Optimising cluster interpretability by increasing prototype dissimilarity

Hidde Dirkse (564706)

Abstract

With the expanding use of machine learning methods, the development of interpretable machine learning is becoming increasingly important. To enable a broader audience to benefit from machine learning results, the research field should focus on building interpretable machine learning methods. Our work enhances the interpretability of cluster analysis (CA) by adding an extra term in the objective function of an existing Mixed Integer Programming formulation (Carrizosa et al., 2022). The original formulation focuses on finding an optimal explanation per cluster that maximises precision and distinctiveness. This means that the chosen explanation should include as many individuals from the corresponding cluster as possible, while excluding as many individuals from other clusters as possible. The explanation is based on the distance from an individual to the chosen prototype. The prototype is chosen per cluster and is intuitively explained as the most 'average' individual in that cluster. In our work, we propose an extra distance term in the objective function that penalises for chosen prototypes that lie closely together. This term encourages the model to choose prototypes that are dissimilar. As the prototypes are chosen to be further apart from each other, the explanations for each cluster will be more distinct as well. Making it easier to interpret the variations between clusters following a CA. Running our experiments on a *mall customer* dataset we find a 9.4% and 25.0% increase in dissimilarity compared to the original model without a significant loss in distinctiveness and precision. Thereby, showing that with a marginal loss in distinctiveness and precision a more interpretable solution can be found.

Keywords: Machine Learning · Interpretability · Cluster Analysis · Prototypes · Mixed-Integer Programming

Supervisor:	Rick Willemsen
Second assessor:	Riley Badenbroek
Date version:	1st July 2024
University:	Erasmus University Rotterdam
School:	Erasmus School of Economics
Program:	Econometrics and Operations Research

1 Introduction

As the application of machine learning methods grows, there is an increasing need for methods that are easy to use and interpret. Interpretable machine learning is a research field entirely focused on building trust in models, performing model debugging, and informing real human decision-making by creating models and methods that are easier to interpret (V. Chen et al., 2022). A major drawback of using complex machine learning methods is that their workings are complicated and hard to grasp for non-experts. For a broader audience to fully benefit from the advances in computer science, interpretable methods are sometimes preferred over state-of-the-art methods that might only provide marginal improvements in results.

In our work, we focus on the machine learning method cluster analysis (CA). This method groups data points into clusters that have high similarity based on a distance metric (Ma & Wu, 2007). This technique is widely used for decision-making, so having interpretable results helps decision-makers better understand the outcomes of CA. The applications of CA are large, but the main motivation for using this technique is usually to help identify patterns and structures in the dataset that are hard to detect by solely observing the data.

Especially within marketing CA plays a crucial role in grouping customers that show similar behaviour. Using these customer segments, specific marketing strategies can be applied to different groups of customers. Cluster targeted marketing can therefore provide large benefits for companies and other organisations. Other fields that use cluster analysis are image processing and biology (Nugent & Meila, 2010) (Mittal et al., 2022). There are two common models to improve the interpretability of CA. Intrinsic models improve interpretability while at the same time forming clusters for the given data. Post-hoc models on the other hand, were developed to increase the interpretability of existing clusters. Since for some data the CA is already performed we can only use the appointed cluster label and use that as information to increase interpretability. In this work we propose a post-hoc model that enhances the interpretability of existing CA by means of prototypes.

In CA, the explanation is often distance-based, meaning that individuals that are close to each other are expected to be in the same cluster, while individuals that are far apart are expected to be in the other clusters. We can explain a cluster by the distance of each individual to the prototype (Rousseeuw & Kaufman, 2009). The prototype serves as something that could be intuitively explained as being the "average" individual in the cluster. From this individual we measure the distance to the other data points in the cluster. To see whether a prototype is a good explanation for the other data points in the cluster, we use two metrics. The first metric is called true positive rate (TPR), which counts the individuals that are correctly explained by the prototype. The second metric is the false positive rate (FPR) which counts the individuals that were supposed to be explained by the prototype, but were not. To find the optimal set of prototypes we use two different Mixed Integer Linear Programming (MILP) formulations.

The main aim of our research is to improve the interpretability of existing CA by increasing the distance between the prototypes in the optimal set. To test our methods we use 3 different datasets, one on Canadian weather, one simulated dataset to test model robustness for large instances and one to evaluate increased interpretability on a marketing dataset. We manage to find optimal sets of solutions that are up to 25% further apart while maintaining acceptable values for TPR and FPR. Showing a trade off between differing prototypes and performance in terms of TPR and FPR.

The rest of the paper is structured as follows: in section 2, we cover relevant related work. In section 3, the data we use to test our model is discussed. Then, in section 4 our methods are introduced and in section 5 the results are shown. Last, in section 6 we give our conclusion and future research directions.

2 Related Work

In this section, we elaborate on the relevant current work surrounding the topic of CA and, more specifically, the interpretability of CA. We provide background information to better understand the proposed methods.

Machine learning models or algorithms can be classified as using either supervised learning or unsupervised learning. Supervised learning is a technique that uses labelled datasets to train models. By doing so, it tries to accurately predict the labels from the data for a given new data point. In contrast, unsupervised learning is a technique that does not use a labelled dataset for training. The model tries to identify patterns and structures in the data without any prior knowledge provided through labelled datasets. CA is an unsupervised learning technique and although unsupervised learning methods often deliver promising results, they can suffer from a lack of clear interpretability (V. Chen et al., 2022). In this work, we aim to improve the interpretability of cluster analysis by using prototypes in combination with Mixed Integer Linear Programming formulations.

CA is a machine learning technique that groups data points into different clusters based on their similarities. Over the years, multiple methods have been proposed to form these clusters. There are partitional methods, such as K-means clustering (MacQueen et al., 1967), which divide the data into a fixed number of clusters, and hierarchical methods, which use a distance-based rule to create nested series of partitions (Sneath, 2005). More sophisticated methods have also been developed to handle special cases or clusters with irregular shapes. Examples include model-based clustering (Hastie et al., 2009) and density-based clustering (Ester et al., 1996).

Improving the interpretability of CA has been a significant research focus. Two main approaches exist in this domain. Firstly, intrinsic models perform CA while simultaneously building explanations for the clusters they create. Examples of these methods are the discriminative rectangle mixture model (DReaM) (J. Chen et al., 2016) and the interpretable clustering via optimal trees model (ICOT) (Bertsimas et al., 2021). The second approach involves post-hoc methods, which enhance the interpretability of a previously performed CA. In this scenario, the only available information is the cluster label assigned to a data point during the CA. This approach verifies whether the given rule-based explanations are satisfactory by evaluating whether individuals satisfy a list of features (Davidson et al., 2018). Our work also presents a post-hoc model, largely inspired by a paper that aims to increase interpretability (Carrizosa et al., 2022).

Recent advancements in explainable artificial intelligence have also influenced the field of CA interpretability. Techniques such as SHAP (SHapley Additive exPlanations) (Lundberg & Lee, 2017) and LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016) have been adapted to provide local explanations for cluster assignments. These methods help in understanding the contribution of individual features to the clustering decisions.

Furthermore, the integration of human centered approaches in interpretability, such as user studies and interactive visualisation tools, has been explored to bridge the gap between technical explanations and user comprehension (Choo & Liu, 2018). These approaches aim to make cluster analysis more accessible and understandable to non-expert users.

