standard deviation of the performance of MIDACC and the baseline methods in the case of signal in numerical and noise in categorical data.*

| | *Acc* | | *ARI* | |
|---|---|---|---|---|
| | Mean | St. Dev. | Mean | St. Dev. |
| MIDACC | 0.957 | 0.022 | 0.834 | 0.080 |
| KAMILA | 0.974 | 0.017 | 0.899 | 0.063 |
| K-prototypes | 0.965 | 0.019 | 0.864 | 0.069 |
| HAC | 0.651 | 0.092 | 0.113 | 0.120 |

Considering all three experiments in this section, we may state that HAC and KAMILA are unable to appropriately capture information in mixed data sets in the presence of noise. Namely, KAMILA breaks down when the numerical data is noisy, while HAC fails in the case of noisy categorical features. MIDACC and K-prototypes seem to be the most appropriate methods, achieving strong performances in all three experiments.

### 5.1.3 Retrieving the Number of Clusters

In this section, we investigate MIDACC's ability to retrieve the true number of clusters in mixed data sets. Whereas baseline methods require preselecting the number of clusters, MIDACC can control for the number of clusters through $\lambda$. Selecting the most suitable value of $\lambda$ allows us to retrieve the 'optimal' number of clusters given the data set at hand. As discussed in Section 3.3.2, we do so using the BIC criterion (22). We use $r = 100$ replications and set $n = 100$.

Computing the BIC for values of $\lambda$ that yield different numbers of clusters and selecting the number of clusters corresponding with the lowest BIC, we retrieve MIDACC's 'optimal' number of clusters given a data set. In Table 6, we compare this optimal number of retrieved clusters with the true number of clusters $|C|$ when $|C|$ is either 2 or 4. The table entries corresponding with correct retrievals of the true number of clusters are given in bold.

*Table 6: Frequencies of MIDACC's retrieved optimal number of clusters for $r = 100$ replications of data sets where the true number of clusters $|C|$ is either 2 or 4.*

| | Retrieved optimal number of clusters | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $\|C\| = 2$ | 0 | **72** | 27 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\|C\| = 4$ | 0 | 23 | 22 | **39** | 15 | 1 | 0 | 0 | 0 | 0 |

From Table 6, it is apparent that MIDACC tends to select the true number of clusters in both cases. When $|C| = 2$, MIDACC retrieves the correct number of clusters in 72 out of 100 cases, while it does so in 39 out of 100 times when $|C| = 4$. For both cases, the true number of clusters is the number that is retrieved most often, which is a promising result that indicates MIDACC's ability to retrieve the true number of clusters in a data set. This ability is highly valuable in practice, where the true number of clusters is usually unknown. Furthermore, we observe that MIDACC always selects the number of clusters to be higher than 1, where the variation tends to be larger when the true number of clusters is higher. This is a common property of the BIC (Schwarz et al., 1978).

### 5.1.4 Sensitivity Analysis on $\gamma$

Now that we have established three of MIDACC's main assets, we turn to investigating the sensitivity of our mixed data convex clustering framework to the control parameter $\gamma$. Recall that this parameter controls the contribution of numerical and categorical data on the clustering outcome, where a larger

value of $\gamma$ corresponds with a relatively larger influence of categorical attributes and $0 \leq \gamma \leq 1$.

We start off by investigating MIDACC's performance over a grid of values for $\gamma$. More specifically, we let $\gamma$ range between 0 and 1 with stepsize 0.1 and run $r = 50$ replications of the experiments from Section 4.1.2. In Figure 1, we plot the ARI of the corresponding experiments.

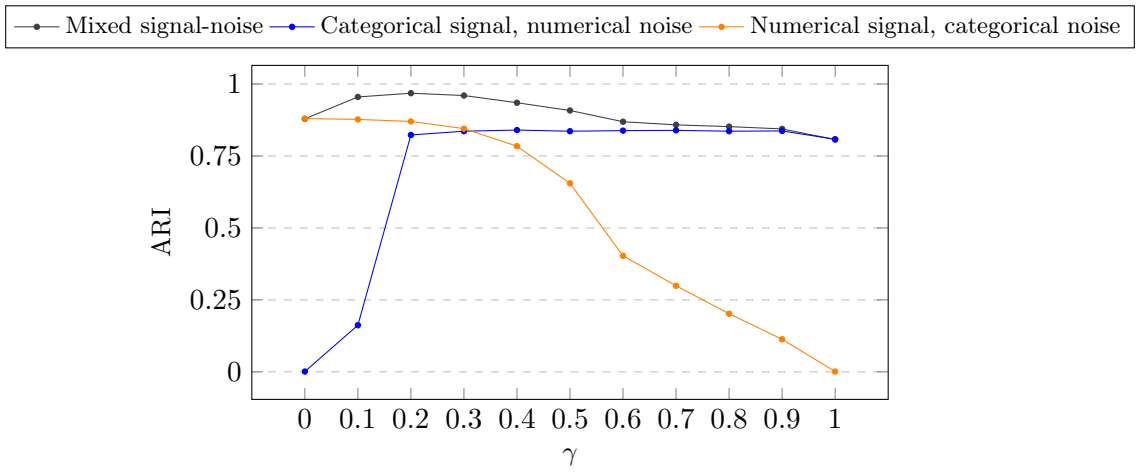


*Figure 1: ARI of MIDACC plotted over a grid of values for $\gamma$, ranging between 0 and 1 with stepsize 0.1, for the three experiments from Section 4.1.2. 'Mixed signal-noise' refers to the experiment with 50%-50% signal-noise features in both data types, 'Categorical signal, numerical noise' refers to the experiment with signal in categorical and noise in numerical data, and 'Numerical signal, categorical noise' refers to the experiment with signal in numerical and noise in categorical data.*

Let us first consider the case in which the categorical data exhibits signal but the numerical data does not ('Categorical signal, numerical noise'). In this case, we expect a larger value of $\gamma$ to result in better performance, which is in line with what we observe; the larger $\gamma$, the higher the ARI tends to be. Also, the ARI is 0 when $\gamma = 0$. Namely, when we do not consider categorical data, we are clustering purely based on noisy data, which, inherently, corresponds with clustering by chance.

In case the numerical data has signal but the categorical data does not ('Numerical signal, categorical noise'), we observe, as expected, the opposite pattern. For a smaller value of $\gamma$, we put a larger weight on the (informative) numerical data, which leads to MIDACC obtaining better performance. Again, at the boundary of $\gamma$ which corresponds with ignoring the signal (in this case, this occurs when $\gamma = 1$), we obtain an ARI of 0.

In the third case, we have signal and noise in both data types ('Mixed signal-noise'). In this case, purely focusing on one of the data types yields reasonable performance, but combining the data allows us to almost perfectly recover the true cluster structure. From Figure 1, we find that the ARI increases away from the boundaries $\gamma = 0$ and $\gamma = 1$, reaching a maximum ARI around $\gamma = 0.2$. Apparently, at this point, the combination between numerical and categorical data is optimal.

Note that the boundaries in the third case coincide with the optimal outcomes of the first two cases. Thus, when we focus on a single data type that contains both signal and noise, we can achieve the same performance as we would in case this type solely bears signal. This even further establishes MIDACC's ability to distinguish signal from noise, both between and within data types.

Besides investigating MIDACC's performance over a grid of values for $\gamma$, we compare our various proposed approaches to estimating $\gamma$ as discussed in Section 3.3.2. In this section, we present results of four approaches; i) $\gamma = 0.5$, ii) $\gamma$ is loss-based ($\gamma$ - loss), iii) $\gamma$ is distance-based ($\gamma$ - dist), and iv) $\gamma$ is based on model selection ($\gamma$ - mod). Note that we proposed other choices for $\gamma$ as well, but we decide to omit these in this section due to their weaker performance. For an extensive overview of all proposed methods, we refer the reader to Appendix G. Besides evaluating MIDACC for various choices of $\gamma$, we include the three baseline methods.

In Table 7, we provide accuracies and ARIs for the above methods using $r = 50$ replications of data sets simulated using the five experiments in Sections 4.1.1 and 4.1.2. Experiments I and II correspond with the spherical and non-spherical cases, respectively, while experiments III, IV, and V refer to the experiments with signal-noise in both data types, signal in categorical but not in numerical data, and signal in numerical but not in categorical data, respectively.

*Table 7: Mean performance of MIDACC for various choices of $\gamma$. Experiments I and II refer to the spherical and non-spherical experiments from Section 4.1.1, respectively. Experiments III, IV, and V refer to the experiments from Section 4.1.2 with noise and signal in both data types, signal in categorical but noise in numerical data, and signal in numerical but noise in categorical data, respectively.*

| | Experiment I | | Experiment II | | Experiment III | | Experiment IV | | Experiment V | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* |
| $\gamma$ - 0.5 | 0.976 | 0.908 | 0.943 | 0.785 | 0.976 | 0.908 | 0.957 | 0.836 | 0.899 | 0.655 |
| $\gamma$ - loss | 0.986 | 0.944 | 0.981 | 0.925 | 0.991 | 0.964 | 0.956 | 0.831 | 0.962 | 0.856 |
| $\gamma$ - dist | 0.984 | 0.938 | 0.973 | 0.895 | 0.991 | 0.964 | 0.955 | 0.826 | 0.961 | 0.850 |
| $\gamma$ - mod | 0.986 | 0.944 | 0.984 | 0.937 | 0.991 | 0.963 | 0.956 | 0.830 | 0.967 | 0.873 |
| KAMILA | 0.986 | 0.944 | 0.967 | 0.872 | 0.993 | 0.974 | 0.652 | 0.200 | 0.971 | 0.887 |
| K-prototypes | 0.986 | 0.946 | 0.973 | 0.894 | 0.990 | 0.961 | 0.952 | 0.817 | 0.962 | 0.852 |
| HAC | 0.960 | 0.846 | 0.886 | 0.600 | 0.944 | 0.789 | 0.940 | 0.775 | 0.649 | 0.120 |

From Table 7, we observe that the model selection method upper bounds the other three choices of $\gamma$ for all five experiments. However, the differences between this choice and the loss- and distance-based methods are relatively small. Since the latter two methods, in contrast to the model selection approach, require only a single MIDACC run per data set, we prefer to use these methods for now. However, the model selection method seems promising for future work. Please note that $\gamma = 0.5$ does

not perform as well as the other methods, such that we advise against using this setting. Although the loss- and distance-based methods result in strong and similar performances for all experiments, the loss-based method clearly outperforms the distance-based method in the second experiment. Thus, we prefer the loss-based method over the distance-based method.

### 5.2 Real-Life Data

In this section, we present the results of MIDACC and the baseline methods for the Cleveland Heart Disease data set. As previously discussed, a true classification of this data set into two clusters is known. To ensure fair comparison of our method with the baseline methods that require specifying $|C|$ prior to running the algorithm, we run MIDACC with $\lambda$ such that we retrieve $|C| = 2$ clusters. Furthermore, just as in the simulation study, we use the loss-based $\gamma$ measure.

Table 8 presents the accuracy and ARI of MIDACC and the three baseline methods for the Heart data. It is apparent that MIDACC outperforms all three baseline methods, reaching an accuracy of 0.815 and an ARI of 0.394. Out of the three baseline methods, KAMILA performs best with an accuracy of 0.805 and an ARI of 0.369, while K-prototypes performs worst, reaching an accuracy and ARI of 0.785 and 0.322, respectively. Although MIDACC clearly outperforms the baseline methods and reaches reasonable accuracy, one may note that the ARI seems to be a bit low just below 0.4. However, this result is highly competitive with the strongest results across literature. To the best of our knowledge, the highest reported ARI is 0.41 (van de Velden et al., 2019), while most reported ARIs reach around or below 0.35. Therefore, we consider MIDACC's performance highly promising.

*Table 8: Performance of MIDACC and the baseline methods on the Heart data set.*

| | *Acc* | *ARI* |
|---|---|---|
| MIDACC | 0.815 | 0.394 |
| KAMILA | 0.805 | 0.369 |
| K-prototypes | 0.785 | 0.322 |
| HAC | 0.798 | 0.353 |

Within MIDACC, different values of $\gamma$ can result in completely different clustering performances. Although we currently implement the loss-based heuristic, which seems to generally achieve satisfactory results, it is interesting to evaluate MIDACC's performance over a grid of values for $\gamma$. We set $\gamma$ to range between 0 and 1 with stepsize 0.05 and evaluate MIDACC for values of $\lambda$ such that we retrieve $|C| = 2$ clusters. In Figure 2, we depict the corresponding results.

