DEGREE PROJECT IN THE FIELD OF TECHNOLOGY
ENGINEERING PHYSICS
AND THE MAIN FIELD OF STUDY
COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Unsupervised learning with mixed type data

for detecting money laundering

SARA ENGARDT

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Unsupervised learning with mixed type data

for detecting money laundering

SARA ENGARDT

Degree Programme in Machine Learning
Date: June 13, 2018
Supervisor: Johan Gustavsson
Examiner: Örjan Ekeberg
Swedish title: Klusteranalys av heterogen data
School of Electrical Engineering and Computer Science

Abstract

The purpose of this master's thesis is to perform a cluster analysis on parts of Handelsbanken's customer database. The ambition is to explore if this could be of aid in identifying type customers within risk of illegal activities such as money laundering.

A literature study is conducted to help determine which of the clustering methods described in the literature are most suitable for the current problem. The most important constraints of the problem are that the data consists of mixed type attributes (categorical and numerical) and the large presence of outliers in the data. An extension to the self-organising map as well as the k-prototypes algorithms were chosen for the clustering.

It is concluded that clusters exist in the data, however in the presence of outliers. More work is needed on handling missing values in the dataset.

Sammanfattning

Syftet med denna masteruppsats är att utföra en klusteranalys på delar av Handelsbankens kunddatabas. Tanken är att undersöka ifall detta kan vara till hjälp i att identifiera typkunder inom olagliga aktiviteter såsom penningtvätt.

Först genomförs en litteraturstudie för att undersöka vilken algoritm som är bäst lämpad för att lösa problemet. Kunddatabasen består av data med både numeriska och kategoriska attribut. Ett utökat Kohonen-nätverk (eng: self-organising map) samt k-prototyp algoritmen används för klustringen.

Resultaten visar att det finns kluster i datat, men i närvaro av brus. Mer arbete behöver göras för att hantera tomma värden bland attributen.

Contents

Chapter 1

Introduction

1.1 Context

1.1.1 Money laundering

Money laundering aims to hide the connection between property and crime and therefore give the (false) impression that the property has been earned in a legal way. Traditional ways of laundering money include false invoices, investments in foreign real estate and gambling sites [1]. Money laundering is linked to virtually all criminal activities generating criminal proceeds, and it sustains and contributes to the growth of criminal markets across the EU [2]. By the Swedish money laundering law, banks operating in Sweden are obligated to assess and follow up the risks of being used for money laundering [3]. This includes getting information about their customers that are believed to be relevant with regards to money laundering risks.

1.1.2 Machine learning and cluster analysis

Machine learning is the group of algorithms that learn from data. It can be either supervised (the data has labels and the algorithm can learn correct classifications), unsupervised (no labelled training data exists, the algorithm can only learn from inherent structures in the data) or reinforcement learning (the algorithm gets a "reward" when it goes in the right direction). Clustering is an important unsupervised learning technique that aims to organise instances such that similar objects are put in the same cluster and dissimilar objects are separated

to different clusters [4]. Different methods for clustering have been studied for a long time, for example the k-means algorithm that was proposed already in the 1960s [5]. Since then a variety of algorithms have been proposed on how to cluster a dataset in the best way, see for example [6] and chapter 2 for a review.

For a long time, the literature focused on clustering algorithms that only handled either numerical or categorical variables. A numerical variable is typically measured in some number or value, corresponding to the quantity of the attribute. A natural ordering exists between different values of the attributes, they can be more or less similar to one another. Examples of numerical variables are age, salary and time. In contrast, categorical variables take on a value that is one out of several possible categories. It can be in the form of text or numerical, but in general there is no natural order in the values. Such an attribute could be gender, zip-code or favourite food. Categorical attributes can have order encoded in them, in this case they are called ordinal variables. An example of an ordinal variable is student grades, if A is the highest grade and E is the lowest, then B is more similar to A than to E.

Modern, real-world datasets can be very large and noisy, and often contain data with mixed type attributes (i.e. numerical and categorical). New algorithms to cluster mixed type data are continuously being proposed, from early work such as k-prototypes [7], to a Bayesian approach [8] or a Neural Network approach such as extended self-organising maps, for example [9]. However, there is no consensus in the field which way to solve the problem.