In summary, the interpretability of cluster analysis is an evolving field with various methodologies ranging from intrinsic models to post-hoc explanations and human centered approaches. Our contribution lies in enhancing interpretability by leveraging prototypes and MILP, thereby providing more interpretable CA results.

3 Data

In this section we consider the three different datasets that we use for testing our methods. In section 4.1 and 4.2 we describe the datasets used in the original work to test the workings of the covering and partitioning model (Carrizosa et al., 2022). In section 4.3, we explain a newly introduced mall customer dataset to evaluate the workings of our proposed extension on the covering model.

3.1 Canadian weather data

This dataset contains 365 observations on the daily temperatures of 35 Canadian cities. All of these cities are clustered in one of four climates: Atlantic, Pacific, Continental or Arctic. This data is publicly available in the *R* package *fda* and shown in figure 1.

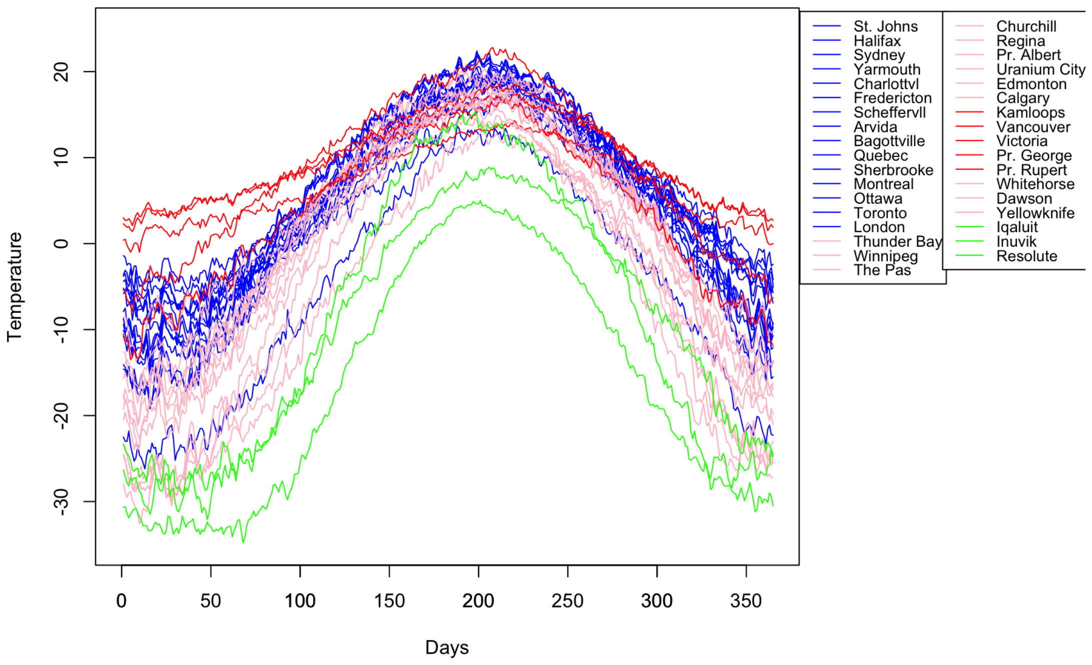

Figure 1: This figure shows the 35 cities from the Canadian weather dataset. It displays the average temperature over 365 days and the colours of the lines correspond to the climate in this city.

3.2 Simulated data

To demonstrate the model's capability to handle large datasets, we use simulated data comprising of three distinct clusters. Each cluster c is constructed by drawing from a multivariate normal distribution $N(\beta^c, \sum^c)$. Specifically, the individuals within each cluster c are drawn according to the following parameters:

$$\beta^1 = (1.45, 1.5) \qquad \beta^2 = (1.8, 1.6) \qquad \beta^3 = (1.4, 2.0)$$

$$\sum^1 = \begin{bmatrix} 0.01 & 0.00 \\ 0.00 & 0.02 \end{bmatrix} \qquad \sum^2 = \begin{bmatrix} 0.02 & 0.00 \\ 0.00 & 0.02 \end{bmatrix} \qquad \sum^3 = \begin{bmatrix} 0.03 & 0.00 \\ 0.00 & 0.04 \end{bmatrix}$$

This simulated approach allows us to effectively test the model's performance and scalability when applied to large datasets. By analysing the model's handling of these large datasets we can validate its robustness and efficiency in real-world applications involving substantial data volumes. In figure 2 you can see what the three clusters look like.

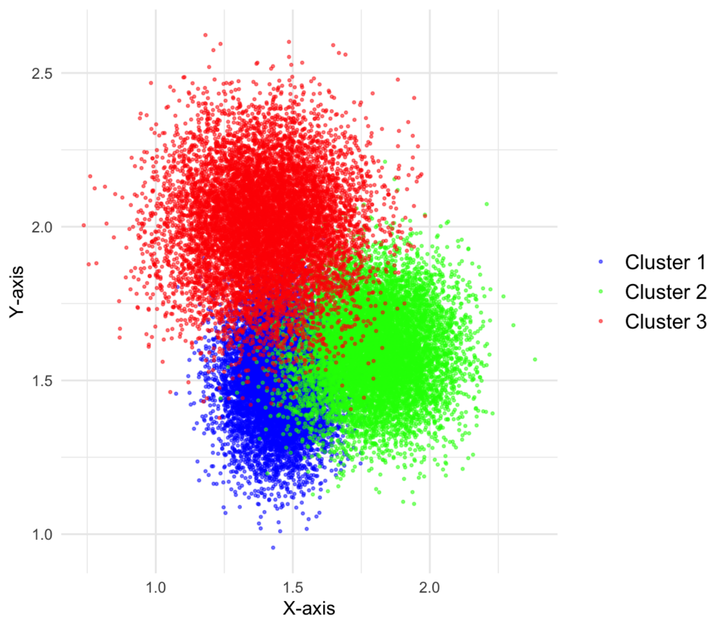

Figure 2: This figure shows the three generated clusters using the multivariate normal distributions indicated in section 2. In this particular example figure $\mathscr{N} = 10^4$.

3.3 Mall customer data

In our work we use a third and final dataset to showcase the increase in interpretability when adding an extra distance term. The dataset that we utilise is a marketing dataset on mall customers. This dataset can be found on Kaggle [1]. In this dataset we find 200 individuals. For each of those individuals the following 5 variables are denoted: *gender, age, annual income* and *spending score*. The variable *gender* equals 1 when the individual is a male and 0 when the individual is a female. The *annual income* is denoted in 1000 US Dollars per year and the variable *spending score* is a score from 1-99 where 1 means little spending and 99 means large spending. We use this dataset to illustrate the effectiveness on distance based interpretability. Next to that, this dataset contains multidimensional numeric data, which is something common in marketing. We show that our methods also provide sensible interpretations when applied to multiple variables. To prepare the dataset for running the adjusted versions of the covering model. We firstly need to perform a clustering ourselves as there are no cluster labels present in the dataset. In this case, we use k-means clustering with $k = 5$. The obtained clusters are used for the implementation of the adjusted covering model.

4 Methodology

In this section we explain the methods that we use in three subsections. In section 4.1 and 4.2 we explain the two models we use in our work (Carrizosa et al., 2022). In section 4.3, we explain our proposed extension for the covering model to optimise the distance between selected prototypes.