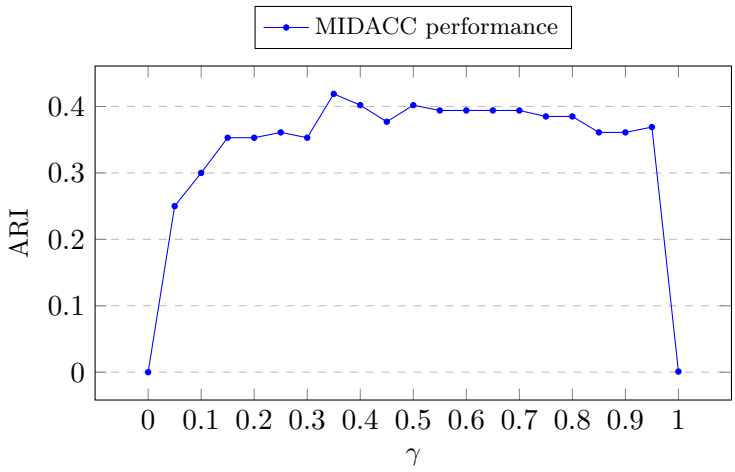


*Figure 2: ARI of MIDACC plotted over a grid of values for $\gamma$, ranging between 0 and 1 with stepsize 0.05, evaluated on the Heart data set.*

From Figure 2, we observe that the performance of MIDACC varies strongly with $\gamma$. MIDACC's performance is poorest at the boundaries $\gamma = 0$ and $\gamma = 1$, where the clustering is purely based on a single data type. More specifically, we find that focusing on a single data type yields an outcome that can be considered 'clustering by chance'. Moving away from the boundaries and thus combining the two data types, MIDACC's performance improves. We reach an optimum when $\gamma = 0.35$, which corresponds with an ARI of 0.419. To the best of our knowledge, this is higher than any ARI reported in literature for this data set. This once again stresses the high benefit of MIDACC.

## 6 Conclusions and Future Work

In this research, we introduced the mixed data convex clustering, or MIDACC, framework. MIDACC adds to the current literature by providing a mixed data clustering method that does not suffer from instabilities. In MIDACC, we combine likelihood-based loss functions for numerical and categorical data, weighted by parameters controlling the importance of both data types on the clustering outcome, with a penalty term that fuses centroids and thus allows for clustering of observations. We implement MIDACC by developing an efficient subgradient descent algorithm.

To evaluate the workings of MIDACC, we apply our method using numerical experiments. First, we find that, as opposed to all baseline methods, MIDACC achieves near perfect recovery of both spherical and non-spherical clusters. Second, we show that MIDACC, as opposed to the HAC and KAMILA baselines, has the capacity to capture information from mixed data whilst effectively distinguishing signal from noise within and across data types. Third, through model selection, we

illustrate MIDACC's ability to recover the true number of clusters present in mixed data sets. Based on a real-life data example, we show that MIDACC outperforms all baseline methods, which further highlights the practical relevance of our method.

There are four main contributions of MIDACC. First, due to its convex clustering objective, MIDACC guarantees convergence to a global optimum. This enhances its applicability and performance compared to other mixed data clustering algorithms. Second, MIDACC is not limited to retrieving spherical clusters but can recover rather complex shapes, which makes our method more flexible than other, spherical, methods. Third, MIDACC can retrieve the true number of clusters in mixed data sets. As the number of clusters is usually unknown prior to clustering, this is a valuable asset in practice. Fourth, MIDACC is able to capture the information present in mixed data sets and appropriately balance numerical and categorical data. Through simulations and a real-life application, these assets prove significantly beneficial compared to state-of-the-art baseline methods.

In the future, we recommend four primary areas for future work. First, we note that, although MIDACC-SGD is relatively efficient, we could focus more on the computational aspect of MIDACC. In Appendix D, we provide an ADMM-type algorithm which, in future work, could be further developed in order to enhance scalability. Another option would be to apply a diagonal majorization technique similar to that of Touw et al. (2020).

Second, we currently compute the control parameter $\gamma$ prior to running the algorithm. In the future, we may focus on strictly optimizing this parameter, either simultaneously or sequentially with respect to the centroids. Such optimization could benefit strongly from the previously described computational improvements and could yield valuable results, the potential of which we have shown for the Heart data set. In similar fashion, we could optimize variable-specific weights in W-MIDACC.

Third, we currently remove entries with missing data from our analysis. However, such data may contain valuable information. Therefore, we could extend MIDACC in order to capture information from observations with missing data. In order to do so, we could consider either using imputation techniques or a model-based approach that directly handles missing values.

Fourth, we note that MIDACC and, to the best of our knowledge, any other existing convex clustering method, assumes conditional independence across attributes, which, in practice, usually does not hold (Hunt and Jorgensen, 2011). Therefore, we could design a convex clustering method that incorporates dependencies between attributes. For categorical data, we could construct new variables with categories representing the 'co-occurrence' of classes, while, for numerical data, we could capture dependencies between variables by relieving the restriction of identity covariance.

## References


Ahmad, A., & Dey, L. (2007). A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, *63*(2), 503–527.

Ahmad, A., & Khan, S. S. (2019). Survey of state-of-the-art mixed data clustering algorithms. *Ieee Access*, *7*, 31883–31902.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, *96*(12), 6745–6750.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data* (pp. 25–71). Springer.

Blake, C. L., & Merz, C. J. (1998). *Uci repository of machine learning databases, 1998.*

Boyd, S., Parikh, N., & Chu, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

Chen, G. K., Chi, E. C., Ranola, J. M. O., & Lange, K. (2015). Convex clustering: An attractive alternative to hierarchical clustering. *PLoS Comput Biol*, *11*(5), e1004228.

Chi, E. C., Gaines, B. R., Sun, W. W., Zhou, H., & Yang, J. (2020). Provable convex co-clustering of tensors. *Journal of Machine Learning Research*, *21*(214), 1–58.

Chi, E. C., & Lange, K. (2015). Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, *24*(4), 994–1013.

Choi, H., & Lee, S. (2019). Convex clustering for binary data. *Advances in Data Analysis and Classification*, *13*(4), 991–1018.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (2004). Clustering large graphs via the singular value decomposition. *Machine learning*, *56*(1), 9–33.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, pp. 226–231).

Foss, A., & Markatou, M. (2018). kamila: clustering mixed-type data in r and hadoop. *Journal of Statistical Software*, *83*(1), 1–44.

Foss, A., Markatou, M., & Ray, B. (2019). Distance metrics and clustering methods for mixed-type data. *International Statistical Review*, *87*(1), 80–109.

Foss, A., Markatou, M., Ray, B., & Heching, A. (2016). A semiparametric method for clustering

mixed data. *Machine Learning*, *105*(3), 419–458.

Fraley, C. (1998). Algorithms for model-based gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, *20*(1), 270–281.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, 857–871.

Hennig, C., & Liao, T. F. (2010). *Comparing latent class and dissimilarity based clustering for mixed type variables with application to social stratification* (Tech. Rep.). Technical report.

Hiruma, A., Yatabe, K., & Oikawa, Y. (2018). Separating stereo audio mixture having no phase difference by convex clustering and disjointness map. In *2018 16th international workshop on acoustic signal enhancement (iwaenc)* (pp. 266–270).

Hocking, T. D., Joulin, A., Bach, F., & Vert, J.-P. (2011). Clusterpath an algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning* (p. 1).

Hoefling, H. (2010). A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, *19*(4), 984–1006.

Hsu, C.-C., Chen, C.-L., & Su, Y.-W. (2007). Hierarchical clustering of mixed data based on distance hierarchy. *Information Sciences*, *177*(20), 4474–4492.

Huang, Z. (1997). Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(pakdd)* (pp. 21–34).

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, *2*(3), 283–304.

Huang, Z., Ng, M. K., Rong, H., & Li, Z. (2005). Automated variable weighting in k-means type clustering. *IEEE transactions on pattern analysis and machine intelligence*, *27*(5), 657–668.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*(1), 193–218.

Hunt, L., & Jorgensen, M. (2011). Clustering mixed data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(4), 352–361.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Kieskamp, T. (2019). Generalised convex clustering: a parametric technique to perform market segmentation.

Lindsten, F., Ohlsson, H., & Ljung, L. (2011). *Just relax and come clustering!: A convexification of k-means clustering*. Linköping University Electronic Press.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*,

*28*(2), 129–137.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(4), 417–473.

Modha, D. S., & Spangler, W. S. (2003). Feature weighting in k-means clustering. *Machine learning*, *52*(3), 217–237.

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 86–97.

Panahi, A., Dubhashi, D., Johansson, F. D., & Bhattacharyya, C. (2017). Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery. In *International conference on machine learning* (pp. 2769–2777).

Pappas, T. N., & Jayant, N. S. (1989). An adaptive clustering algorithm for image segmentation. In *International conference on acoustics, speech, and signal processing,* (pp. 1667–1670).

Park, C., Choi, H., Delcher, C., Wang, Y., & Yoon, Y. J. (2019). Convex clustering analysis for histogram-valued data. *Biometrics*, *75*(2), 603–612.

Pelckmans, K., De Brabanter, J., Suykens, J. A., & De Moor, B. (2005). Convex clustering shrinkage. In *Pascal workshop on statistics and optimization of clustering workshop.*

Punj, G., & Stewart, D. W. (1983). Cluster analysis in marketing research: Review and suggestions for application. *Journal of marketing research*, *20*(2), 134–148.

R Core Team. (2021). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Radchenko, P., & Mukherjee, G. (2014). Convex clustering via $\ell_1$ fusion penalization. *arXiv preprint arXiv:1412.0753*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*(336), 846–850.

Schwarz, G., et al. (1978). Estimating the dimension of a model. *Annals of statistics*, *6*(2), 461–464.

Steinley, D. (2003). Local optima in k-means clustering: what you don't know may hurt you. *Psychological methods*, *8*(3), 294.

Sun, D., Toh, K.-C., & Yuan, Y. (2021). Convex clustering: model, theoretical guarantee and efficient algorithm. *Journal of Machine Learning Research*, *22*(9), 1–32.

Szepannek, G. (2018). clustmixtype: User-friendly clustering of mixed-type data in r. *R J.*, *10*(2), 200.

Tachikawa, T., Yatabe, K., & Oikawa, Y. (2018). 3d sound source localization based on coherence-adjusted monopole dictionary and modified convex clustering. *Applied Acoustics*, *139*, 267–281.

Tan, K. M., & Witten, D. (2015). Statistical properties of convex clustering. *Electronic journal of statistics*, *9*(2), 2324.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(1), 91–108.

Touw, D. J., Groenen, P., & Alfons, A. (2020). Convex clustering: efficiency, efficiency, efficiency.

van den Berg, T., Groenen, P., & Birbil, S. (2018). Majorization methods for solving the convex clustering problem.

van de Velden, M., Iodice D'Enza, A., & Markos, A. (2019). Distance-based clustering of mixed data. *Wiley Interdisciplinary Reviews: Computational Statistics*, *11*(3), e1456.

Wang, M., & Allen, G. I. (2021). Integrative generalized convex clustering optimization and feature selection for mixed multi-view data. *Journal of Machine Learning Research*, *22*(55), 1–73.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*(301), 236–244.

Wei, M., Chow, T. W., & Chan, R. H. (2015). Clustering heterogeneous data with k-means by mutual information-based unsupervised feature transformation. *Entropy*, *17*(3), 1535–1548.

Weylandt, M., Nagorski, J., & Allen, G. I. (2020). Dynamic visualization and fast computation for convex clustering via algorithmic regularization. *Journal of Computational and Graphical Statistics*, *29*(1), 87–96.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), 645–678.

Zhao, Y., & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on information and knowledge management* (pp. 515–524).

# A Loss Functions

In this section, we provide derivations for the loss functions presented in Section 3.2.3.

## A.1 Numerical Loss

Below, we show that, similar to Kieskamp (2019), the numerical loss function $\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) = \frac{1}{2}||\mathbf{x}_i^{num} - \mathbf{m}_i^{num}||_2^2$ originates from a multivariate normal distribution with location parameter $\mathbf{m}_i$ and identity covariance $\mathbf{\Sigma} = \mathbf{I}_{\mathbf{p^{num}}}$.