1.2 Problem specification

1.2.1 Specified problem definition

This thesis was written at Handelsbanken, one of Scandinavia's largest banks with businesses in over 20 countries. The aim of the project was to explore whether the use of cluster analysis on their customer database could identify type customers within risk of illegal activities such as money laundering. The database contains mixed type data, and the prior belief was that it was very noisy. This data was used as input for the cluster algorithm that was chosen during the literature study. Only features of the data that are considered important with regards to money laundering was entered into the clustering al-

gorithm (this assessment was done by an expert group at Handelsbanken). Such features could for example be age, gender and account balance.

1.2.2 Challenges

Aside from being an unsupervised learning problem, the dataset imposes the following constraints on the algorithm (with decreasing significance):

- Mixed type attributes: the data constitutes a mix of numerical and categorical data (e.g. age and gender).
- Noisy data: it is believed that the data contains points that do not belong to any cluster. These points could also be of interest because of the dissimilarity and should be treated wisely.
- Clusters of arbitrary shape: Many algorithms assume clusters of convex (spherical) shape, but there is no reason to believe this would be the case in this dataset.
- Large dataset: The dataset is relatively large ($\sim 10^5$ data points, $\sim$ 20-30 features), which puts some constraints on speed and memory.

1.2.3 Research question

There are two main tasks involved in this degree project, one is to find the most appropriate algorithm given some criteria (as described below), and the second is to implement and evaluate the chosen algorithm.

As the project involves two phases, choosing and implementing some clustering algorithm, there are two separate research questions for the two tasks:

What unsupervised learning algorithm is, according to the literature, best suited to solve the clustering problem specified by the criteria listed above?

Will the chosen clustering algorithm yield better in-cluster compactness when performed on the customer database than when performed on random data?

1.2.4 Objective

The goal of the degree project is to perform an exploratory investigation of a selection of features in the banks customer database using cluster analysis. The author wishes to investigate whether it is at all possible to find distinct clusters using an unsupervised learning technique, and in that case evaluate the chosen method and possibly compare different methods for clustering.

1.3 Sustainability and ethics

When using sensitive, personal data such as what has been used in this thesis, one must always act with care. The data used has been anonymised, and no sensitive information will be published. This is to ensure that the integrity of the customers is kept intact.

Using machine learning methods to determine a person's or company's culpability, or level of risk, could be a great way to ensure fair judgement. An algorithm will make the same choice for all given the same variables without bias, in contrast to a human. However, when developing algorithms and choosing what features are of importance, there is always a risk of building our own biases into the algorithm. We as humans, and our society, are biased and have prejudice and will thus forward those prejudice. If we only look at a person as a sum of their features connected to risk of crime, some will be unfairly judged just for coming from a certain background and some will unfairly walk free. It should never be up to only a machine learning algorithm to decide how someone is judged, human compassion will always be needed as well.

Money laundering is a big problem in the modern society and working against it is very important for our economical sustainability. With technology becoming more sophisticated, so are the methods criminals use to laundry their money. It is very important that anti-money laundering techniques follow, or preferably precede, criminals in using new techniques such as machine learning to fight these activities.

Chapter 2

Background

2.1 Applications of cluster analysis

Cluster analysis has been successful in a number of applications such as image segmentation [10], clustering gene expression data to understand previously unknown functions of genes [11] or in market research to segment the market and determine target customers [12]. Previous applications of machine learning in the bank sector have mainly focused on supervised tasks such as credit risk evaluation, for example [13], or unsupervised transaction monitoring to detect money laundering, for example [14]. In [15], Alexandre and Balsa try to find customer profiles for anti-money laundering systems, similar to this project. They use the k-means algorithm for initial exploration of the dataset, resulting in seven clusters that are deemed relevant by a domain expert. They thereafter move on to a supervised task, trying to learn classification rules, using the clusters as labels. In the article they use 1-hot-encoding to handle categorical data (see section 2.2.3). As discussed in section 2.2.3, this method suffers from drawbacks such as increase in dimensionality and loss of semantics and can yield poor results. This is noticed also in [15], as the results are clearly better for the numerical attributes than for the categorical. The current project aims to use more sophisticated methods for clustering mixed type data. Apart from [15], we are not aware of any previous studies concerning unsupervised learning for identifying risk-customers within money laundering. One reason for the limited literature could be due to secrecy, that work has been done but not published.

2.2 Cluster analysis

2.2.1 Distance measures

Almost all clustering algorithms use distance measures to find clusters in a dataset. The following sections provide a brief introduction to commonly used distance measures for different types of data. Unless other is stated, x and y are two $d-$dimensional points.