To explain the models used we first need to introduce some general definitions and notation.

[1]https://www.kaggle.com/datasets/shwetabh123/mall-customers/data

We define a set of individuals $\mathscr{N}$, where all individuals have a cluster label $c \in \mathscr{C}$. Where $\mathscr{C}$ is the set of all clusters. We want to assign these individuals an optimal prototype. We also define a set of prototypes $\mathscr{I}$, from this set we choose a prototype that is used as the explanation of the cluster it is appointed to. We try to improve interpretability by formulating an explanation for each cluster c that is as precise and distinct as possible i.e. having the highest true positive rate (TPR) and the lowest false positive rate (FPR). The TPR and FPR are measures defined per cluster. The TPR represents the fraction of individuals out of all individuals that have cluster label c, that is correctly explained by the selected prototype for cluster c. The FPR represents the fraction of individuals out of all individuals that have cluster label c, that is supposed to be explained by the prototype for cluster c but was not. In our model we want the selected prototype for cluster c to be correct for as many individuals as possible within the cluster. Next to that, we want the prototype not to explain individuals that are not in the cluster that the prototype is in, as this would increase the value of the FPR. For both models we are given the same existing clustering $\mathscr{C}$ that is obtained by clustering $\mathscr{N}$ individuals in c clusters. This yields $\mathscr{N}_c$ where $\mathscr{N} = \bigcup_{c\in\mathscr{C}} \mathscr{N}_c$. The prototypes $\mathscr{I}_c$ are selected from $\mathscr{N}_c$. Essentially, the prototypes are the same set of individuals but they are separately considered for the purpose of the MILP formulation. It also holds up for the set of prototypes that $\mathscr{I} = \bigcup_{c\in\mathscr{C}} \mathscr{I}_c$. Furthermore, we have the dissimilarity matrix δ that contains the distance between prototype $i \in \mathscr{I}$ and individual $n \in \mathscr{N}$. The distance is calculated by means of the Euclidean distance. As we work on a post-hoc method we assume that we are only given the clusters and not necessarily the dissimilarity matrix used to construct them. Therefore, this is not necessarily the same dissimilarity matrix that is used to create the clusters in $\mathscr{C}$. Furthermore, we introduce the binary decision variable z_i:

$$z_i = \begin{cases} 1 & \text{if } i \text{ is chosen as prototype,} \\ 0 & \text{otherwise.} \end{cases}$$

4.1 The covering model

In the covering model, we define a threshold value r_c that represents the maximum distance that can be between an individual j and the prototype i for a cluster c. All individuals that have a distance smaller than r_c are considered to be in cluster c, and all individuals with a value larger than r_c are considered to be outside of cluster c. In this optimisation we thus need to find the set of prototypes as well as the radii for the different clusters.

Let r_c be the radius of cluster c. For each prototype $i \in \mathscr{I}$, we define the binary decision variable π_{in}:

$$\pi_{in} = \begin{cases} 1 & \text{individual } n \text{ lies in the ball of radius } r_c \text{ spanned by prototype } i, \\ 0 & \text{otherwise.} \end{cases}$$

To prevent a bi-linear formulation of the problem, we can apply the Fortet transformation for decision variables π_{in} and z_i (Fortet, 1960). We define a new decision variable $y_{in} = \pi_{in} z_i$. We can denote the number of true positive cases as $\sum_{i\in\mathscr{I}_c} \sum_{n\in\mathscr{N}_c} y_{in}$ and the true positive rate as: $TPR_c = \frac{\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c} y_{in}}{|\mathscr{N}_c|}$. Similarly, we can write down the number of false positive cases as $\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c} y_{in}$ and the false positive rate as: $FPR_c = \frac{\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c} y_{in}}{|\mathscr{N}\setminus\mathscr{N}_c|}$.

Given all notation, the set covering model consists of the following equations:

$$\begin{aligned}
\max_{y,r} \quad & \sum_{c\in\mathscr{C}}\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c} y_{in} - \theta\sum_{c\in\mathscr{C}}\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c} y_{in}, && && (1)\\
\text{s.t.} \quad & \sum_{i\in\mathscr{I}_c} z_i = 1, && \forall c\in\mathscr{C}, && (2)\\
& r_c \geq \delta_{in}\pi_{in}, && \forall (i,n)\in\mathscr{I}_c\times\mathscr{N}_c, \forall c\in\mathscr{C}, && (3)\\
& r_c \leq \delta_{in} + (r_c^{\max} - \delta_{in})\pi_{in}, && \forall (i,n)\in\mathscr{I}_c\times\mathscr{N}\setminus\mathscr{N}_c, \forall c\in\mathscr{C}, && (4)\\
& \sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c} \pi_{in}z_i \geq \lceil\lambda_c|\mathscr{N}_c|\rceil, && \forall c\in\mathscr{C}, && (5)\\
& \sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c} \pi_{in}z_i \leq \lfloor\mu_c|\mathscr{N}\setminus\mathscr{N}_c|\rfloor, && \forall c\in\mathscr{C}, && (6)\\
& r_c^{\min} \leq r_c \leq r_c^{\max}, && \forall c\in\mathscr{C}, && (7)\\
& y_{in} \leq \pi_{in}, && \forall (i,n)\in\mathcal{I}_c\times\mathcal{N}, \forall c\in\mathcal{C}, && (8)\\
& y_{in} \leq z_i, && \forall (i,n)\in\mathcal{I}_c\times\mathcal{N}, \forall c\in\mathcal{C}, && (9)\\
& y_{in} \geq \pi_{in} + z_i - 1, && \forall (i,n)\in\mathcal{I}_c\times\mathcal{N}, \forall c\in\mathcal{C}, && (10)\\
& y_{in} \in \{0,1\}, && \forall (i,n)\in\mathcal{I}_c\times\mathcal{N}, \forall c\in\mathcal{C}, && (11)\\
& z_i \in \{0,1\}, && \forall i\in\mathscr{I}_c, \forall c\in\mathscr{C}, && (12)\\
& \pi_{in} \in \{0,1\}, && \forall (i,n)\in\mathscr{I}_c\times\mathscr{N}, \forall c\in\mathscr{C}. && (13)
\end{aligned}$$

The objective function (1) maximises the number of true positive cases over all clusters while minimising the number of false positive cases over all clusters with penalty term θ. Constraints (2) ensure that for each cluster exactly one prototype is chosen. Constraints (3) and (4) make sure that π_{in} is well defined. Whenever an individual is farther away from the prototype than the threshold r_c, π_{in} has to equal 0. This is ensured by constraints (3). For all individuals outside of cluster c, we need to make sure that if $r_c > \delta_{in}$ then π_{in} equals 1. In the case that $r_c = \delta_{in}$, $\pi_{in} = 1$ for the individuals that are in cluster c and $\pi_{in} = 0$ for the individuals who are not. The constraints (5) and (6) allow for a minimum or maximum requirement for the TPR and FPR respectively. We can adjust this requirement by varying the parameter values of $\lambda_c \in [0,1]$ or $\mu_c \in [0,1]$. Lastly, constraints (7) indicate that r_c must always be between the minimum and maximum value for the radius. We define these bounds as, $r_c^{\min} = \min_{(i,n)\in\mathcal{I}_c\times\mathcal{N}_c, i\neq n}\delta_{in}$ and $r_c^{\max} = \max_{(i,n)\in\mathcal{I}_c\times\mathcal{N}_c}\delta_{in}$. Constraints (8), (9) and (10) are the result of the Fortet transformation and ensure that y_{in} is well defined as the product of the decision variables π_{in} and z_i. Constraints (11), (12) and (13) indicate that z_i, π_{in} and y_{in} are binary variables.