The pdf of $p$-dimensional multivariate normally distributed $\mathbf{x}_i$ with location parameter $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$ is given by

$$f(\mathbf{x}_i; \boldsymbol{\mu}, \mathbf{\Sigma}) = (2\pi)^{-\frac{p}{2}} \det(\mathbf{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})\right), \tag{25}$$

which, when we fill in $\boldsymbol{\mu} = \mathbf{m}_i$ and $\mathbf{\Sigma} = \mathbf{I}$, evaluates to

$$\begin{aligned} f(\mathbf{x}_i; \mathbf{m}_i, \mathbf{I}) &= (2\pi)^{-\frac{p}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{m}_i)^T(\mathbf{x}_i - \mathbf{m}_i)\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{m}_i)^T(\mathbf{x}_i - \mathbf{m}_i)\right). \end{aligned} \tag{26}$$

Taking the logarithm of this pdf gives us the log-likelihood $\ell(\mathbf{x}_i; \mathbf{m}_i, \mathbf{I})$, for which thus holds that

$$\begin{aligned} \ell(\mathbf{x}_i; \mathbf{m}_i, \mathbf{I}) &\propto -\frac{1}{2}(\mathbf{x}_i - \mathbf{m}_i)^T(\mathbf{x}_i - \mathbf{m}_i) \\ &= -\frac{1}{2}||\mathbf{x}_i - \mathbf{m}_i||_2^2. \end{aligned} \tag{27}$$

Since the loss is minimized when the log-likelihood is maximized, we have $\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) = -\ell(\mathbf{x}_i; \mathbf{m}_i, \mathbf{I})$, such that we arrive at the formulation

$$\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) = \frac{1}{2}||\mathbf{x}_i^{num} - \mathbf{m}_i^{num}||_2^2. \tag{28}$$

This is the exact loss function specified in (4), such that we conclude this proof.

## A.2 Categorical Loss

In the following, we derive the categorical loss function from multinomial distributions with a single trial for each, independent, feature. We use a distribution with a single trial as our formulation defines parameters for every single observation. The corresponding distribution is sometimes also called the *categorical* distribution. For feature $j \in \mathcal{C}$ and observation $i$, where $i = 1, ..., n$, the probability mass function of this distribution is given by

$$f\left(x_{ij}^1, x_{ij}^2, \ldots, x_{ij}^{K_j}; p_{ij}^1, p_{ij}^2, \ldots, p_{ij}^{K_j}\right) = \prod_{k=1}^{K_j} \left(p_{ij}^k\right)^{x_{ij}^k}, \tag{29}$$

where $p_{ij}^k$ denotes the probability that feature $j$ of observation $i$ belongs to class $k$, such that $\sum_{k=1}^{K_j} p_{ij}^k = 1$ and $p_{ij}^k \geq 0$, $\forall k$. Furthermore, $x_{ij}^k \in \{0, 1\}$ is a dummy denoting whether or not feature $j$ of observation $i$ belongs to class $k$. Since an observation can belong to only a single class, i.e., $\sum_{k=1}^{K_j} x_{ij}^k = 1$, and due to the class probabilities adding up to 1, we can rewrite this pmf to

$$f\left(x_{ij}^1, x_{ij}^2, \ldots, x_{ij}^{K_j-1}; p_{ij}^1, p_{ij}^2, \ldots, p_{ij}^{K_j-1}\right) = \left(\prod_{k=1}^{K_j-1} \left(p_{ij}^k\right)^{x_{ij}^k}\right) \left(1 - \sum_{k=1}^{K_j-1} p_{ij}^k\right)^{1-\sum_{k=1}^{K_j-1} x_{ij}^k}. \tag{30}$$

For simplicity, we denote the vector of $K_j - 1$ dummies $x_{ij}^k$ and parameters $p_{ij}^k$ by $\mathbf{x}_{ij}$ and $\mathbf{p}_{ij}$, respectively. Taking the logarithm of the above expression gives us the log-likelihood $\ell(\mathbf{x}_{ij}; \mathbf{p}_{ij})$, given by

$$\ell(\mathbf{x}_{ij}; \mathbf{p}_{ij}) = \sum_{k=1}^{K_j-1} x_{ij}^k \log\left(p_{ij}^k\right) + \left(1 - \sum_{k=1}^{K_j-1} x_{ij}^k\right) \log\left(1 - \sum_{k=1}^{K_j-1} p_{ij}^k\right). \tag{31}$$

Next, we show that, using the logit transform $m_{ij}^k = \log\left(\frac{p_{ij}^k}{1-\sum_{k=1}^{K_j-1} p_{ij}^k}\right)$, we can rewrite $\ell(\mathbf{x}_{ij}; \mathbf{p}_{ij})$ to obtain the categorical loss function in (5).

$$\begin{aligned}
\ell(\mathbf{x}_{ij}; \mathbf{p}_{ij}) &= \sum_{k=1}^{K_j-1} x_{ij}^k \log\left(p_{ij}^k\right) + \left(1 - \sum_{k=1}^{K_j-1} x_{ij}^k\right) \log\left(1 - \sum_{k=1}^{K_j-1} p_{ij}^k\right) \\
&= \sum_{k=1}^{K_j-1} x_{ij}^k \left(\log\left(p_{ij}^k\right) - \log\left(1 - \sum_{k=1}^{K_j-1} p_{ij}^k\right)\right) + \log\left(1 - \sum_{k=1}^{K_j-1} p_{ij}^k\right) \\
&= \sum_{k=1}^{K_j-1} x_{ij}^k \log\left(\frac{p_{ij}^k}{1 - \sum_{k=1}^{K_j-1} p_{ij}^k}\right) - \log\left(\frac{1}{1 - \sum_{k=1}^{K_j-1} p_{ij}^k}\right) \\
&= \sum_{k=1}^{K_j-1} x_{ij}^k \log\left(\frac{p_{ij}^k}{1 - \sum_{k=1}^{K_j-1} p_{ij}^k}\right) - \log\left(1 + \sum_{k=1}^{K_j-1} \frac{p_{ij}^k}{1 - \sum_{k=1}^{K_j-1} p_{ij}^k}\right),
\end{aligned} \tag{32}$$

which can be written in terms of $m_{ij}^k$ to obtain

$$\ell(\mathbf{x}_{ij}, \mathbf{m}_{ij}) = \sum_{k=1}^{K_j-1} x_{ij}^k m_{ij}^k - \log\left(1 + \sum_{k=1}^{K_j-1} \exp\left(m_{ij}^k\right)\right). \tag{33}$$

Independence of features $j \in \mathcal{C}$ implies that the log-likelihood $\ell(\mathbf{x}_i, \mathbf{m}_i)$ is simply the sum of the separate log-likelihoods $\ell(\mathbf{x}_{ij}, \mathbf{m}_{ij})$, or, equivalently,

$$\ell(\mathbf{x}_i, \mathbf{m}_i) = \sum_{j \in \mathcal{C}} \left(\sum_{k=1}^{K_j-1} x_{ij}^k m_{ij}^k - \log\left(1 + \sum_{k=1}^{K_j-1} \exp\left(m_{ij}^k\right)\right)\right). \tag{34}$$

As minimizing a loss function corresponds with maximizing the log-likelihood, we have $\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) = -\ell(\mathbf{x}_i, \mathbf{m}_i)$, such that we arrive at the formulation

$$\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) = \sum_{j \in \mathcal{C}} \left(\sum_{k=1}^{K_j-1} -x_{ij}^k m_{ij}^k + \log\left(1 + \sum_{k=1}^{K_j-1} \exp\left(m_{ij}^k\right)\right)\right). \tag{35}$$

This is the exact same expression as in (5) and thus concludes this derivation.

# B Convex Clustering Boundaries

This section derives the boundary outcomes of the convex clustering problem (3), given by

$$\min_{\mathbf{M}} \; f_\lambda(\mathbf{M}) = (1-\gamma)\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2, \quad (36)$$

where the boundaries are defined in terms of $\lambda$. More specifically, we consider the cases i) where $\lambda = 0$, and ii) where $\lambda$ is so large that a single cluster is formed. The minimum value of $\lambda$ leading to the latter outcome of 'total' clustering is defined as $\lambda^{tot}$.

## B.1 Case I: $\lambda = 0$

When $\lambda = 0$, we can write (36) as

$$\begin{aligned} \min_{\mathbf{M}} \; f_0(\mathbf{M}) &= (1-\gamma)\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) \qquad (37) \\ &= \frac{(1-\gamma)}{2}\sum_{i=1}^{n} ||\mathbf{x}_i^{num} - \mathbf{m}_i^{num}||_2^2 + \gamma \sum_{i=1}^{n}\sum_{j\in\mathcal{C}} \left( \sum_{k=1}^{K_j-1} -x_{ij}^k m_{ij}^k + \log\left(1 + \sum_{k=1}^{K_j-1} \exp\left(m_{ij}^k\right)\right)\right). \end{aligned}$$

Note that the above expression is separable across variables as well as observations and can thus be minimized piecewise. For numerical data, we derive the optimal $\hat{\mathbf{m}}_i^{num}$, $i = 1, \ldots, n$, as

$$\begin{aligned} \frac{\partial f_0(\mathbf{M})}{\partial \mathbf{m}_i^{num}} = 0 \Rightarrow \\ -(1-\gamma)(\mathbf{x}_i^{num} - \mathbf{m}_i^{num}) = 0 \Rightarrow \\ \hat{\mathbf{m}}_i^{num} = \mathbf{x}_i^{num}, \end{aligned} \qquad (38)$$

where we have assumed $\gamma \neq 1$. In case $\gamma = 1$, the optimal centroids $\hat{\mathbf{m}}_i^{num}$ can take on any value. For categorical data, due to its more involved loss function, we derive the optimal $m_{ij}^k$ in element-wise

fashion, or, equivalently,

$$\begin{aligned} \frac{\partial f_0(\mathbf{M})}{\partial m_{ij}^k} = 0 \Rightarrow \\ \gamma \left( -x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1 + \sum_{l=1}^{K_j-1} \exp\left(m_{ij}^l\right)} \right) = 0 \Rightarrow \\ \hat{m}_{ij}^k = \log\left( \frac{x_{ij}^k}{1 - \sum_{l=1}^{K_j-1} x_{ij}^l} \right), \end{aligned} \tag{39}$$

for $i = 1, \ldots, n$, $j \in \mathcal{C}$, and $k = 1, \ldots, K_j - 1$. Here, we have assumed $\gamma \neq 0$. In case $\gamma = 0$, $\hat{m}_{ij}^k$ can take on any value. Note that the above expression for $\hat{m}_{ij}^k$ purely gives a theoretically optimal outcome to the clustering objective. In practice, however, this expression can never be evaluated. This is due to the dummy-coded nature of $x_{ij}^k$ leading to either taking the logarithm of 0 or evaluating a fraction with a denominator equal to 0. This is no problem for us, since we do not run our algorithm for $\lambda = 0$ and, if we did, would converge before getting in trouble.

### B.2 Case II: $\lambda = \lambda^{tot}$

In case $\lambda = \lambda^{tot}$, we have $\mathbf{m}_i = \mathbf{m}_s$, $\forall i, s$, such that the penalty term is 0. We denote the single centroid by $\mathbf{m}$, with its numerical part denoted $\mathbf{m}^{num}$ and its categorical part $\mathbf{m}^{cat}$. This way, (36) is given by

$$\begin{aligned} \min_{\mathbf{M}} \; f_{\lambda^{tot}}(\mathbf{M}) &= (1-\gamma) \sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}^{cat}) \\ &= \frac{(1-\gamma)}{2} \sum_{i=1}^{n} ||\mathbf{x}_i^{num} - \mathbf{m}^{num}||_2^2 + \gamma \sum_{i=1}^{n} \sum_{j \in \mathcal{C}} \left( \sum_{k=1}^{K_j-1} -x_{ij}^k m_j^k + \log\left( 1 + \sum_{k=1}^{K_j-1} \exp\left(m_j^k\right) \right) \right). \end{aligned} \tag{40}$$

Note that the above problem is separable across variables $j$ but not across observations. For numerical data, we can find the optimal $\hat{\mathbf{m}}^{num}$ by solving

$$\begin{aligned} \frac{\partial f_{\lambda^{tot}}(\mathbf{M})}{\partial \mathbf{m}^{num}} = 0 \Rightarrow \\ -(1-\gamma) \sum_{i=1}^{n} (\mathbf{x}_i^{num} - \mathbf{m}^{num}) = 0 \Rightarrow \\ \hat{\mathbf{m}}^{num} = \frac{\sum_{i=1}^{n} \mathbf{x}_i^{num}}{n}, \end{aligned} \tag{41}$$

which simply yields the sample average $\bar{\mathbf{x}}^{num}$ as optimal centroid. Note that, similar to Case I, we have implicitly made the assumption that $\gamma \neq 1$. In case $\gamma = 1$, $\hat{\mathbf{m}}^{num}$ is unconstrained.