Measures for numeric data

The Euclidean distance is probably the most used measure for numeric data (for example [5]). It is defined as:

$$d_{Euclidean} = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} \tag{2.1}$$

The Manhattan distance, or the "city block-distance" [4] is the sum of the difference of all attributes:

$$d_{Manhattan} = \sum_{j=1}^{d} |x_j - y_j| \tag{2.2}$$

The Minkowski distance [4] is a generalisation of both the Euclidean and the Manhattan distance. It is defined as:

$$d_{Minkowski} = \left[\sum_{j=1}^{d} |x_j - y_j|^r \right]^{\frac{1}{r}}, r \geq 1 \tag{2.3}$$

Measures for categorical data

The simple matching coefficient [7] simply counts the number of matching attributes between two points as follows:

$$d_{simple\_match} = \sum_{j=1}^{d} \frac{(n_{x_j} + n_{y_j})}{n_{x_j} n_{y_j}} \delta(x_j, y_j) \tag{2.4}$$

where n_{x_j} and n_{y_j} are the numbers of objects in the dataset that have categories x_j and y_j for attribute j, respectively and

$$\delta(x, y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$$

The category utility (CU) function was originally proposed by [16] as a way to mathematically express which features best describe a category. It has since been used as a similarity measure for clustering categorical data, for example by [17]. Let k be the number of classes, I the number of attributes and J the number of values, $P(A_i = V_{ij}|C_k)$ the conditional probability that attribute i has the value V_{ij} given cluster C_k, and $P(A_i = V_{ij})$ the overall probability of the attribute i having values V_{ij} in the entire dataset. Then the category utility function can be written:

$$CU = \frac{\sum_{k=1}^{K} P(C_k) \sum_{i=1}^{I} \sum_{j=1}^{J} [P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2]}{K} \quad (2.5)$$

The higher the CU-value, the higher the similarity.

A distance hierarchy [18], [19], [20] takes the semantics between categorical attributes into account. Consider for example the tree attributes `<Carrot, Cucumber, Pizza>`. If these were answers on one's favourite food, intuitively `Carrot` and `Cucumber` should be closer since they are both vegetables. The distance between any two attributes is then the shortest path in the tree, meaning that:

$$d(Carrot, Cucumber) = d(Cucumber, Carrot) = 2$$

but we also get that:

$$d(Carrot, Pizza) = d(Cucumber, Pizza) = 3$$

in the small example in Figure 2.1. The distance hierarchy of the three variables is visualised in a tree as follows:

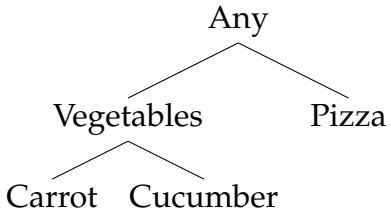


Figure 2.1: Distance hierarchy between the categorical values `Carrot`, `Cucumber`, `Pizza`. Here `Carrot` and `Cucumber` are closer to each other than `Pizza` since they are both vegetables. In this example the edges of the tree have unit weights, but any weights can be assigned if appropriate

Some authors have used entropy as a metric for categorical data. It can be used to measure the similarity of attributes within a cluster, and is defined as follows:

$$H(X) = -\sum_{x \in X} P(x) log_2(P(x)) \tag{2.6}$$

where X is a categorical attribute taking different values x, and $P(x)$ is the probability of value x.

Mixed type measures

Gower proposed a distance measure for mixed type data as mentioned in [4]. The distance between x and y can be defined as:

$$d_{gower}(x, y) = \left(\frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k) d^2(x_k, y_k) \right)^{\frac{1}{2}} \tag{2.7}$$

where $w(x_k, y, k)$ equals one if a comparison is valid for the k:th attribute of the data points, and zero if there are missing values. $d^2(x_k, y_k)$ is a squared distance component for the k:th component and is defined differently for different data types. For numeric data, let R_k be the range of the k:th attribute. Then,

$$d(x_k, y_k) = \frac{|x_k - y_k|}{R_k} \tag{2.8}$$

For categorical data, the simple matching coefficient as defined previously is used. A similarity measure can be defined correspondingly.

2.2.2 Traditional clustering algorithms

Traditionally, clustering algorithms can be partitioned into four main groups: partitional, hierarchical, density-based and grid clustering [4], [6]. This section discusses the general properties and some widely used algorithms for of these types of methods, as well as some other notable approaches.