We provide an additional approach for instances that have many individuals. The essence of this approach revolves around using a partial solution with a smaller number of individuals to optimise the larger instance. To obtain the partial solution we apply a sampling procedure to select a smaller group of prototypes and individuals $\widetilde{\mathscr{I}_c} \subset \mathscr{I}_c$ and $\widetilde{\mathscr{N}_c} \subset \mathscr{N}_c$. We use these newly defined sets to solve a 'reduced' covering model with chosen prototypes z_i^R and radii r_c^R. We use this solution as the partial solution for the larger original problem and define $\mathbf{z^O} = \mathbf{z^R}$ and $\mathbf{r^O} = \mathbf{r^R}$. The solution $(\mathbf{z^O}, \pi^{\mathbf{O}}, \mathbf{r^O})$ is found when constraints (5) and (6) are satisfied. This procedure is applicable to the partitioning model too.

4.2 The partitioning model

In the partitioning model, the cluster c consists of the individuals that are closest to the prototype of cluster c out of all the prototypes of the other clusters. We define one extra variable ρ_{in} which indicates whether individual n is closest to prototype i:

$$\rho_{in} = \begin{cases} 1 & \text{if individual } n \text{ is closest to prototype } i, \\ 0 & \text{otherwise.} \end{cases}$$

For this formulation, the true positive cases are calculated as $\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c}\rho_{in}$ and the false positive cases similarly as $\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c}\rho_{in}$. Due to this change the TPR_c and FPR_c become $TPR_c = \frac{\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c}\rho_{in}}{|\mathscr{N}_c|}$ and $FPR_c = \frac{\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c}\rho_{in}}{|\mathscr{N}\setminus\mathscr{N}_c|}$.

We formulate the partitioning model as follows:

$$\max_{z,\rho} \quad \sum_{c\in\mathcal{C}}\sum_{i\in\mathcal{I}_c}\sum_{n\in\mathcal{N}_c}\rho_{in} - \theta\sum_{c\in\mathcal{C}}\sum_{i\in\mathcal{I}_c}\sum_{n\in\mathcal{N}\setminus\mathcal{N}_c}\rho_{in}, \tag{14}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}_c} z_i = 1, \qquad \forall c\in\mathcal{C}, \tag{15}$$

$$\sum_{j\in\mathcal{I}_c:\delta_{jn}\le\delta_{in}} z_j + \sum_{j\in\mathcal{I}:\delta_{jn}>\delta_{in}} \rho_{jn} \le 1, \qquad \forall (i,n)\in\mathcal{I}_c\times\mathcal{N}, \forall c\in\mathcal{C}, \tag{16}$$

$$\rho_{in} \le z_i, \qquad \forall (i,n)\in\mathcal{I}\times\mathcal{N}, \tag{17}$$

$$\sum_{i\in\mathcal{I}}\rho_{in} = 1, \qquad \forall n\in\mathcal{N}, \tag{18}$$

$$\sum_{i\in\mathcal{I}_c}\sum_{n\in\mathcal{N}_c}\rho_{in} \ge \lceil\lambda_c|\mathcal{N}_c|\rceil, \qquad \forall c\in\mathcal{C}, \tag{19}$$

$$\sum_{i\in\mathcal{I}_c}\sum_{n\in\mathcal{N}\setminus\mathcal{N}_c}\rho_{in} \le \lfloor\mu_c|\mathcal{N}\setminus\mathcal{N}_c|\rfloor, \qquad \forall c\in\mathcal{C}, \tag{20}$$

$$z_i \in \{0,1\}, \qquad \forall i\in\mathcal{I}, \tag{21}$$

$$\rho_{in} \in \{0,1\}, \qquad \forall (i,n)\in\mathcal{I}\times\mathcal{N}. \tag{22}$$

The objective function for this formulation is similar to the one in the covering model, only $\pi_{in}z_i$, which is equal to y_{in}, is replaced by ρ_{in}. The objective function also maximises the TPR and minimises the FPR that is weighted by θ. Constraints (15) are equal to constraints (2) and also ensure that for each cluster exactly one prototype is selected. Constraints (19) and (20) have the same function in controlling the TPR_c and the FPR_c using λ and μ as in (5) and (6). The main differences here are in constraints (16)-(18). Constraints (16) realise that an individual is assigned to the closest prototype by leveraging that for each cluster only one prototype is chosen. Constraints (17) ensure that individuals can only be considered closest to prototypes that are selected in the optimisation. Constraints (18) ensure that each individual is closest to exactly one prototype. Lastly, constraints (21) and (22) define the nature of the decision variables.

When the number of individuals is large, we can apply the same reduction technique that is used for the covering model. We take the same steps, first reducing the original data $\widetilde{\mathscr{I}_c} \subset \mathscr{I}_c$ and $\widetilde{\mathscr{N}_c} \subset \mathscr{N}_c$. Then, using these smaller sets to find a partial solution to the larger problem. To obtain a feasible solution for the larger instance we use the found partial solution.

4.3 Distance models

To increase interpretability further, we introduce an extra term in the objective function of the covering model. This extra term encourages the model to find an optimal set of prototypes that are far apart. In this way the cluster representatives will be more distinct, and therefore make interpretation of the different clusters easier. We need the distance term to only penalise or reward for prototypes that are part of the selected set. A logical choice would be to multiply z_i with z_j to only consider selected prototypes, but then we obtain a quadratic formulation which is less efficient than a linear formulation. Therefore, we introduce binary decision variable $h_{ij} = z_i z_j$ again using a Fortet transformation (Fortet, 1960).

We propose a penalty term that can be added to the existing objective function, with two variations represented by $\sum_{i,j\in\mathscr{I},i\neq j} h_{ij}P(\delta_{ij})$. To obtain the penalty term, we define $P(\delta_{ij})$ for each model.

The first expression is an inverse distance penalty, which we introduce as:

$$-\alpha \sum_{i,j\in\mathscr{I},i\neq j} h_{ij}\frac{1}{\delta_{ij}}.$$

Thus, for the inverse penalty model, $P(\delta_{ij}) = -\alpha\frac{1}{\delta_{ij}}$. We sum over all combinations of prototypes where $i \neq j$. In the summation, we multiply h_{ij} with the inverse of the distance between prototype i and prototype j, where δ_{ij} is the distance between prototype i and j. This distance originates from the dissimilarity matrix δ that was introduced at the beginning of the methodology. By taking the inverse distance, we strongly penalise prototypes that are close together, thus encouraging solutions with prototypes that lie far apart from each other.

The second expression is the squared distance term, which we define as:

$$\beta \sum_{i,j\in\mathscr{I},i\neq j} h_{ij}\delta_{ij}^2.$$

Thus, for the squared distance model, $P(\delta_{ij}) = \beta\delta_{ij}^2$. As in the inverse distance penalty, we sum over all combinations of i and j without them being equal. In this expression, we multiply h_{ij} with the squared distance. By multiplying with the squared distance, we encourage the model to select prototypes that are far apart. The parameters α and β ensure that the penalty terms are of the same magnitude as the TPR and FPR for the given problem.