For categorical data, we derive the optimal centroids in piecewise fashion, i.e., we perform variable-specific optimization to find $\hat{m}_j^k$. We find this optimal centroid parameter by solving

$$\begin{aligned} \frac{\partial f_{\lambda^{tot}}(\mathbf{M})}{\partial m_j^k} &= 0 \Rightarrow \\ \gamma \sum_{i=1}^{n} \left( -x_{ij}^k + \frac{\exp\left(m_j^k\right)}{1 + \sum_{l=1}^{K_j-1} \exp\left(m_j^l\right)} \right) &= 0 \Rightarrow \\ -\frac{\sum_{i=1}^{n} x_{ij}^k}{n} + \frac{\exp\left(m_j^k\right)}{1 + \sum_{l=1}^{K_j-1} \exp\left(m_j^l\right)} &= 0 \Rightarrow \\ \hat{m}_j^k &= \log\left( \frac{\bar{x}_j^k}{1 - \sum_{l=1}^{K_j-1} \bar{x}_j^l} \right), \end{aligned} \tag{42}$$

where $\bar{x}_j^k = \frac{\sum_{i=1}^{n} x_{ij}^k}{n}$, $j \in \mathcal{C}$, and $k = 1, \ldots, K_j - 1$. Here, we have made the assumption that $\gamma \neq 0$. If $\gamma = 0$, $\hat{m}_j^k$ can take on any value. Note that the above expression for $\hat{m}_j^k$ is evaluable in case each class $k = 1, \ldots, K_j$ occurs at least once in the data. Furthermore, inspecting the above expression, it is easy to see that the optimal centroid $\hat{m}_j^k$ is nothing else than a logit transform of the maximum likelihood estimates of the parameters (i.e., probabilities) from a multinomial distribution.

# C Clustering Formulations

Below, we provide two of the objective functions that we considered as formulations for the mixed data convex clustering task.

## C.1 Formulation I

The formulation we apply in this research is as in (3) and thus given by

$$\min_{\mathbf{M}} (1-\gamma) \sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2,$$

where $0 \leq \gamma \leq 1$ controls the relative influence of numerical and categorical attributes on the clustering task. The interplay between loss functions and penalty term works as follows. In case $\gamma$ is low, the categorical loss function will have a small contribution on the objective function. This leads to categorical centroids being able to freely move without affecting the categorical loss much. As the categorical centroids do have a relatively strong influence on the penalty term contribution, these centroids will be incentivized to move towards each other. This way, the total contribution of the categorical features on the clustering, which is a combination of loss and penalty term, will be limited. Therefore, the clustering will be mostly based on numerical attributes. In similar fashion, a large value of $\gamma$ leads to the numerical centroids being incentivized to move towards each other and the clustering being mostly dependent on the categorical attributes. This is the exact behavior we aim to achieve, such that this formulation is appropriate for the task at hand.

## C.2 Formulation II

We also considered an objective function where $\gamma$ appears in the penalty term, or, equivalently,

$$\begin{aligned} \min_{\mathbf{M}} (1-\gamma) \sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \\ \lambda \sum_{i<s} \omega_{is} \left|\left| \begin{bmatrix} (1-\gamma)(\mathbf{m}_i^{num} - \mathbf{m}_s^{num}) \\ \gamma(\mathbf{m}_i^{cat} - \mathbf{m}_s^{cat}) \end{bmatrix} \right|\right|_2 . \end{aligned} \tag{43}$$

Besides the $(1-\gamma)$ and $\gamma$ terms appearing in the penalty term, this formulation is exactly the same as (3). Due to the adjusted penalty term, the interplay between loss functions and penalty is slightly different. Namely, when $\gamma$ is small, the contributions from both the categorical loss function and

the categorical part of the penalty term are small. This way, regardless of the centroids in $\mathbf{M}^{cat}$, the categorical contribution to the objective function is small. Therefore, the categorical centroids can move rather freely and are hardly incentivized to move in a specific direction.

Besides potentially leading to insensible centroids, this way clustering of categorical centroids will not be enforced. This is contrary to our aim, as, when $\gamma$ is low, i.e., close to 0, the categorical centroids are inherently considered not to be highly informative and should thus move towards each other. In similar fashion, when $\gamma$ is high, i.e., close to 1, we will not directly enforce clustering in numerical attributes. Therefore, we find this formulation less desirable than Formulation I.

# D MIDACC-ADMM

Instead of subgradient descent, we could solve the MIDACC problem using other, potentially more computationally efficient, convex optimization algorithms. One of the most common type of methods is ADMM, which uses variable splitting; the optimization problem is split into different pieces, all of which are optimized sequentially given the values of the other pieces. Chi and Lange (2015) derive ADMM for convex clustering with purely numerical data, which forms the cornerstone on which MIDACC-ADMM is built.

Remember that the MIDACC objective function is given by (3), or,

$$\min_{\mathbf{M}} (1-\gamma)\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2.$$

In order to separate the loss functions and the penalty term, Chi and Lange (2015) propose to recast this problem as the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{M},\mathbf{V}} \quad & (1-\gamma)\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{l\in\mathcal{E}} \omega_l ||\mathbf{v}_l||_2 \\ \text{s.t.} \quad & \mathbf{m}_{l_1} - \mathbf{m}_{l_2} - \mathbf{v}_l = 0, \qquad \forall l \in \mathcal{E}, \end{aligned} \tag{44}$$

where we have introduced the new variables $\mathbf{v}_l = \mathbf{m}_{l_1} - \mathbf{m}_{l_2}$ and the set $\mathcal{E} = \{l = (l_1, l_2) : l_1 < l_2 \text{ and } \omega_{l_1 l_2} > 0\}$. Furthermore, we adopt the notation $\mathbf{V}$ to denote the stacked matrix of $\mathbf{v}_l$, $\forall l \in \mathcal{E}$. Chi and Lange (2015) propose to solve an equality-constrained problem like the one above by using the so-called augmented Lagrangian $\mathcal{L}_\nu(\mathbf{M}, \mathbf{V}, \mathbf{Z})$, which is given by

$$\begin{aligned} \mathcal{L}_\nu(\mathbf{M}, \mathbf{V}, \mathbf{Z}) = & (1-\gamma)\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{l\in\mathcal{E}} \omega_l ||\mathbf{v}_l||_2 \\ & + \sum_{l\in\mathcal{E}} \langle \boldsymbol{\zeta}_l, \mathbf{v}_l - \mathbf{m}_{l_1} + \mathbf{m}_{l_2} \rangle + \frac{\nu}{2} \sum_{l\in\mathcal{E}} ||\mathbf{v}_l - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}||_2^2. \end{aligned} \tag{45}$$

Here, $\boldsymbol{\zeta}_l$, $\forall l \in \mathcal{E}$, is a vector of Lagrange multipliers or dual variables, which, in stacked form, is contained in $\mathbf{Z}$. Also, $\nu$ is a nonnegative tuning parameter and $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}$. The idea of ADMM is to optimize the augmented Lagrangian by means of sequential stepwise updates across the separate

variables. These updates are represented as follows:

$$\mathbf{M}^{t+1} := \underset{\mathbf{M}}{\operatorname{argmin}}\ \mathcal{L}_\nu(\mathbf{M}, \mathbf{V}^t, \mathbf{Z}^t), \tag{46a}$$

$$\mathbf{V}^{t+1} := \underset{\mathbf{V}}{\operatorname{argmin}}\ \mathcal{L}_\nu(\mathbf{M}^{t+1}, \mathbf{V}, \mathbf{Z}^t), \tag{46b}$$

$$\boldsymbol{\zeta}_l^{t+1} := \boldsymbol{\zeta}_l^t + \nu(\mathbf{v}_l^{t+1} - \mathbf{m}_{l_1}^{t+1} + \mathbf{m}_{l_2}^{t+1}), \qquad \forall l \in \mathcal{E}. \tag{46c}$$

Note that the first two updates (46a-46b) are sub-optimization problems, while the update of dual variables is exact. The optimization problem for $\mathbf{V}$ is exactly the same as in Chi and Lange (2015), since this variable term does not appear in the loss functions. Therefore, we have

$$\mathbf{v}_l^{t+1} := \operatorname{prox}_{\frac{\lambda}{\nu}\omega_l||\cdot||_2}(\mathbf{m}_{l_1}^{t+1} - \mathbf{m}_{l_2}^{t+1} - \frac{1}{\nu}\boldsymbol{\zeta}_l^t), \qquad \forall l \in \mathcal{E}, \tag{47}$$

where

$$\operatorname{prox}_{af(\cdot)}(\mathbf{x}) = \underset{\mathbf{v}}{\operatorname{argmin}} \left( af(\mathbf{v}) + \frac{1}{2}||\mathbf{x} - \mathbf{v}||_2^2 \right), \tag{48}$$

is the proximal map function. In our case, we have $a = \frac{\lambda}{\nu}\omega_l$ and $f(\cdot) = ||\cdot||_2$. The exact solution of (47) can, as Chi and Lange (2015) show, be written as

$$\mathbf{v}_l^{t+1} := \left(1 - \frac{\frac{\lambda}{\nu}\omega_l}{||\mathbf{m}_{l_1}^{t+1} - \mathbf{m}_{l_2}^{t+1} - \frac{1}{\nu}\boldsymbol{\zeta}^t||_2}\right)_+ ||\mathbf{m}_{l_1}^{t+1} - \mathbf{m}_{l_2}^{t+1} - \frac{1}{\nu}\boldsymbol{\zeta}^t||_2, \qquad \forall l \in \mathcal{E}, \tag{49}$$

with $(c)_+ = \max\{0, c\}$.

As for the sub-problem of $\mathbf{M}$, we can rewrite this problem to

$$\begin{aligned}
\mathbf{M}^{t+1} &:= \underset{\mathbf{M}}{\operatorname{argmin}}(1-\gamma)\sum_{i=1}^{n}\ell^{num}(\mathbf{x}_i^{num},\mathbf{m}_i^{num})+\gamma\sum_{i=1}^{n}\ell^{cat}(\mathbf{x}_i^{cat},\mathbf{m}_i^{cat})+\lambda\sum_{l\in\mathcal{E}}\omega_l||\mathbf{v}_l^t||_2 \\
&+\sum_{l\in\mathcal{E}}\langle\boldsymbol{\zeta}_l^t,\mathbf{v}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\rangle+\frac{\nu}{2}\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2 \\
&= \underset{\mathbf{M}}{\operatorname{argmin}}(1-\gamma)\sum_{i=1}^{n}\ell^{num}(\mathbf{x}_i^{num},\mathbf{m}_i^{num})+\gamma\sum_{i=1}^{n}\ell^{cat}(\mathbf{x}_i^{cat},\mathbf{m}_i^{cat}) \\
&+\sum_{l\in\mathcal{E}}\langle\boldsymbol{\zeta}_l^t,\mathbf{v}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\rangle+\frac{\nu}{2}\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2 \\
&= \underset{\mathbf{M}}{\operatorname{argmin}}(1-\gamma)\sum_{i=1}^{n}\ell^{num}(\mathbf{x}_i^{num},\mathbf{m}_i^{num})+\gamma\sum_{i=1}^{n}\ell^{cat}(\mathbf{x}_i^{cat},\mathbf{m}_i^{cat}) \\
&+\frac{\nu}{2}\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2,
\end{aligned} \tag{50}$$

which, in contrast to the purely numerical problem of Chi and Lange (2015), does not yield a closed-form expression. Wang and Allen (2021) encounter a similar issue in their ADMM implementation. Based on non-differentiability of the sub-problem objective, they propose to use proximal gradient descent or nested ADMM to update $\mathbf{M}$. However, we note that, in our case, (50) does yield a differentiable expression. This way, we can apply a simple gradient-based optimization scheme to solve the $\mathbf{M}$-update problem. We propose to use Newton's method, since this method is known to generally converge relatively quickly.