Partitional Clustering

In partitional clustering (sometimes referred to as centre-based clustering), one regards the centre of some data points as the centre of the corresponding cluster. For numeric data, the centre of the data points is usually the arithmetic mean. k-means [5] is one of the most famous clustering algorithm of this type, where the basic idea is to iteratively update the centre of the clusters to better fit the given dataset. This process is continued until some criteria for convergence is met, usually minimising the least square error of the Euclidean distance. See Figure 2.2 for an example. The concept of a cluster centre means that one can only find spherical clusters using these types of algorithms. They are in general also quite sensible to noise. A great advantage of partitional clustering algorithms is that they have a low computational time in general, k-means is linear in time, $\mathcal{O}(n)$, where n is the number of data points.

Hierarchical Clustering

The general idea in hierarchical clustering algorithms is to cluster data points based on some hierarchical relationship. Two types of hierarchical clustering techniques exist, namely *agglomerative* and *divisive*. The agglomerative approach is a bottom-up approach where every data point starts in its own cluster, and larger clusters are formed by merging neighbouring clusters together. In the divisive approach one starts with all data points in the same cluster, and then splits the data points that are most dissimilar into different clusters. BIRCH [21] and ROCK [22] are examples agglomerative clustering algorithms. In BIRCH a feature tree, called CF-tree, is dynamically built to summarise the dataset. After the CF-tree is built, an agglomerative hierarchical algorithm is applied directly to the nodes in the CF-tree. ROCK is an agglomerative clustering algorithm suitable for categorical data. A

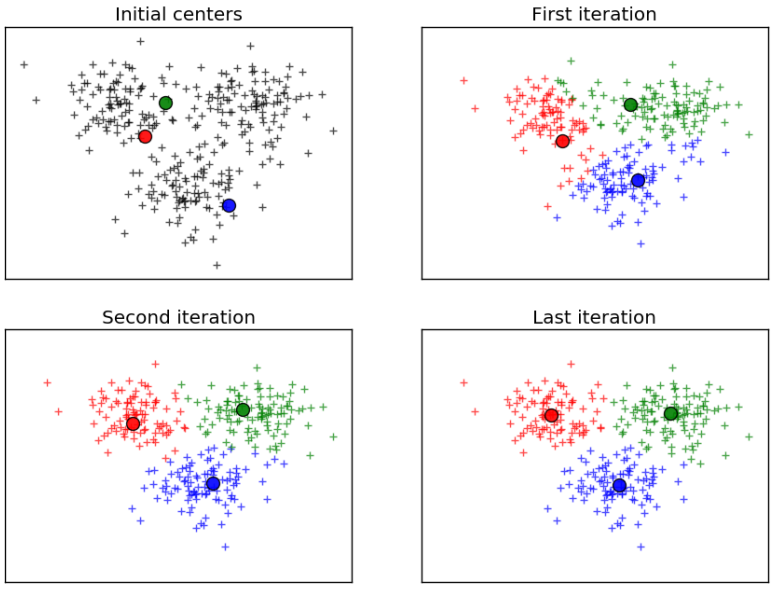


Figure 2.2: Small example of the k-means algorithm. Three data points are randomly assigned as initial cluster centres (first image), corresponding to the red, green and blue dot. At the first iteration, all data points are assigned to the cluster that corresponds to the cluster centre that is closest to the data point at hand. Each cluster centre is then moved to the centre of cluster. This is then repeated until no points are reassigned and the algorithm is said to have converged.

link-based similarity measure is used to measure the distance between clusters and data points, see section 2.2.1. BIRCH is an example of a highly scalable hierarchical algorithm having a computational complexity of $\mathcal{O}(n)$, but hierarchical algorithms in general have a high time complexity. ROCK has a time complexity of $\mathcal{O}(n^2 log(n))$. Hierarchical clustering algorithms can detect clusters of arbitrary shape and are in general robust to noise.

Density-Based Clustering

The idea for this type of clustering algorithm is that a cluster is defined by a higher density of data points than their surroundings. A typical

density-based algorithm is DBSCAN [23] which is based on the concept of density-reachability. A point q is density-reachable from p if they are within a distance ϵ from each other, and there are sufficiently many points ($minPts$) surrounding p so that a cluster can be formed including p and q. Two parameters have to be set by the user, ϵ (maximum distance between two points) and $minPts$ (minimum points to form a cluster). These parameters can be hard to find which is one major drawback of this algorithm. Advantages of this approach is that clusters of arbitrary shape can be found, and that it is robust to noise. DBSCAN have a medium time complexity of $\mathcal{O}(n * log(n))$ which is representative of density-based clustering methods.