Below, we display the newly introduced constraints that should be added to the covering model:

$$\max_{y,r} \quad \sum_{c\in\mathscr{C}}\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}_c} y_{in} - \theta\sum_{c\in\mathscr{C}}\sum_{i\in\mathscr{I}_c}\sum_{n\in\mathscr{N}\setminus\mathscr{N}_c} y_{in} + \sum_{i,j\in\mathscr{I},i\neq j} h_{ij}P(\delta_{ij}), \tag{23}$$

$$h_{ij} \leq z_i, \qquad \forall i \in \mathcal{I}, \tag{24}$$

$$h_{ij} \geq z_i + z_j - 1, \qquad \forall (i,j) \in \mathcal{I}, \tag{25}$$

$$h_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{I}. \tag{26}$$

Lastly, we introduce a measure to showcase the difference between found sets of optimal prototypes. This measure consists of the summed distances between all individuals in the optimal set. The dissimilarity matrix δ contains the distances between all individuals, including the prototypes. It is important to note that during the calculation of this dissimilarity matrix, the variables are scaled so that no single variable contributes disproportionately to the distance.

Given the dissimilarity matrix δ, we can define a pairwise distance between the prototypes in the optimal set which we name *Total Dissimilarity*. We denote the set of selected prototypes as $\mathscr{O}$. The measure is defined as:

$$\text{Total Dissimilarity } = \sum_{i=1}^{\mathscr{O}}\sum_{j=i+1}^{\mathscr{O}} d_{ij}.$$

5 Numerical results

In this section, we elaborate on the results we find. Firstly, we try to replicate the results that were originally presented for the covering and partitioning model for the indicated datasets (Carrizosa et al., 2022) and we discuss the differences we obtain. Then, we consider the improvements we find on the interpretability by adding the distance terms in the objective function. The numerical results section is separated on datasets. Hence we discuss the Canadian weather dataset first, then we discuss the simulated data instances and lastly, we elaborate on the findings for the marketing dataset.

To solve the mathematical optimisation models, we use Gurobi (version 11.0.2) with Python (version 3.12.0) on a MacBook Pro 2020 with an Apple M1 chip and 8GB of memory. We implement the models in the same way as done in the original work, ensuring that we can either prove optimality or demonstrate infeasibility within a reasonable time frame.

5.1 Canadian weather data

The results that we find for the Canadian weather dataset are similar to the results reported in the original work. We follow the same procedure that is described, but still get minor variations on the numeric values of TPR and FPR compared to the original work. For the example used in the original work where $\lambda_c = 0.8$ and $\mu_c = 0.2$ we obtain the same values for TPR and

FPR. Furthermore, the selected prototypes for the parameters set at these values are *Charlottvl, Uranium City, Victoria* and *Resolute* which are the same prototypes that are selected in the original work.

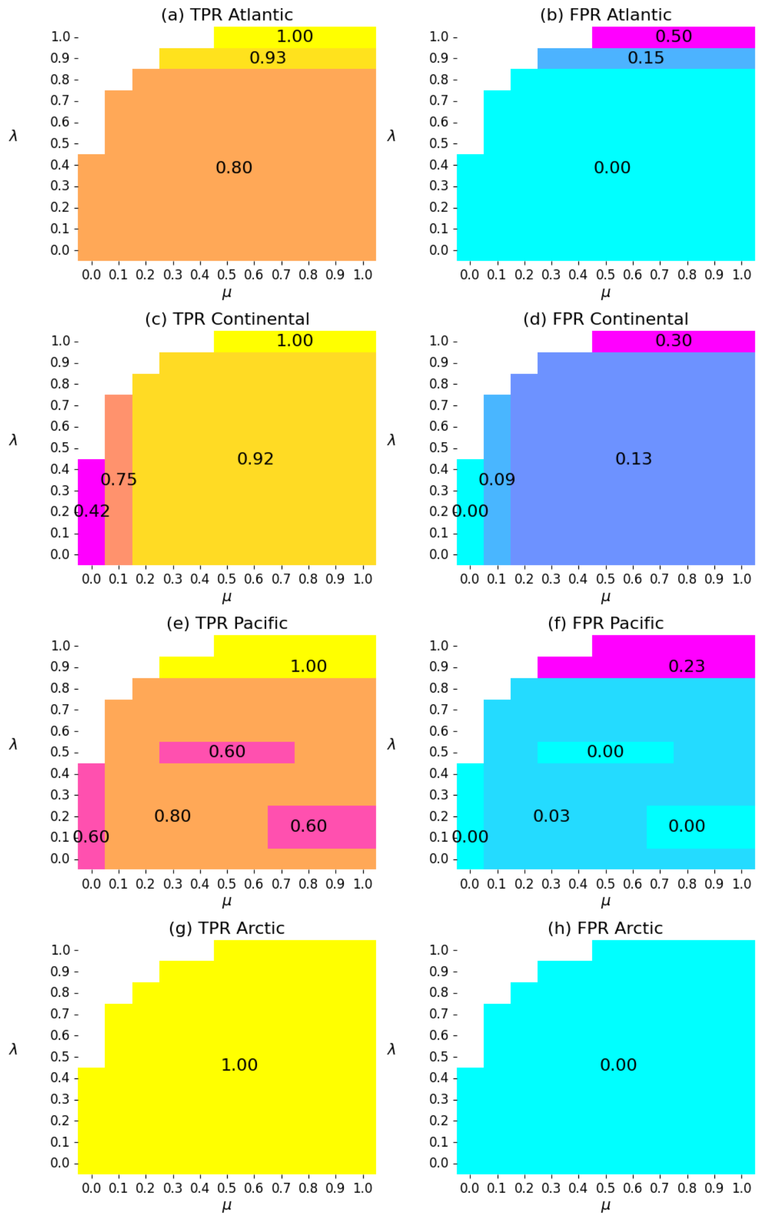

Figure 3: The results for the replication of the covering model. The y-axis and x-axis represent the variation of λ_c and μ_c both variables range from $[0, 1]$. The white background represents that the model is infeasible. Every graph denotes the TPR (left-side) or the FPR (right-side) for each of the 4 clusters.

For the Atlantic and Arctic cluster our results do not deviate from the original results as can be seen in figure 3. The Continental and Pacific cluster however have minor deviations from the original results. For the Continental cluster we observe that for the values of $\mu = 0.2$ the TPR is equal to 0.75 instead of 0.67. This indicates that our implementation actually performed slightly better in classifying individuals to the correct prototypes, we do however also see a slight loss for the FPR in the same cluster given that also for the same value of μ the value increases from

0.04 to 0.09. The biggest differences occur for the Pacific cluster. We see that the output is mostly the same, but the area where the TPR and FPR change from 0.8 to 0.6 or from 0.0 to 0.03 is different.

As we implement the model exactly as stated in the original work, the origin of the found differences are not immediately clear. We do however think that a difference in optimisation settings in the used software could be one of the explanations as well as the versions of the software used. Furthermore, we find that not all of the modelling choices are clearly documented, making it harder to exactly replicate the results. The main reason for differing results could be due to a slight difference in the dissimilarity matrix that was used to calculate the distances between individuals. Especially, the rounding of variables could play a role as this influences the optimal solution if changes are big enough.