We refer to the above optimization problem as $\mathbf{M}^{t+1} := \underset{\mathbf{M}}{\operatorname{argmin}}\ h^{t+1}(\mathbf{M})$, such that

$$\begin{aligned}
h^{t+1}(\mathbf{M}) &= (1-\gamma)\sum_{i=1}^{n}\ell^{num}(\mathbf{x}_i^{num},\mathbf{m}_i^{num})+\gamma\sum_{i=1}^{n}\ell^{cat}(\mathbf{x}_i^{cat},\mathbf{m}_i^{cat}) \\
&+\frac{\nu}{2}\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2.
\end{aligned} \tag{51}$$

Newton's method seeks to optimize $\mathbf{M}^{t+1}$ by means of iterative updates

$$\mathbf{m}_i := \mathbf{m}_i - \left(\frac{\partial^2\left(h^{t+1}(\mathbf{M})\right)}{\partial\mathbf{m}_i\partial\mathbf{m}_i^T}\right)^{-1}\frac{\partial\left(h^{t+1}(\mathbf{M})\right)}{\partial\mathbf{m}_i}, \qquad i=1,\ldots,n, \tag{52}$$

such that we update $\mathbf{m}_i$ by subtracting its inverse Hessian times the gradient. When some convergence criterion is reached, we set $\mathbf{M}^{t+1}=\mathbf{M}$ and complete our $\mathbf{M}$-update. Below, we continue our derivation for updating a single $\mathbf{m}_i$. Note, however, that this derivation applies to all $i=1,\ldots,n$.

In order to implement Newton's method, we need the gradient and Hessian of $h^{t+1}(\mathbf{M})$ to $\mathbf{m}_i$. First of all, we can write the gradient as

$$\begin{aligned}\frac{\partial\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i} &= (1-\gamma)\frac{\partial\left(\sum_{s=1}^{n}\ell^{num}(\mathbf{x}_s^{num},\mathbf{m}_s^{num})\right)}{\partial \mathbf{m}_i} + \gamma\frac{\partial\left(\sum_{s=1}^{n}\ell^{cat}(\mathbf{x}_s^{cat},\mathbf{m}_s^{cat})\right)}{\partial \mathbf{m}_i} \\ &+ \frac{\nu}{2}\frac{\partial\left(\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i}.\end{aligned} \tag{53}$$

We know, from (12a-12b), that

$$\frac{\partial\left(\sum_{s=1}^{n}\ell^{num}(\mathbf{x}_s^{num},\mathbf{m}_s^{num})\right)}{\partial \mathbf{m}_i} = -(\mathbf{x}_i^{num}-\mathbf{m}_i^{num}), \tag{54a}$$

$$\frac{\partial\left(\sum_{s=1}^{n}\ell^{cat}(\mathbf{x}_s^{cat},\mathbf{m}_s^{cat})\right)}{\partial m_{ij}^k} = -x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)}, \quad j\in\mathcal{C},\ k=1,\ldots,K_j-1, \tag{54b}$$

which we derive in Appendix E. Furthermore, we can derive

$$\begin{aligned}&\frac{\partial\left(\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i} = \sum_{l\in\mathcal{E}}\frac{\partial\left(||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i} \\ &= \sum_{l\in\mathcal{E},\ l_1=i}\frac{\partial\left(||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i} + \sum_{l\in\mathcal{E},\ l_2=i}\frac{\partial\left(||\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i} \\ &= -2\sum_{l\in\mathcal{E},\ l_1=i}\left(\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\right) + 2\sum_{l\in\mathcal{E},\ l_2=i}\left(\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\right) \\ &= 2\left(\sum_{l\in\mathcal{E},\ l_2=i}\left(\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\right) - \sum_{l\in\mathcal{E},\ l_1=i}\left(\mathbf{v}_l^t+\frac{1}{\nu}\boldsymbol{\zeta}_l^t-\mathbf{m}_{l_1}+\mathbf{m}_{l_2}\right)\right).\end{aligned} \tag{55}$$

Plugging in the gradient expressions from (54-55) into (53), we obtain the gradient of $h^{t+1}(\mathbf{M})$ to $\mathbf{m}_i$. In order to find the Hessian, we use this gradient alongside the fact that $\frac{\partial^2\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i \partial \mathbf{m}_i^T} = \frac{\partial}{\partial \mathbf{m}_i^T}\left(\frac{\partial\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i}\right)$, such that we have

$$
\begin{aligned}
\frac{\partial^2\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i \partial \mathbf{m}_i^T} &= \frac{\partial}{\partial \mathbf{m}_i^T}\left((1-\gamma)\frac{\partial\left(\sum_{s=1}^{n}\ell^{num}(\mathbf{x}_s^{num},\mathbf{m}_s^{num})\right)}{\partial \mathbf{m}_i} + \gamma\frac{\partial\left(\sum_{s=1}^{n}\ell^{cat}(\mathbf{x}_s^{cat},\mathbf{m}_s^{cat})\right)}{\partial \mathbf{m}_i}\right. \\
&\quad \left. + \frac{\nu}{2}\frac{\partial\left(\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i}\right) \\
&= (1-\gamma)\frac{\partial}{\partial \mathbf{m}_i^T}\left(\frac{\partial\left(\sum_{s=1}^{n}\ell^{num}(\mathbf{x}_s^{num},\mathbf{m}_s^{num})\right)}{\partial \mathbf{m}_i}\right) \\
&\quad + \gamma\frac{\partial}{\partial \mathbf{m}_i^T}\left(\frac{\partial\left(\sum_{s=1}^{n}\ell^{cat}(\mathbf{x}_s^{cat},\mathbf{m}_s^{cat})\right)}{\partial \mathbf{m}_i}\right) \\
&\quad + \frac{\nu}{2}\frac{\partial}{\partial \mathbf{m}_i^T}\left(\frac{\partial\left(\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i}\right).
\end{aligned} \tag{56}
$$

Using the above expression, we can, in parts, derive the Hessian. For the first part (the numerical loss), we use the fact that this part is purely dependent on the numerical part of the centroid matrix, or $\mathbf{M}^{num}$. This means that the categorical centroids have contribution 0 to the Hessian of the numerical loss, such that the Hessian is a block matrix with nonzero entries only in the $(p^{num} \times p^{num})$ block formed by the numerical part of the centroids. We derive this block as

$$
\frac{\partial}{\partial (\mathbf{m}_i^{num})^T}\left(\frac{\partial\left(\sum_{s=1}^{n}\ell^{num}(\mathbf{x}_s^{num},\mathbf{m}_s^{num})\right)}{\partial \mathbf{m}_i^{num}}\right) = \frac{\partial\left(-(\mathbf{x}_i^{num} - \mathbf{m}_i^{num})\right)}{\partial (\mathbf{m}_i^{num})^T} = I_{p^{num}}. \tag{57}
$$

In computing the second part of the Hessian (the categorical loss), we use a similar dependence. That is, the Hessian of the categorical loss is a block matrix with nonzero contribution only for the categorical part of the centroids. To find an expression for this contribution, we perform an element-wise derivation, such that we seek to find

$$
\frac{\partial}{\partial m_{iq}^b}\left(\frac{\partial\left(\sum_{s=1}^{n}\ell^{cat}(\mathbf{x}_s^{cat},\mathbf{m}_s^{cat})\right)}{\partial \mathbf{m}_{ij}^k}\right) = \frac{\partial}{\partial m_{iq}^b}\left(-x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)}\right), \tag{58}
$$

with $j, q \in \mathcal{C}$ and $k, b = 1, \ldots, K_j - 1$. For this expression, the partial derivative depends on the nature of $q$ and $b$. More specifically, there are three cases: i) $j \neq q$, ii) $j = q$ and $k = b$, and iii) $j = q$ and $k \neq b$.

In the first case, we have

$$
\frac{\partial}{\partial m_{iq}^b}\left(-x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)}\right) = 0, \qquad j \neq q, \tag{59}
$$

since none of the terms are dependent on $q$ and $b$ when $j \neq q$.

In the second case, we derive

$$\frac{\partial}{\partial m_{ij}^k}\left(-x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)}\right) = \frac{\exp\left(m_{ij}^k\right)\left(1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)\right) - \exp\left(m_{ij}^k\right)^2}{\left(1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)\right)^2} \tag{60}$$
$$= \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)} - \frac{\exp\left(m_{ij}^k\right)^2}{\left(1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)\right)^2}.$$

For the third case, we can write

$$\frac{\partial}{\partial m_{ij}^b}\left(-x_{ij}^k + \frac{\exp\left(m_{ij}^k\right)}{1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)}\right) = \frac{-\exp\left(m_{ij}^k\right)\exp\left(m_{ij}^b\right)}{\left(1+\sum_{l=1}^{K_j-1}\exp\left(m_{ij}^l\right)\right)^2}, \qquad k \neq b. \tag{61}$$

Putting together the three cases, we get the complete second part of the Hessian, which is the part based on the categorical loss.

The third part of the Hessian yields

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{m}_i^T}\left(\frac{\partial\left(\sum_{l\in\mathcal{E}}||\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}||_2^2\right)}{\partial \mathbf{m}_i}\right) = \dots \\
\dots &= \frac{\partial}{\partial \mathbf{m}_i^T}\left(2\left(\sum_{l\in\mathcal{E},\ l_2=i}\left(\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right) - \sum_{l\in\mathcal{E},\ l_1=i}\left(\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right)\right)\right) \\
&= 2\sum_{l\in\mathcal{E},\ l_2=i}\frac{\partial}{\partial \mathbf{m}_i^T}\left(\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right) - 2\sum_{l\in\mathcal{E},\ l_1=i}\frac{\partial}{\partial \mathbf{m}_i^T}\left(\mathbf{v}_l^t + \frac{1}{\nu}\boldsymbol{\zeta}_l^t - \mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right) \\
&= 2\sum_{l\in\mathcal{E},\ l_2=i}\frac{\partial}{\partial \mathbf{m}_i^T}\left(-\mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right) - 2\sum_{l\in\mathcal{E},\ l_1=i}\frac{\partial}{\partial \mathbf{m}_i^T}\left(-\mathbf{m}_{l_1} + \mathbf{m}_{l_2}\right) \\
&= 2\sum_{l\in\mathcal{E}_i}\mathbf{I}_p,
\end{aligned} \tag{62}$$

where $\mathcal{E}_i = \{l = (l_1, l_2) : l_1 < l_2 \text{ and } (l_1 = i \text{ or } l_2 = i)\}$.

Plugging in the expressions from (57), (59-61), and (62) into (56), we obtain the Hessian of $h^{t+1}(\mathbf{M})$ to $\mathbf{m}_i$. Using this Hessian alongside the gradient from (53), we can iteratively update $\mathbf{m}_i$ following (52). Doing so for all $i = 1, \dots, n$ until convergence, we solve the $\mathbf{M}$-update step of the ADMM algorithm. To put ADMM in MIDACC into perspective, we provide Algorithm 3.

In Algorithm 3, we use the convergence criteria on primal and dual residuals as specified by Boyd et al. (2011) and worked out by Chi and Lange (2015). Furthermore, the $Newton(\cdot)$ function

**Algorithm 3:** MIDACC-ADMM

**Data:** Data matrix $\underset{(n\times p)}{\mathbf{X}}$ ; $\phi$; $K$; $\gamma$; $\lambda$; $\delta$; $\nu$

**Result:** Optimal centroids $\mathbf{M}^*$; optimal $\mathbf{V}^*$; optimal $\mathbf{Z}^*$

Compute pairwise weights $\omega_{is} = I_{k,is}\exp(-\phi d(\mathbf{x}_i, \mathbf{x}_s)), \qquad \forall i, s = 1, \ldots, n$

Initialize $\mathbf{V}^0$ and $\mathbf{Z}^0$; set t $\leftarrow 0$; $\mathbf{M}^0 \leftarrow \mathbf{X}$

**while** *convergence criteria not met* **do**

  $\mathbf{M}^{t+1} := Newton(\mathbf{M}^t, \mathbf{V}^t, \mathbf{Z}^t)$

  **for** $\forall l \in \mathcal{E}$ **do**

    $\mathbf{v}_l^{t+1} := \left(1 - \frac{\frac{\lambda}{\nu}\omega_l}{||\mathbf{m}_{l_1}^{t+1} - \mathbf{m}_{l_2}^{t+1} - \frac{1}{\nu}\boldsymbol{\zeta}^t||_2}\right)_+ ||\mathbf{m}_{l_1}^{t+1} - \mathbf{m}_{l_2}^{t+1} - \frac{1}{\nu}\boldsymbol{\zeta}^t||_2$

    $\boldsymbol{\zeta}_l^{t+1} := \boldsymbol{\zeta}_l^t + \nu(\mathbf{v}_l^{t+1} - \mathbf{m}_{l_1}^{t+1} + \mathbf{m}_{l_2}^{t+1})$

  $t := t + 1$

$\mathbf{M}^* \leftarrow \mathbf{M}^t$

$\mathbf{V}^* \leftarrow \mathbf{V}^t$

$\mathbf{Z}^* \leftarrow \mathbf{Z}^t$

**Algorithm 4:** *Newton*

**Data:** $\mathbf{M}^t$; $\mathbf{V}^t$; $\mathbf{Z}^t$

**Result:** Updated centroids $\mathbf{M}^{t+1}$

Initialize $\mathbf{M} \leftarrow \mathbf{M}^t$

**while** *convergence criteria not met* **do**

  **for** $\forall i = 1, \ldots, n$ **do**

    Compute gradient $\frac{\partial\left(h^t(\mathbf{M})\right)}{\partial \mathbf{m}_i}$ using equation (53)

    Compute Hessian $\frac{\partial^2\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i \partial \mathbf{m}_i^T}$ using equation (56)

    $\mathbf{m}_i := \mathbf{m}_i - \left(\frac{\partial^2\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i \partial \mathbf{m}_i^T}\right)^{-1} \frac{\partial\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i}$

$\mathbf{M}^{t+1} \leftarrow \mathbf{M}$

performs Newton's method for the **M**-update step, as explicitly presented in Algorithm 4.