Grid-Based Clustering

Grid-based clustering is based on dividing the original data space into a grid structure of some coarseness. This makes grid-based clustering techniques very fast (linear in the number of data points) and scalable, given a coarse granularity. It can however be hard to find the appropriate granularity. STING [24] is an example of a grid-based algorithm, that divides the feature space into rectangular units by constructing hierarchical structures. Data with different structure levels are then clustered respectively. STING can find clusters of arbitrary shape and it is robust to outliers.

Other Methods

Another notable approach is clustering methods based on statistical models. In these types of algorithms, one assumes that the dataset is generated by some underlying statistical distribution, and then tries to fit the data to the chosen distribution. Different clustering methods based on statistical models have been proposed, for example The Infinite Gaussian Mixture Model [25], or COBWEB [26]. Other modern clustering algorithms include *subspace clustering* and *ensemble clustering*. In subspace clustering, different dimensions can have a varying impact on different clusters. It is suitable for high-dimensional data [27]. In ensemble clustering multiple clustering algorithms collaborate to make the final partitioning [28]. Clustering with neural network methods have also become more popular, for example self-organising maps (SOM) [29] or adaptive resonance theory (ART) [30]. A neural network is a collection of connected units (inspired by neurons in

a brain), where the units and the connections adapt to data they are exposed to. Self-organising maps can also be used to visualise high dimensional data by projecting them onto a two-dimensional map. In general, neural network methods are computationally expensive, but are robust against outliers and can handle clusters of arbitrary shape.

Common for all algorithms mentioned above are that they are only suitable for single-type data (mostly numerical data). After their initial proposal, many algorithms have been extended and modified to work for heterogeneous data with varying success. The next chapter describes such extension and other algorithms that cluster mixed-attribute data.

2.2.3 Clustering mixed attribute data

Many approaches to handle mixed attribute data have been proposed. Early approaches can be summarised into two categories:

- Transform categorical values to a set of binary, numeric values and then apply regular numeric distance measures, the so called 1-of-k or 1-hot-encoding (used for example in [15]). The transformation suffers from an (in some cases extreme) increase in dimensionality, as well as loss of semantics in categorical variables. In addition, Guha et al. [22] shows that the Euclidean distance can be a bad similarity measure for categorical attributes when the domain is large.
- Discretise numerical attributes in categories, and then cluster as a categorical dataset (for example [5] or [21]). This poses difficult problems regarding how to discretise numeric features. A boundary issue also arises as two similar numeric values might be partitioned into different bins.

Using the above techniques any clustering algorithm can be applied to a mixed dataset, but as discussed it causes severe problems. New ideas have been proposed that overcome these shortcomings. Below are presented some solutions that have been suggested.

Partitional Clustering for Mixed Type Data

Many different extensions to k-means have been proposed to handle mixed data. One of the first was the k-prototypes algorithm suggested

by Z. Huang [7], where the Euclidean measure is used for numeric data, and the simple matching coefficient for categorical data. The final distance measure is a linear combination of the two, where the influence of each component is determined by a parameter γ. Since then many other extensions to k-means have been proposed for mixed type data, for example K-Means Clustering for Mixed Datasets (KMCMD) by Ahmad and Dey [31]. The authors propose a more sophisticated cost function for the mixed type data, and they incorporate ideas of subspace clustering. The properties of partitional clustering algorithms are true also in the case of mixed type data, that is low time complexity but sensibility to outliers and only finds spherical clusters.

Hierarchical Clustering for Mixed Type Data

Chiu et al. proposed an agglomerative algorithm [32] that is based on BIRCH [21] but uses a probabilistic distance measure. The distance between two clusters is based on the decrease in log-likelihood as a result of merging them together. Just as BIRCH, it builds a feature tree for the clustering, which enables the linear time complexity. Another clustering algorithm for mixed data based on BIRCH is presented in Rendón and Sánchez [33]. In this paper the authors use a distance measure based on compression cost. SBAC (Unsupervised Learning with Mixed Numeric and Nominal Data) [34] is a hierarchical clustering method that uses a probabilistic distance measure. It assigns higher values to uncommon feature values, thus rewards rare combinations of attributes. SBAC is computationally expensive, with a complexity of $\mathcal{O}(n^2)$. These algorithms can find clusters of arbitrary shape and are not sensible for noise.