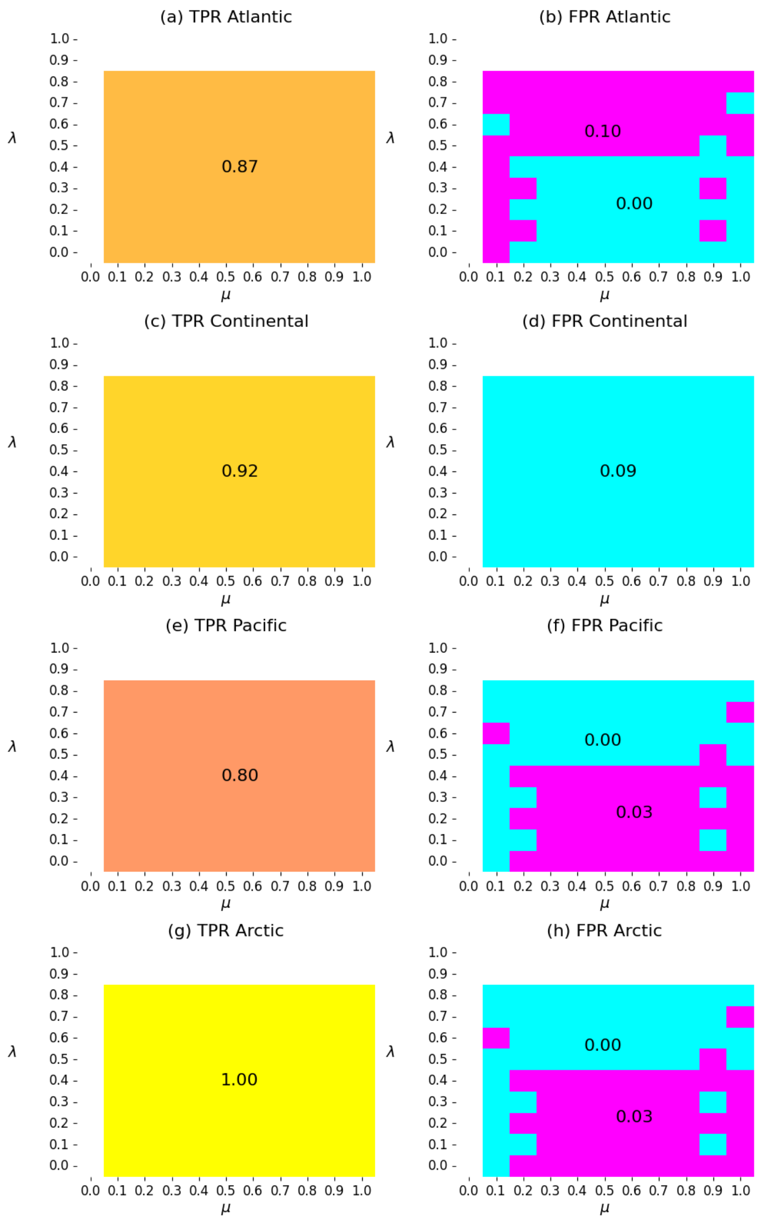

Figure 4: The results for the replication of the partitioning model. The y-axis and x-axis represent the variation of λ_c and μ_c both variables range from $[0, 1]$. The white background represents that the model is infeasible. Every graph denotes the TPR (left-side) or the FPR (right-side) for each of the 4 clusters.

For the partitioning model we obtain similar results to the original work for the TPR for all 4 clusters, as can be seen in figure 4. For the FPR however, we only get identical results for the Continental cluster. For the other 3 clusters we see a similar pattern as to where the graph differs from the original results. Figure 3f and 3h show exactly the same pattern for a differing value from 0.03 to 0.0, whereas 3b shows the inverse pattern for a FPR of 0.1 instead of 0.0. As we use the same software and the same dissimilarity matrix for the partitioning model as for the covering model. The reasons for difference in results for the covering model can be applied here too. What is notable here, is that the TPR results are the same while FPR results differ. This can be interpreted as our model being consistent with the original results for individuals that are close to the prototype, but inconsistent for individuals that are just on the boundary of being the closest to one prototype or another. That is why especially in this case the different rounding in the calculation of the dissimilarity matrix, could play a role in the difference in results.

5.2 Simulated data

In this section we elaborate on the results that we find for running the covering and partitioning model on the simulated dataset. Firstly, we must further specify the parameter values and the data reduction technique that we implement from the original work. The parameter values that control the TPR and the FPR are on a smaller grid namely: $\lambda \in \{0.85, 0.86, .087, 0.88, 0.89, 0.90\}$ and $\mu \in \{0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$. We choose this smaller interval because we want to verify whether the model works properly for relevant values, rather than verifying its workings for all parameter values. To apply the reduction technique, we firstly apply hierarchical clustering with Euclidean distance as a dissimilarity measure between the individuals. We do this for the entire dataset per cluster c and from $\mathscr{N}_c$ we create $\widetilde{\mathscr{N}}_c$. We then choose a threshold that yields exactly $\left|\widetilde{\mathscr{N}}_c\right|$ groups of individuals. From these $\left|\widetilde{\mathscr{N}}_c\right|$ groups, we choose one representative with a weight $\tilde{w_n}$ that corresponds to the number of individuals in its group. All selected individuals together then form $\widetilde{\mathscr{N}}$. We use a similar approach to reduce the set of prototypes. Again, we firstly use hierarchical clustering to form $\widetilde{\mathscr{I}_c}$ from $\mathscr{I}_c$. We then split $\widetilde{\mathscr{I}_c}$ into $\left|\widetilde{\mathscr{I}_c}\right|$ groups of prototypes. From these groups we select a prototype that then becomes a member of $\widetilde{\mathscr{I}_c}$.

Following the original work, we choose the instances of the prototypes and the individuals to be $\left|\widetilde{\mathscr{N}}_c\right| = 125$ and $\left|\widetilde{\mathscr{I}_c}\right| = 25$. In figure 5, the results can be found for the covering model and in figure 6 for the results for the partitioning model.