In Algorithm 4, we simply calculate the gradient and Hessian for all separate observation centroids and update these until some convergence criterion is reached. Similar to MIDACC-SGD, we choose to terminate the algorithm when either the Frobenius norm of the stacked gradients $\frac{\partial\left(h^{t+1}(\mathbf{M})\right)}{\partial \mathbf{m}_i}$ over $i$ is smaller than a certain relative threshold $\epsilon$, or when the iteration count exceeds a threshold, e.g., 1000.

Note that, in general, ADMM is a faster optimization method than subgradient descent. This

way, computation-wise, MIDACC-ADMM may be preferred over MIDACC-SGD. However, as we previously noted, this research emphasizes interpretability over computational benefits, such that we implement MIDACC-SGD as our main method. In addition, we should note that although ADMM itself generally requires few iterations, a single **M**-update requires a full run of Algorithm 4. This may be computationally costly and could thus lead to diminishing computational benefits of MIDACC-ADMM over MIDACC-SGD. However, remedies for this issue have recently been proposed; namely, instead of running Newton's method until convergence, Weylandt et al. (2020) and Wang and Allen (2021) advise to perform a single iteration before moving on to the **V**- and **Z**-updates. This is an interesting proposition for future research.

# E Gradient Derivations

Below, we derive the gradients of the numerical and categorical loss functions as well as the penalty term with respect to $\mathbf{M}$.

## E.1 Numerical Loss

The gradient of the numerical loss function $\ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})$ with respect to $\mathbf{m}_i^{num}$, $i = 1, \ldots, n$, is given by

$$\begin{aligned} \frac{\partial \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i^{num}} &= \frac{\partial \left( \frac{1}{2} ||\mathbf{x}_i^{num} - \mathbf{m}_i^{num}||_2^2 \right)}{\partial \mathbf{m}_i^{num}} \\ &= -(\mathbf{x}_i^{num} - \mathbf{m}_i^{num}). \end{aligned} \tag{63}$$

## E.2 Categorical Loss

The gradient of the categorical loss function $\ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})$ with respect to $\mathbf{m}_i^{cat}$, $i = 1, \ldots, n$ is easily derived element-wise. Thus, we calculate the derivative with respect to $m_{ij}^k$, or,

$$\begin{aligned} \frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial m_{ij}^k} &= \frac{\partial \left( \sum_{j \in \mathcal{C}} \left( \sum_{l=1}^{K_j - 1} -x_{ij}^l m_{ij}^l + \log \left( 1 + \sum_{l=1}^{K_j - 1} \exp \left( m_{ij}^l \right) \right) \right) \right)}{\partial m_{ij}^k} \\ &= \frac{\partial \left( -x_{ij}^k m_{ij}^k + \log \left( 1 + \sum_{l=1}^{K_j - 1} \exp \left( m_{ij}^l \right) \right) \right)}{\partial m_{ij}^k} \\ &= -x_{ij}^k + \frac{\exp \left( m_{ij}^k \right)}{1 + \sum_{l=1}^{K_j - 1} \exp \left( m_{ij}^l \right)}, \end{aligned} \tag{64}$$

for $j \in \mathcal{C}$ and $k = 1, \ldots, K_j - 1$.

### E.3 Penalty Term

We compute the gradient of the penalty term $\lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2$ with respect to $\mathbf{m}_i$, $i = 1, \ldots, n$, as follows:

$$\begin{aligned}
\frac{\partial \left( \sum_{r<s} \omega_{rs} ||\mathbf{m}_r - \mathbf{m}_s||_2 \right)}{\partial \mathbf{m}_i} &= \lambda \left( \sum_{s>i} \omega_{is} \frac{\partial \left( ||\mathbf{m}_i - \mathbf{m}_s||_2 \right)}{\partial \mathbf{m}_i} + \sum_{s<i} \omega_{si} \frac{\partial \left( ||\mathbf{m}_s - \mathbf{m}_i||_2 \right)}{\partial \mathbf{m}_i} \right) \\
&= \lambda \sum_{s \neq i} \omega_{is} \frac{\partial \left( ||\mathbf{m}_i - \mathbf{m}_s||_2 \right)}{\partial \mathbf{m}_i} \\
&= \lambda \sum_{s \neq i} \omega_{is} \frac{\partial \left( \left( (\mathbf{m}_i - \mathbf{m}_s)^T (\mathbf{m}_i - \mathbf{m}_s) \right)^{\frac{1}{2}} \right)}{\partial \mathbf{m}_i} \\
&= \lambda \sum_{s \neq i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2},
\end{aligned} \tag{65}$$

where the second line follows from symmetry in the pairwise weights and Euclidean distance, i.e., $\omega_{is} = \omega_{si}$ and $||\mathbf{m}_i - \mathbf{m}_s||_2 = ||\mathbf{m}_s - \mathbf{m}_i||_2$.

# F Penalty Subgradient Derivation

In this section, we derive the subgradient of the penalty term $\sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2$ that we provided in (13). Note that, for simplicity reasons, we exclude the constant $\lambda$ from this derivation. Since we know that the gradient of a sum is equal to the sum of gradients, we can rewrite this subgradient to

$$g_i \left( \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2 \right) = \sum_{i<s} \omega_{is} g_{is} \left( ||\mathbf{m}_i - \mathbf{m}_s||_2 \right), \tag{66}$$

where $g_{is} \left( ||\mathbf{m}_i - \mathbf{m}_s||_2 \right)$ is the subgradient belonging to the pair of observations $i, s = 1, \ldots, n$. For simplicity, we use $\boldsymbol{\beta}_{is} = g_{is} \left( ||\mathbf{m}_i - \mathbf{m}_s||_2 \right)$. Before deriving the form of $\boldsymbol{\beta}_{is}$, let us take a look at the definition of a subgradient. We say that $\mathbf{g}$ is a subgradient of function $f : D \to R$ if

$$f(\mathbf{x} + \mathbf{a}) \geq f(\mathbf{x}) + \langle \mathbf{g}, \mathbf{a} \rangle, \qquad \text{for all } a \in D, \tag{67}$$

where $\langle \mathbf{g}, \mathbf{a} \rangle = \mathbf{g}^T \mathbf{a}$. In our case, let $f(\cdot)$ be the Euclidean distance, that is, $f(\mathbf{x}) = ||\mathbf{x}||_2$. Furthermore, $\mathbf{g} = \boldsymbol{\beta}_{is}$, and let $\mathbf{x} = \mathbf{0}$ and $\mathbf{a} = \mathbf{m}_i - \mathbf{m}_s$, such that (67) resolves to

$$||\mathbf{m}_i - \mathbf{m}_s||_2 \geq \langle \boldsymbol{\beta}_{is}, (\mathbf{m}_i - \mathbf{m}_s) \rangle, \qquad \forall \mathbf{m}_i, \mathbf{m}_s \in \mathbb{R}^p. \tag{68}$$

In case $\mathbf{m}_i - \mathbf{m}_s \neq 0$, we can analytically solve for $\boldsymbol{\beta}_{is}$ (see Appendix E.3), which leads to the solution

$$\boldsymbol{\beta}_{is} = \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2}, \qquad \mathbf{m}_i \neq \mathbf{m}_s. \tag{69}$$

In case $\mathbf{m}_i - \mathbf{m}_s = 0$, we cannot evaluate this expression. However, using (68), we can specify the form of $\boldsymbol{\beta}_{is}$ even in this case. Firstly, let us compare $\boldsymbol{\beta}_{is}$ with $\boldsymbol{\beta}_{si}$. For the latter subgradient, we get

$$||\mathbf{m}_s - \mathbf{m}_i||_2 \geq \langle \boldsymbol{\beta}_{si}, (\mathbf{m}_s - \mathbf{m}_i) \rangle \Leftrightarrow ||\mathbf{m}_i - \mathbf{m}_s||_2 \geq \langle \boldsymbol{\beta}_{si}, -(\mathbf{m}_i - \mathbf{m}_s) \rangle, \tag{70}$$

which, in case $\boldsymbol{\beta}_{si} = -\boldsymbol{\beta}_{is}$, gives us the exact same expression as (68). Thus, if $\boldsymbol{\beta}_{is}$ is a valid subgradient, we take the negative of this subgradient to obtain a valid subgradient $\boldsymbol{\beta}_{si}$. This is also established from (69), where $\boldsymbol{\beta}_{is} = -\boldsymbol{\beta}_{si}$. Therefore, from now on, we use $\boldsymbol{\beta}_{is} = -\boldsymbol{\beta}_{si}$.

To extract information on the size of the subgradient $\boldsymbol{\beta}_{is}$, we use

$$\begin{aligned}\langle \boldsymbol{\beta}_{is}, (\mathbf{m}_i - \mathbf{m}_s)\rangle &= ||\langle \boldsymbol{\beta}_{is}, (\mathbf{m}_i - \mathbf{m}_s)\rangle||_2 \\ &\leq ||\boldsymbol{\beta}_{is}^T||_2 ||\mathbf{m}_i - \mathbf{m}_s||_2 \\ &= ||\boldsymbol{\beta}_{is}||_2 ||\mathbf{m}_i - \mathbf{m}_s||_2 \\ &\leq ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \text{if } ||\boldsymbol{\beta}_{is}||_2 \leq 1,\end{aligned} \tag{71}$$

where we have used the Cauchy-Schwarz inequality $||\mathbf{ab}||_2 \leq ||\mathbf{a}||_2 ||\mathbf{b}||_2$. This derivation implies that $||\boldsymbol{\beta}_{is}||_2 \leq 1$ is a sufficient condition on the magnitude of the subgradient. Furthermore, Hoefling (2010) derives that $||\boldsymbol{\beta}||_2 > 1$ does not satisfy (68).

Thus, we have established that

$$\boldsymbol{\beta}_{is} = \begin{cases} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2}, & \text{if } \mathbf{m}_i \neq \mathbf{m}_s \\ \{\boldsymbol{\beta}_{is} : \boldsymbol{\beta}_{is} = -\boldsymbol{\beta}_{si} \text{ and } ||\boldsymbol{\beta}_{is}||_2 \leq 1\}, & \text{if } \mathbf{m}_i = \mathbf{m}_s. \end{cases} \tag{72}$$

Therefore, we can rewrite (66) to

$$\begin{aligned} g_i\left(\sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2\right) &= \sum_{i<s} \omega_{is} \boldsymbol{\beta}_{is} \\ &= \sum_{s\neq i,\ \mathbf{m}_s \neq \mathbf{m}_i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2} + \sum_{s\neq i,\ \mathbf{m}_s = \mathbf{m}_i} \omega_{is} \boldsymbol{\beta}_{is}, \end{aligned} \tag{73}$$

with $\boldsymbol{\beta}_{is} = -\boldsymbol{\beta}_{si}$ and $||\boldsymbol{\beta}_{is}||_2 \leq 1$. This concludes the derivation of (13).