Model Based Clustering for Mixed Type Data

Many different approaches for clustering mixed type data based on statistical models have been proposed. One example is Autoclass [8] which is a classic finite mixture model where Bayesian inference is used to estimate the number of clusters and their distributions. A modified Expectation Maximisation (EM) algorithm is used to estimate the parameters. Since it is assumed that the data is generated by a distribution, only clusters of the shape of the distribution can be found. Other work includes COBWEB/3 [17], which is an extension to the original COBWEB algorithm [26] that handles mixed type data.

It uses the CU-measure for categorical data, a slight modification of it for numeric data, and a linear combination of the two for the final measure. These types of algorithms are in general good at handling outliers, but they are computationally expensive.

Neural Network Based Clustering for Mixed Type Data

Extensions to the original SOM to handle mixed type data have been proposed, for example [20], [18]. In these articles, Hsu et al. use the distance hierarchy measure for categorical data, that encodes semantics between attributes. In [9] the authors use a frequency-based distance measure for categorical data, and the Euclidean measure for numeric data. Han et al. [35] propose an extension to the original ART algorithm for mixed attribute data. SOMs can find clusters of arbitrary shape and are robust against outliers, the same applies for the extension to mixed type data. One disadvantage of neural network methods is the low transparency in the model.

Information Theory Methods

To measure similarity in categorical data, some authors have taken inspiration from information theory and proposed distance measures based on entropy. In [36] and [37], Böhm et al. wishes to find clusters in the dataset that minimises the compression cost. For categorical data they use the entropy as a measure of the compression cost, and numerical data is modelled by a normal distribution. This kernel method to compute numeric distances is computationally expensive. They then proceed to do a hierarchical clustering using these distance measures. In [19] the distance of categorical data is again measured with entropy, but they also include the concept of distance hierarchy to include semantic similarities between categorical variables. Numeric data is measured using variance within the cluster. These methods have a varying ability to handle noise but are relatively fast. The entropy function is independent of the shape of the clusters, but all methods described here to measure numerical data can only find elliptical clusters.

Other Methods

Some other notable work has been done on mixed type data. For example, in [38], He et al. view ensemble clustering as a categorical clustering problem in itself. The original dataset is split into two subsets of pure categorical and numerical data. Some suitable clustering algorithms are applied to the two datasets, producing some categorical categorisation of the data. Finally, a categorical clustering algorithm is applied to the resulting categories. In [39], the authors propose a subset clustering method for mixed type data where different clusters can have different distinguishing attributes.

2.2.4 Summary

Table 2.1 shows a summary of the algorithms discussed in the previous sections.

Clustering methods based on partition is together with hierarchical clustering the most widely used methods for cluster analysis. Even though centre-based clustering algorithms can only find spherical clusters and suffer from sensibility to outliers, these methods have the great advantage of having a high transparency, in addition to being fast. k-prototypes [7] have been widely used as benchmark (for example by [40],[37] and [19]), and there are multiple ready-to-use implementations available.

Model based clustering methods are computationally slow, and assume a specific distribution of the data, usually a normal distribution. The same goes for the clustering methods based on information-theory found for this project, as they all assume spherically shaped clusters in the numerical dimensions.

Self-organising maps have won great popularity when applied to purely numeric data and are therefore promising also for mixed type data. They are robust against outliers and handles non-spherical clusters and are therefore considered a very good method for this project. They also provide a way to easily visualise high-dimensional data which is another big advantage, as it will increase the interpretability of the results.

Hierarchical clustering algorithms have many advantages, for example their ability to handle non-spherical clusters and their robustness to noise. Their greatest drawback is that they have high computational cost in general. Two methods ([32], [33]) that overcome this

drawback were found during the literature study.

Conclusions of literature review

The wide spread use of k-prototypes makes it a good candidate as a benchmark method also for this thesis. One of the existing ready-to-use implementations will therefore be used in this project.

From Table 2.1 it can be seen that the algorithms that handle mixed type data with clusters of arbitrary shape are:

- Extended BIRCH
- SBAC
- COBWEB / 3
- Extended self-organising map
- Art 2a
- Cosa

Of these algorithms, SBAC, Cosa and COBWEB/3 have very high time complexity and will thus be too slow for the current project. In addition, Art 2a is highly sensible to noise, making also this algorithm unsuitable.