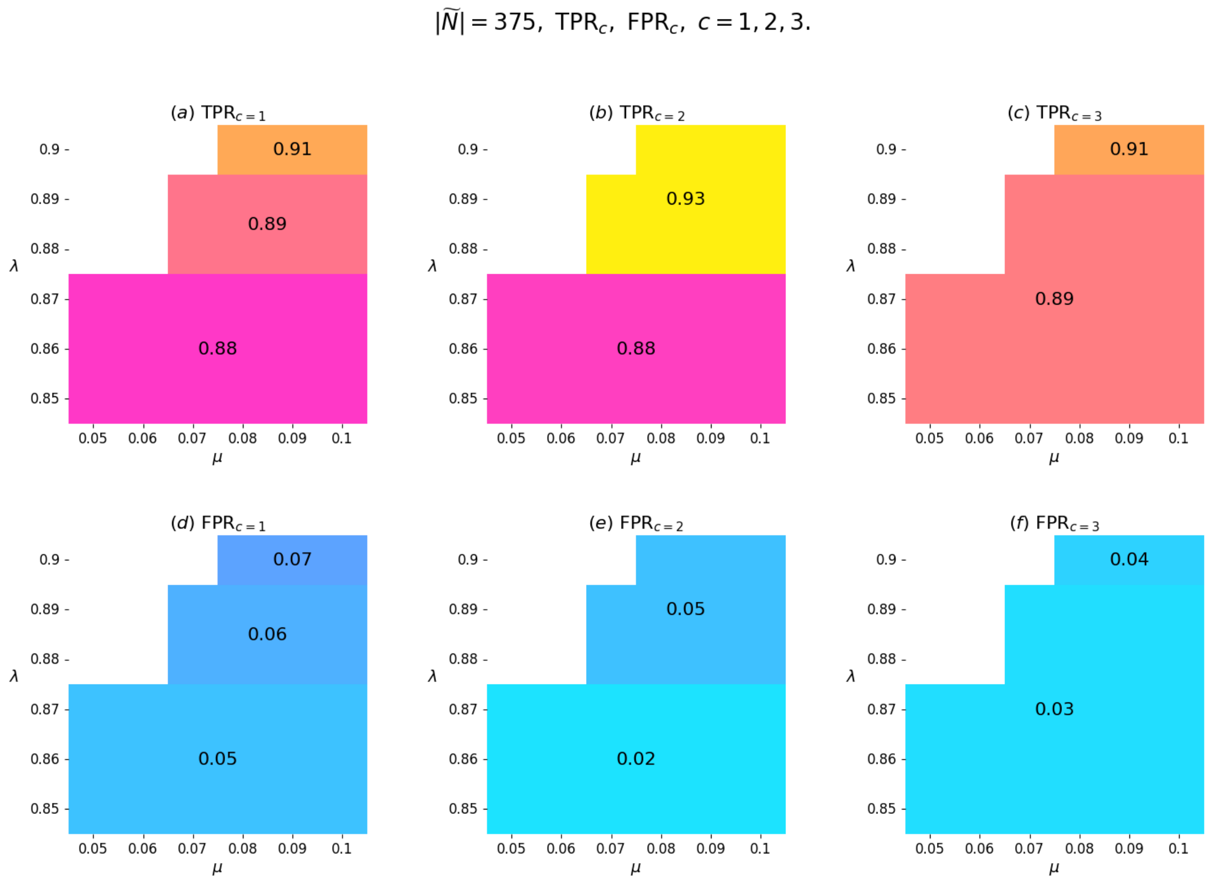

Figure 5: These results demonstrate how the covering model's performance in terms of TPR and FPR changes across different parameter settings for the reduced simulated data. The parameters vary from $\lambda_c \in [0.85; 0.9]$ and $\mu_c \in [0.05; 0.1]$. The white background represents that the model is infeasible.

Looking at figure 5, we again see that we obtain largely the same results. For all graphs the region of infeasibility is slightly smaller than in the original results. We also see that although we observe a similar trend in the rise of both TPR and FPR as λ increases, the values for TPR and FPR do not exactly correspond to the values presented in the original work. One of the main reasons as to why the results do not exactly match is that the seed of the generated data points is not given. This makes it impossible to generate exactly the same points and consequently causes the dissimilarity matrix to be different. Through the difference in dissimilarity and the difference in original points the optimal solution can change. Similarly, the data reduction technique can affect the results. Within this approach there are two points of uncertainty, one being the random selection of individuals and prototypes and the other being the associated weights passed on to the individuals. Both of these events lead to different output, this could lead to a larger feasible region and it can also be the main cause of the numerical results to not match one to one.

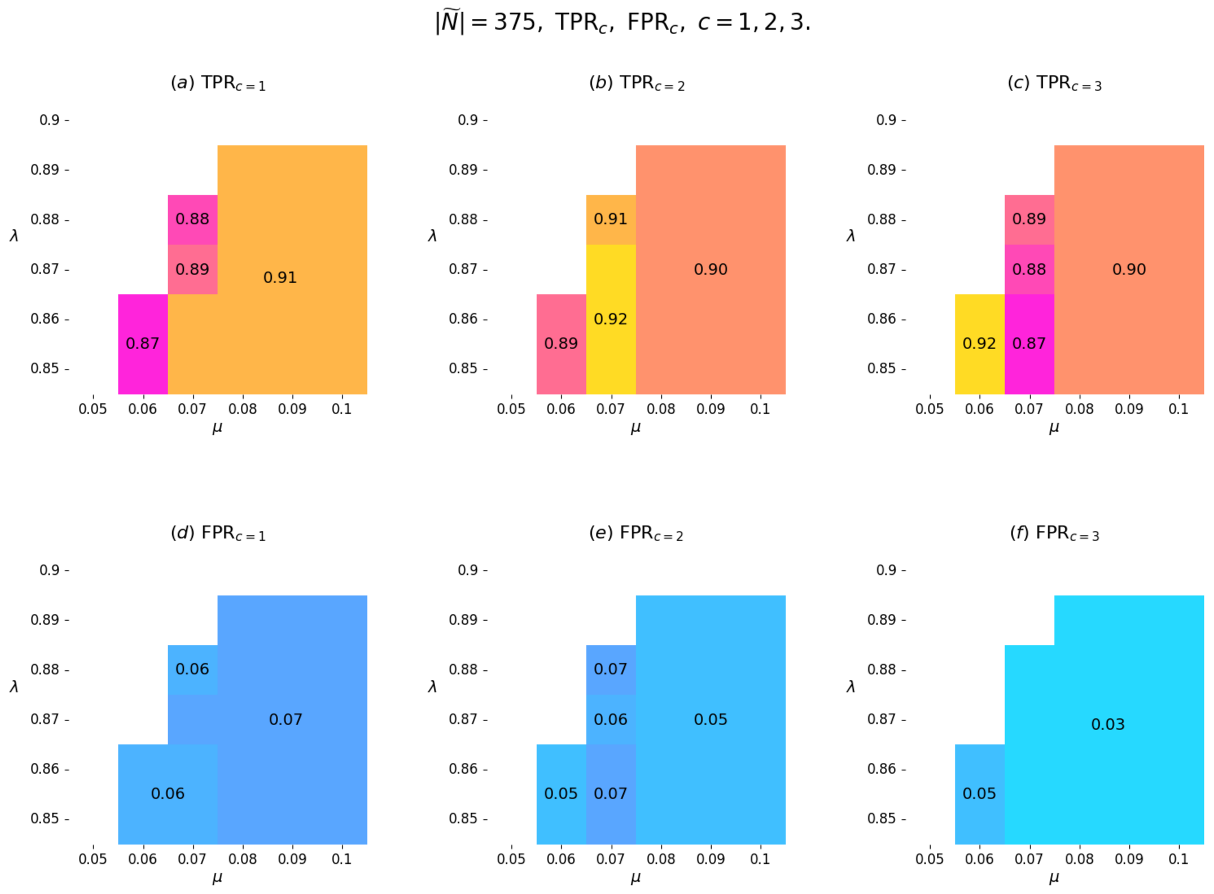

Figure 6: These results demonstrate how the partitioning model's performance in terms of TPR and FPR changes across different parameter settings for the reduced simulated data. The parameters vary from $\lambda_c \in [0.85; 0.9]$ and $\mu_c \in [0.05; 0.1]$. The white background represents that the model is infeasible.

In figure 6 we see similar differences to the original results as in figure 5. The feasibility region is bigger and the values are similar in pattern but not exactly the same as reported in the original work. Again, this is most likely largely due to the randomness used in the simulated data.