# G Extended Sensitivity Analysis on $\gamma$

Here, we present extended results for the sensitivity analysis on $\gamma$ as given in Section 5.1.4, thus providing results for all five experiments from Sections 4.1.1 and 4.1.2. We do so using all proposed $\gamma$ measures as well as values of $\gamma$ ranging between 0 and 1 with stepsize 0.1. Here, we use notation $\text{proto}_1$ and $\text{proto}_2$ for $\gamma = \frac{h^{num}}{h^{num}+h^{cat}}$ where we calculate $h^{num}$ and $h^{cat}$ using (18a-18b). More specifically, $\text{proto}_1$ uses $h_{cat} = \frac{1}{p^{cat}} \sum_{j \in \mathcal{C}} \left(1 - \sum_{k=1}^{K_j} \left(\hat{p}_j^k\right)^2\right)$, while $\text{proto}_2$ has $h_{cat} = \frac{1}{p^{cat}} \sum_{j \in \mathcal{C}} \left(1 - \max_k \hat{p}_j^k\right)$. For reference, we also provide baseline results.

*Table 9: Mean performance of MIDACC for various choices of $\gamma$. Experiments I and II refer to the spherical and non-spherical experiments from Section 4.1.1, respectively. Experiments III, IV, and V refer to the experiments from Section 4.1.2 with noise and signal in both data types, signal in categorical but noise in numerical data, and signal in numerical but noise in categorical data, respectively.*

| | Experiment I | | Experiment II | | Experiment III | | Experiment IV | | Experiment V | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* |
| $\gamma$ - 0 | 0.890 | 0.614 | 0.981 | 0.926 | 0.969 | 0.879 | 0.535 | 0.001 | 0.969 | 0.880 |
| $\gamma$ - 0.1 | 0.972 | 0.893 | 0.986 | 0.946 | 0.989 | 0.955 | 0.637 | 0.162 | 0.968 | 0.877 |
| $\gamma$ - 0.2 | 0.986 | 0.945 | 0.984 | 0.937 | 0.992 | 0.968 | 0.954 | 0.823 | 0.966 | 0.870 |
| $\gamma$ - 0.3 | 0.983 | 0.932 | 0.975 | 0.903 | 0.990 | 0.960 | 0.957 | 0.836 | 0.960 | 0.845 |
| $\gamma$ - 0.4 | 0.981 | 0.924 | 0.962 | 0.852 | 0.983 | 0.935 | 0.958 | 0.840 | 0.942 | 0.784 |
| $\gamma$ - 0.5 | 0.976 | 0.908 | 0.943 | 0.785 | 0.976 | 0.908 | 0.957 | 0.836 | 0.899 | 0.655 |
| $\gamma$ - 0.6 | 0.972 | 0.892 | 0.910 | 0.676 | 0.966 | 0.869 | 0.958 | 0.838 | 0.793 | 0.403 |
| $\gamma$ - 0.7 | 0.970 | 0.885 | 0.892 | 0.616 | 0.963 | 0.858 | 0.958 | 0.839 | 0.743 | 0.299 |
| $\gamma$ - 0.8 | 0.970 | 0.885 | 0.879 | 0.578 | 0.961 | 0.852 | 0.957 | 0.836 | 0.688 | 0.202 |
| $\gamma$ - 0.9 | 0.969 | 0.882 | 0.864 | 0.534 | 0.959 | 0.844 | 0.957 | 0.837 | 0.644 | 0.113 |
| $\gamma$ - 1.0 | 0.964 | 0.860 | 0.576 | 0.097 | 0.949 | 0.807 | 0.941 | 0.808 | 0.515 | 0.001 |
| $\gamma$ - loss | 0.986 | 0.944 | 0.981 | 0.925 | 0.991 | 0.964 | 0.956 | 0.831 | 0.962 | 0.856 |
| $\gamma$ - dist | 0.984 | 0.938 | 0.973 | 0.895 | 0.991 | 0.964 | 0.955 | 0.826 | 0.961 | 0.850 |
| $\gamma$ - $\text{proto}_1$ | 0.972 | 0.892 | 0.920 | 0.707 | 0.969 | 0.881 | 0.958 | 0.837 | 0.847 | 0.531 |
| $\gamma$ - $\text{proto}_2$ | 0.971 | 0.888 | 0.908 | 0.666 | 0.967 | 0.871 | 0.958 | 0.837 | 0.804 | 0.429 |
| $\gamma$ - mod | 0.986 | 0.944 | 0.984 | 0.937 | 0.991 | 0.963 | 0.956 | 0.830 | 0.967 | 0.873 |
| KAMILA | 0.986 | 0.944 | 0.967 | 0.872 | 0.993 | 0.974 | 0.652 | 0.200 | 0.971 | 0.887 |
| K-prototypes | 0.986 | 0.946 | 0.973 | 0.894 | 0.990 | 0.961 | 0.952 | 0.817 | 0.962 | 0.852 |
| HAC | 0.960 | 0.846 | 0.886 | 0.600 | 0.944 | 0.789 | 0.940 | 0.775 | 0.649 | 0.120 |

## H MIDACC Convexity in $\gamma$

In this section, we prove that, given $\mathbf{M}$, the MIDACC objective function is convex in $\gamma$. Remember that the MIDACC objective function is given by

$$\min_{\mathbf{M}} (1-\gamma) \sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \gamma \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2,$$

which, given $\mathbf{M}$, we consequently denote by $f(\gamma|\mathbf{X}, \mathbf{M})$. This proof relies on the pairwise weights being dense, i.e., $\omega_{is} = \exp(-\phi d(\mathbf{x}_i, \mathbf{x}_s))$. Thus, we do not use sparse, KNN, weights. Invoking sparsity namely leads to non-differentiability of the weights to $\gamma$. Furthermore, we use

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_s) &= \frac{1-\gamma}{p^{num}} \sum_{j \in \mathcal{N}} \frac{|x_{ij} - x_{sj}|}{\max_i x_{ij} - \min_i x_{ij}} + \frac{\gamma}{p^{cat}} \sum_{j \in \mathcal{C}} I(x_{ij} \neq x_{sj}) \\ &= (1-\gamma) d^{num}(\mathbf{x}_i, \mathbf{x}_s) + \gamma d^{cat}(\mathbf{x}_i, \mathbf{x}_s), \end{aligned} \tag{74}$$

for $i, s = 1, \ldots, n$. In order to prove convexity in $\gamma$, it suffices to show that $\frac{\partial^2 f(\gamma|\mathbf{X},\mathbf{M})}{\partial \gamma^2} > 0 \Leftrightarrow \frac{\partial}{\partial \gamma}\left(\frac{\partial f(\gamma|\mathbf{X},\mathbf{M})}{\partial \gamma}\right) > 0$. We derive

$$\begin{aligned} \frac{\partial f(\gamma|\mathbf{X}, \mathbf{M})}{\partial \gamma} &= -\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) \\ &\quad + \lambda\phi \sum_{i<s} \left(d^{num}(\mathbf{x}_i, \mathbf{x}_s) - d^{cat}(\mathbf{x}_i, \mathbf{x}_s)\right) \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2. \end{aligned} \tag{75}$$

Thus,

$$\begin{aligned} \frac{\partial^2 f(\gamma|\mathbf{X}, \mathbf{M})}{\partial \gamma^2} &= \frac{\partial}{\partial \gamma} \left( -\sum_{i=1}^{n} \ell^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \sum_{i=1}^{n} \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) \right. \\ &\quad \left. + \lambda\phi \sum_{i<s} \left(d^{num}(\mathbf{x}_i, \mathbf{x}_s) - d^{cat}(\mathbf{x}_i, \mathbf{x}_s)\right) \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2 \right) \\ &= \lambda\phi \sum_{i<s} \left(d^{num}(\mathbf{x}_i, \mathbf{x}_s) - d^{cat}(\mathbf{x}_i, \mathbf{x}_s)\right) ||\mathbf{m}_i - \mathbf{m}_s||_2 \frac{\partial \omega_{is}}{\partial \gamma} \\ &= \lambda\phi^2 \sum_{i<s} \left(d^{num}(\mathbf{x}_i, \mathbf{x}_s) - d^{cat}(\mathbf{x}_i, \mathbf{x}_s)\right)^2 \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2 \\ &> 0, \qquad \text{if } |C| > 1, \end{aligned} \tag{76}$$

which concludes this proof. The inequality follows from $\lambda > 0$, $\phi^2 > 0$, $\left(d^{num}(\mathbf{x}_i, \mathbf{x}_s) - d^{cat}(\mathbf{x}_i, \mathbf{x}_s)\right)^2 > 0$ and $\omega_{is} > 0$ for some pair $\{i, s\}$, and $||\mathbf{m}_i = \mathbf{m}_s||_2 > 0$ for some $i, s$ with $\mathbf{m}_i \neq \mathbf{m}_s$. The latter property holds iff $|C| > 1$. This makes sense, as the second partial derivative of the objective function to $\gamma$ purely originates from the penalty term, which is 0 in case $|C| = 1$. Note that we assume that $\lambda > 0$, which is logical given the fact that, when $\lambda = 0$, no regularization takes place. In this case, we can analytically solve the MIDACC objective function, for which we provide solutions, which are independent of $\gamma$, in Appendix B.1.

# I W-MIDACC

In this appendix, we present, in detailed fashion, W-MIDACC, the corresponding W-MIDACC-SGD algorithm, and some rationale on its potential through illustrative experiments.

In MIDACC, we apply standardization to unit variance for numerical and dummy-coding for categorical attributes. This scheme makes sure that all attributes are processed on the same scale, such that undesirably and unnecessarily disproportional variable contributions are corrected for. In addition, through the MIDACC parameter $\gamma$, we allow the user to control the relative weight placed on numerical and categorical features when employing mixed data clustering. Thus, within the two data types, we unify the scale of measurement, and across data types, we allow for user-specific control on the clustering contribution. However, this formulation does not allow for specific variables being up- or downweighted in the clustering process. Since it is realistic that we may want to weight some variables more strongly than others, below we introduce a feature-specific weighted formulation of MIDACC, or W-MIDACC.

The idea of W-MIDACC is simple; we employ variable-specific weights in the objective function (3) in similar fashion to incorporating $\gamma$. This set-up, which is inspired by the W-K-means formulation of Huang et al. (2005), allows variables to differently contribute to the clustering task.

As previously noted, the MIDACC loss functions are based on the assumption of conditional independence across attributes. This way, we can write the combined loss function across attributes as the sum of loss functions per attribute, or

$$\ell(\mathbf{X}, \mathbf{M}) = \sum_j \ell(\mathbf{x}_j, \mathbf{m}_j). \tag{77}$$

In similar fashion, this means that we can write the variable-weighted loss function $\ell_w(\mathbf{X}, \mathbf{M})$ as the weighted sum of attribute-specific losses, or,

$$\ell_w(\mathbf{X}, \mathbf{M}) = \sum_j w_j \ell_w(\mathbf{x}_j, \mathbf{m}_j). \tag{78}$$

Using the regular loss functions (4-5), we can write the W-MIDACC loss functions as

$$\ell_w^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) = \frac{1}{2}||(\mathbf{w}^{num})^T(\mathbf{x}_i^{num} - \mathbf{m}_i^{num})||_2^2, \qquad i = 1, \ldots, n, \tag{79a}$$

$$\ell_w^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) = \sum_{j \in \mathcal{C}} w_j \left( \sum_{k=1}^{K_j - 1} -x_{ij}^k m_{ij}^k + \log\left(1 + \sum_{k=1}^{K_j-1} \exp\left(m_{ij}^k\right)\right)\right), \quad i = 1, \ldots, n, \tag{79b}$$

where $(\mathbf{w}^{num})^T = [w_1 \quad w_2 \quad \ldots \quad w_{p^{num}}]$. The objective function can now be written as

$$\min_{\mathbf{M}} \sum_{i=1}^{n} \ell_w^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num}) + \sum_{i=1}^{n} \ell_w^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat}) + \lambda \sum_{i<s} \omega_{is} ||\mathbf{m}_i - \mathbf{m}_s||_2, \tag{80}$$

where $w_j \geq 0$, $j = 1, \ldots, p$, and $\sum_{j=1}^{p} w_j = 1$. For notation purposes, the complete weight vector can be written as $\mathbf{w} = \begin{bmatrix}\mathbf{w}^{num} & \mathbf{w}^{cat}\end{bmatrix}$ with $\mathbf{w}^{cat} = \begin{bmatrix}w_1 & w_2 & \ldots & w_{p^{cat}}\end{bmatrix}$ and $\mathbf{w}^{num}$ as before. Note that we do not specify parameter $\gamma$ in the objective anymore; via the variable-specific weights, the relative effect of categorical variables can now be directly accounted for. As with $\gamma$, we also incorporate the variable-specific weights in the distance function $d(\mathbf{x}_i, \mathbf{x}_s)$ used for computing pairwise weights $\omega_{is}$. We can write this distance function as

$$\begin{aligned}\omega_{is} &= I_{k,is} \exp\left(-\phi d(\mathbf{x}_i, \mathbf{x}_s)\right) \\ &= I_{k,is} \exp\left(-\phi \left(\frac{1}{p^{num}} \sum_{j \in \mathcal{N}} w_j \frac{|x_{ij} - x_{sj}|}{\max_i x_{ij} - \min_i x_{ij}} + \frac{1}{p^{cat}} \sum_{j \in \mathcal{C}} w_j I(x_{ij} \neq x_{sj})\right)\right).\end{aligned} \tag{81}$$

In terms of interpretation, W-MIDACC works as follows. A larger value of $w_j$ causes feature $j$ to be more important in the clustering task. When $w_j = 0$, the variable is considered negligible. In case all variables are considered equally important prior to running W-MIDACC, we can use weights $w_j = \frac{1}{p}, \forall j = 1, \ldots, p$. In case we wish to use type-specific weights, we can set $w_j = \frac{(1-\gamma)}{p}$, $j \in \mathcal{N}$ and $w_j = \frac{\gamma}{p}$, $j \in \mathcal{C}$. Thus, the weighted formulation encaptures regular MIDACC, albeit with weights scaled by the number of features $p$. Since the weights $w_j$ are simple constants independent of $\mathbf{M}$, the (sub)gradient calculations do not change apart from a variable-specific multiplicative factor. In addition, due to the nonnegative weights and piecewise convexity across features, the W-MIDACC formulation is additionally convex. Using this fact, we can run W-MIDACC with only some minor adjustments to Algorithm 1.

Note that, similar to regular MIDACC, the variable-specific weights do not appear in the penalty term itself. In line with the reasoning provided in Appendix C, this is due to the behavior that smaller

variable-specific weights inherently lead to relatively larger weights for that variable being put on the penalty term, driving cluster centroids on that variable towards each other without bearing (much) costs. This means that the clustering is then driven by the other variables for which the objective function remains a trade-off between loss and penalty. This is the exact behavior we wish to achieve.

Were we to put the variable-specific weights in the penalty term, this would mean that, in case $w_j$ is small, the centroids of feature $j$ are hardly incentivized to take on specific values. This way, we do not achieve clustering on these variables, but rather average their 'random' values when merging centroids based on the other features (that have larger weight). This is not the behavior we aim for, such that we choose not to apply variable-specific weights in the penalty term.

In terms of parameter selection, we need to specify the variable weights in $\mathbf{w}$. To do so, we propose to use pre-specified weights. Besides saving computation time, we do so as we view these variable-specific weights more as a 'user flexibility' feat of the algorithm than an optimization task. Thus, we recommend the user to input weights based on domain knowledge or context-specific wishes. Optimization of the weights given the data is outside the scope of this paper.

## I.1 W-MIDACC-SGD

In similar fashion to regular MIDACC, we can solve the W-MIDACC convex optimization problem (80) by means of a subgradient descent (SGD) algorithm, which we dub W-MIDACC-SGD. This algorithm is highly similar to Algorithm 1, although there are some small differences.

Due to the variable-specific weights, the (sub)gradients of W-MIDACC differ from those derived for MIDACC. We can write the partial derivatives of the loss functions $\ell_w^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})$ and $\ell_w^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})$, and the penalty term $\sum_{i<s} \omega_{rs} ||\mathbf{m}_i - \mathbf{m}_s||_2$, respectively, as

$$\frac{\partial \ell_w^{num}(\mathbf{x}_i^{num}, \mathbf{m}_i^{num})}{\partial \mathbf{m}_i^{num}} = -\mathbf{w}^{num} \circ (\mathbf{x}_i^{num} - \mathbf{m}_i^{num}), \tag{82a}$$

$$\frac{\partial \ell^{cat}(\mathbf{x}_i^{cat}, \mathbf{m}_i^{cat})}{\partial m_{ij}^k} = -w_j x_{ij}^k + \frac{w_j \exp\left(m_{ij}^k\right)}{1 + \sum_{l=1}^{K_j - 1} \exp\left(m_{ij}^l\right)}, \quad j \in \mathcal{C},\ k = 1, \ldots, K_j - 1, \tag{82b}$$

$$\frac{\partial \left(\sum_{r<s} \omega_{rs} ||\mathbf{m}_r - \mathbf{m}_s||_2\right)}{\partial \mathbf{m}_i} = \sum_{s \neq i} \omega_{is} \frac{\mathbf{m}_i - \mathbf{m}_s}{||\mathbf{m}_i - \mathbf{m}_s||_2}, \tag{82c}$$

where $\circ$ denotes the Hadamard (entrywise) vector product and $i = 1, \ldots, n$. Note that the above partial derivatives differ from those of MIDACC only through the variable-specific weight factor in the loss functions. The form of the penalty term partial derivative (82c) is exactly the same as that

in (12c), although we now apply the pairwise weights from (81). This way, the subgradient of the penalty term for observation $i$ has the same form as in (13) and is given by

$$g_i\left(\sum_{i<s}\omega_{is}||\mathbf{m}_i-\mathbf{m}_s||_2\right)=\sum_{s\neq i,\ \mathbf{m}_s\neq\mathbf{m}_i}\omega_{is}\frac{\mathbf{m}_i-\mathbf{m}_s}{||\mathbf{m}_i-\mathbf{m}_s||_2}+\sum_{s\neq i,\ \mathbf{m}_s=\mathbf{m}_i}\omega_{is}\boldsymbol{\beta}_{is}, \tag{83}$$

with $\boldsymbol{\beta}_{is}=-\boldsymbol{\beta}_{si}$ and $||\boldsymbol{\beta}_{is}||_2\leq 1$. The subgradient of the penalty term for cluster $c$ is as in (14), or,

$$g_c\left(\sum_{i<s}\omega_{is}||\mathbf{m}_i-\mathbf{m}_s||_2\right)=\frac{1}{|c|}\sum_{i\in c}\left(\sum_{s\neq i,\ \mathbf{m}_s\neq\mathbf{m}_i}\omega_{is}\frac{\mathbf{m}_i-\mathbf{m}_s}{||\mathbf{m}_i-\mathbf{m}_s||_2}\right),\qquad i\in c. \tag{84}$$

We can now write the subgradient of the W-MIDACC objective function as

$$\begin{aligned}g_c(f(\mathbf{M}))=&\frac{1}{|c|}\sum_{i\in c}\left(\frac{\partial\ell_w^{num}(\mathbf{x}_i^{num},\mathbf{m}_i^{num})}{\partial\mathbf{m}_i}+\frac{\partial\ell_w^{cat}(\mathbf{x}_i^{cat},\mathbf{m}_i^{cat})}{\partial\mathbf{m}_i}\right)\\&+\lambda g_c\left(\sum_{i<s}\omega_{is}||\mathbf{m}_i-\mathbf{m}_s||_2\right),\end{aligned} \tag{85}$$

where we can fill in the expressions (82a-82b) and (84) for the loss functions and the penalty term, respectively. Using these expressions, we can solve the W-MIDACC convex optimization problem using subgradient descent, for which we provide Algorithm 5.

In Algorithm 5, the function $W-SubGradient(\mathbf{X},\mathbf{M},\mathbf{w})$ computes the cluster subgradients based on (85). We compute the minimum fusion distance $\delta$ slightly differently from the computation (16) used in MIDACC. Namely, we use

$$\delta=\frac{1}{2}\min_{i,s}\left(||(\mathbf{w}^{num})^T(\mathbf{x}_i^{num}-\mathbf{x}_s^{num})||_2+\sum_{j\in\mathcal{C}}w_jI(x_{ij}\neq x_{sj})\right). \tag{86}$$

Note that the rest of the algorithm is the same as in MIDACC-SGD, i.e., we use the same procedures for line search, fusion, and convergence. In similar fashion to Appendix D, we can also extend W-MIDACC to apply an ADMM-type algorithm.

**Algorithm 5:** W-MIDACC-SGD

**Data:** Data matrix $\underset{(n\times p)}{\mathbf{X}}$; $\phi$; $K$; variable weights $\underset{(p\times 1)}{\mathbf{w}}$; $\lambda$; minimum fusion distance $\delta$

**Result:** Optimal centroids $\underset{(n\times p)}{\mathbf{M}^*}$; optimal set of clusters $C^*$

Compute pairwise weights $\omega_{is} = I_{k,is}\exp(-\phi d(\mathbf{x}_i, \mathbf{x}_s)), \qquad \forall i, s = 1, \ldots, n$

Initialize t $\leftarrow 1$; $\mathbf{M}^0 \leftarrow \mathbf{X}$; $\mathbf{G}^0 \leftarrow W - SubGradient(\mathbf{X}, \mathbf{M}^0, \mathbf{w})$; $C \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}$

**while** *convergence criteria not met* **do**

  $\alpha \leftarrow LineSearch(\mathbf{M}^{t-1}, \mathbf{G}^{t-1}, t)$

  $\mathbf{M}^t \leftarrow \mathbf{M}^{t-1} - \alpha\mathbf{G}^{t-1}$

  $d^* \leftarrow \min_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  $i^*, s^* \leftarrow \operatorname{argmin}_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  **while** $d^* < \delta$ **do**

    $\mathbf{M}^t, C \leftarrow Fuse(\mathbf{m}_{i^*}, \mathbf{m}_{s^*})$

    **if** $|C| = 1$ **then**

      **break**

    $d^* \leftarrow \min_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

    $i^*, s^* \leftarrow \operatorname{argmin}_{i,s} ||\mathbf{m}_i - \mathbf{m}_s||_2, \qquad \mathbf{m}_i \neq \mathbf{m}_s$

  $\mathbf{G}^t \leftarrow W - SubGradient(\mathbf{X}, \mathbf{M}^t, \mathbf{w})$

  $t := t + 1$

$\mathbf{M}^* \leftarrow \mathbf{M}^{t-1}$

$C^* \leftarrow C$

## I.2 Usage

In our opinion, W-MIDACC could be used for two ends. Firstly, we note that W-MIDACC increases flexibility of the MIDACC framework. That is, if the user aims to cluster a data set from an online webshop and is more interested in clustering based on product data compared to user data, this user can assign larger weight to the product attributes compared to user features without having to remove entire user features from the analysis. This behavior makes for a convenient situation that can potentially be extremely valuable.

Secondly, given a data set where some features prove to be more important for the clustering than others, W-MIDACC can improve performance. We show some numerical examples in Table 10 using the three signal-noise experiments from Section 4.1.2, where we use $r = 200$ replications. We assume prior knowledge on the signal and noise features is available and assign 8 times as large

weights to signal features compared to noise features.

In Table 10, experiments I, II, and III refer to the experiments with signal and noise in both data types, signal in categorical but noise in numerical data, and signal in numerical but noise in categorical data, respectively. For completeness, we have also included MIDACC and baseline results. Although MIDACC already exhibits strong performance for all three experiments, we show that W-MIDACC is able to increase this performance. This is a promising result, which underlines the value that lies in further exploring W-MIDACC.

*Table 10: Mean performance of (W-)MIDACC and the baseline methods for the three experiments from Section 4.1.2. Experiments I, II, and III refer to the cases with noise and signal in both data types, signal in categorical but noise in numerical data, and signal in numerical but noise in categorical data, respectively.*

| | Experiment I | | Experiment II | | Experiment III | |
|---|---|---|---|---|---|---|
| | *Acc* | *ARI* | *Acc* | *ARI* | *Acc* | *ARI* |
| W-MIDACC | 0.992 | 0.968 | 0.961 | 0.849 | 0.970 | 0.880 |
| MIDACC | 0.989 | 0.956 | 0.961 | 0.849 | 0.957 | 0.834 |
| KAMILA | 0.992 | 0.968 | 0.624 | 0.158 | 0.974 | 0.899 |
| K-prototypes | 0.987 | 0.950 | 0.953 | 0.820 | 0.965 | 0.864 |
| HAC | 0.944 | 0.791 | 0.942 | 0.781 | 0.651 | 0.113 |