5.3 Notes on the replication

We attempt to replicate the larger instances of the reduction method described in the original work, which involves constructing and solving models with $10^4, 10^4, 10^6$ individuals. However, our replication efforts are unsuccessful due to a memory error encountered when constructing the dissimilarity matrix. The dissimilarity matrix, which is needed for the hierarchical clustering process, requires significant computational power. This high memory demand exceeds the capacity that is at our disposal, leading to failures during the matrix construction phase. We are therefore unable to replicate all figures from the original work.

5.4 Distance models

In this section, we show the results we obtain for the covering model when implementing the extra terms that we propose in section 4 on the mall customer dataset.

We choose to take all individuals to be eligible as prototypes to extend the choice of possible prototypes. We implement the models by adding the distance terms and run the model for an example case of $\lambda = 0.8$ and $\mu = 0.2$. We take $\alpha = 100$ and $\beta = 10$. To be able to see the effect of our extra terms we introduce a benchmark model. This model is the standard covering model as it is described in 4.1.

Cluster	TPR_b	TPR_{IP}	TPR_{SD}	FPR_b	FPR_{IP}	FPR_{SD}
0	0.89	0.93	0.93	0.00	0.02	0.04
1	0.84	0.84	0.81	0.03	0.035	0.11
2	0.93	0.93	0.80	0.04	0.04	0.07
3	1.00	1.00	0.95	0.00	0.00	0.06
4	0.80	0.80	0.83	0.01	0.02	0.14

Table 1: This table shows the difference in TPR and FPR across three different models per cluster. b represents the benchmark model, IP represents the inverse penalty model and SD represents the squared distance model.

Table 1 shows that the inverse penalty model slightly improves the categorisation of individuals, with the TPR_{IP} of cluster 0 increasing by 0.04 compared to TPR_b, while the other clusters remain unchanged. In contrast, the TPR_{SD} decreases for three clusters but shows slight improvements for the other two, with the largest loss in accuracy of 0.13 observed in cluster 2. The FPR shows a similar pattern of variation between the two models. Specifically, the FPR_{IP} is marginally higher for three clusters, whereas the FPR_{SD} significantly increases across all clusters.

Model	**Chosen Prototypes**	**Total Dissimilarity**
Benchmark	[57, 39, 140, 192, 68]	35.943
Inverse Penalty	[10, 39, 140, 192, 41]	39.316
Squared Distance	[8, 7, 12, 198, 185]	44.941

Table 2: This table shows the total dissimilarity between the chosen prototypes for each model. In the column of chosen prototypes the difference in selected prototypes can be observed. The numbers correspond to the indices of the 200 prototypes in the mall customer dataset.

Table 2 further highlights the differences between the models. The benchmark model achieves a total dissimilarity of 35.943. In comparison, the inverse penalty model's dissimilarity increases to 39.316, with two out of five prototypes chosen differently, therefore showing a 9.4% increase in total dissimilarity. Despite this higher dissimilarity, the TPR and FPR remain similar to the benchmark, indicating that the selected prototypes are farther apart, and therefore become more distinct and interpretable, while maintaining high TPR and low FPR.

For the squared distance model, the total dissimilarity increases by 25.0% to 44.941. Although this model shows a slight decrease in TPR and an increase in FPR, the obtained difference in solution is significant, therefore showing a potentially better suited optimal set for interpretation. The model also demonstrates a trade-off between prototype interpretability and classification performance. The results suggest that it is possible to identify a set of optimal prototypes that are more widely separated while still meeting the constraints defined by the λ and μ values, balancing interpretability and performance effectively.

6 Conclusion

To improve the interpretability of CA, we propose a new formulation of the covering model initially suggested by (Carrizosa et al., 2022). By introducing additional penalty terms, specifically the inverse distance penalty and the squared distance term, we aim to increase the distance between selected prototypes, making the clusters more distinct and easier to interpret. Using a Canadian weather dataset, a simulated dataset, and a mall customer dataset, we demonstrate the effectiveness of our approach.

For the inverse penalty model, we achieve a 9.4% increase in the distance between selected prototypes compared to the benchmark solution. Additionally, we observe a slight improvement in the TPR for certain clusters while maintaining a comparable FPR, indicating equal categorisation quality without losing overall accuracy. The squared distance model, despite showing some decrease in TPR for certain clusters, significantly increases the total dissimilarity between prototypes by 25.0%, enhancing interpretability at the cost of a slight reduction in classification performance as showed by the increase in FPR and the slight decrease in TPR.

In future work, we aim to consider a wider scope of parameter values for λ and μ to evaluate our proposed method's performance under different restrictions. Additionally, we would like to introduce a fine-tuning approach for selecting α and β to generalise our methodology to other datasets.

References

Bertsimas, D., Orfanoudaki, A. & Wiberg, H. (2021, jan). Interpretable clustering: an optimization approach. *Mach. Learn.*, *110*(1), 89–138. doi: 10.1007/s10994-020-05896-2

Carrizosa, E., Kurishchenko, K., Marín, A. & Morales, D. R. (2022). Interpreting clusters via prototype optimization. *Omega*, *107*.

Chen, J., Chang, Y., Hobbs, B., Castaldi, P., Cho, M., Silverman, E. & Dy, J. (2016). Interpretable clustering via discriminative rectangle mixture model. In *2016 ieee 16th international conference on data mining (icdm)* (p. 823-828). doi: 10.1109/ICDM.2016.0097

Chen, V., Li, J., Kim, J. S., Plumb, G. & Talwalkar, A. (2022). Interpretable machine learning: Moving from mythos to diagnostics. *Queue*, *19*(6), 28–56. doi: 10.1145/3511299

Choo, J. & Liu, S. (2018). Visual analytics for explainable deep learning. *IEEE Computer Graphics and Applications*, *38*(4), 84-92.

Davidson, I., Gourru, A. & Ravi, S. (2018). The cluster description problem-complexity results, formulations and approximations. *Advances in Neural Information Processing Systems*, *31*.

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).

Fortet, R. (1960). Applications de l'algebre de boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, *4*(14), 17–26.

Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems* (p. 4768–4777). Curran Associates Inc.

Ma, C. & Wu, J. (2007). *Data clustering: Theory, algorithms, and applications.* doi: 10.1137/1.9780898718348

MacQueen, J. et al. (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).

Mittal, H., Pandey, A. C., Saraswat, M., Kumar, S., Pal, R. & Modwel, G. (2022). A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. *Multimedia Tools and Applications*, 1–26.

Nugent, R. & Meila, M. (2010). An overview of clustering applied to molecular biology. *Statistical methods in molecular biology*, 369–404.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 1135–1144). Association for Computing Machinery.

Rousseeuw, P. J. & Kaufman, L. (2009). *Finding groups in data: An introduction to cluster analysis.* John Wiley & Sons Hoboken, New Jersey.

Sneath, P. H. (2005). Numerical taxonomy. In *Bergey's manual® of systematic bacteriology* (pp. 39–42). Springer.

A Programming code

In the provided programming code, all python and R files can be found that were used to obtain the results stated in this thesis. The files can be divided into 3 categories per dataset used, and then furthermore divided into two by the kind of model used. One of the models being the covering and the other one being the partitioning. The code contains some inline comments for clarification.