The extended self-organising maps and the extended BIRCH algorithms both fulfil the criterion demanded for this project. Due to time limitations only one algorithm can be implemented. The only drawbacks of self-organising maps are their low transparency, but they do provide a good way to visualise the data that make the results interpretable. For this reason, an extended self-organising map will be included.

In the literature study three different extensions of the self-organising map for mixed type data was found. In [20] and [18], the authors use a distance hierarchy to measure the distance of categorical data. In [9] the authors use a frequency measure for categorical data. The latter is newer and better documented and was therefore chosen for this project.

Summary of algorithms					
Algorithm name	Type of algorithm	Data type	Time Complexity	Sensitive to noise	Shape of cluster
K-means [5]	Partitional	Numeric	$\mathcal{O}(nkt)$	High	Convex
BIRCH [21]	Hierarchical	Numeric	$\mathcal{O}(n)$	Low	Arbitrary
ROCK [22]	Hierarchical	Categoric	$\mathcal{O}(n^2 log(n))$	Low	Arbitrary
DBSCAN [23]	Density	Numeric	$\mathcal{O}(nlog(n)$	Low	Arbitrary
STING [24]	Grid	Numeric	$\mathcal{O}(n)$	Low	Arbitrary
GMM [25]	Model	Numeric	$\mathcal{O}(n^2kt)$	Low	Arbitrary
COBWEB [26]	Model	Numeric	Distribution	Moderate	Arbitrary
SOM [29]	N.N.	Numeric	Layer	Low	Arbitrary
ART [41]	N.N.	Numeric	Type + layer	High	Arbitrary
K-prototypes [7]	Partitional	Mixed	$\mathcal{O}(nk(t+1))$	High	Convex
KMCMD [31]	Partitional	Mixed	$\mathcal{O}(nkt)$	Low	Convex
Extended BIRCH [32],[33]	Hierarchical	Mixed	$\mathcal{O}(n)$	Little	Arbitrary
SBAC [34]	Hierarchical	Mixed	$\mathcal{O}(n^2)$	Low	Arbitrary
Autoclass [8]	Model	Mixed	$\mathcal{O}(nktd^2)$	Low	Convex
COBWEB/3 [17]	Model	Mixed	Distribution	Moderate	Arbitrary
Extended SOM [18],[20],[9]	N.N.	Mixed	Type + Layer	Low	Arbitrary
ART 2a [35]	N.N.	Mixed	Type + layer	High	Arbitrary
Inconco [37],[36]	I.T.	Mixed	Kernel	Low	*
CAVE [19]	I.T.	Mixed	$\mathcal{O}(n^2)$	High	*
Ensemble [38]	Ensemble	Mixed	*	*	*
Cosa [39]	Subset	Mixed	$\mathcal{O}(n^3)$	Low	Arbitrary

Table 2.1: Summary of the properties of the algorithm presented in chapter 2. In the time complexity column, n is the number of data points, k the number of clusters, t the number of iterations and d the dimensionality of the data. In the type column N.N. stands for neural network and I.T. for information theory methods. Fields marked with * see text for discussion.

2.3 Self-organising maps

The self-organising map was proposed by Kohonen [29] and is a clustering algorithm for numeric data. It is also widely used to visualise high-dimensional data as it projects $n-$dimensional data into a two-dimensional map. The algorithm can find clusters of arbitrary shape and is very robust against outliers. It has gained much popularity in recent years and the method has produced very good results [9].

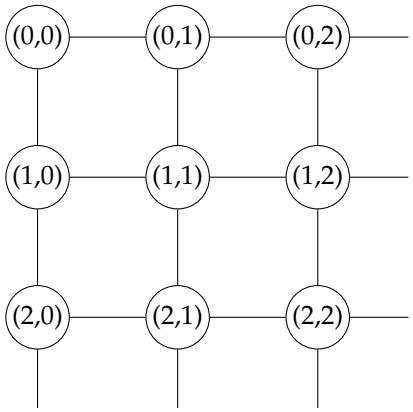


Figure 2.3: Schematic view of a neuron grid. Neighbouring neurons are connected with a line. Each data point is "assigned" to one neuron, its *BMU* (Best Matching Unit). The neurons have weight vectors for each dimension in the dataset, that are updated according to some update rule to become more similar to the data points that have it as its BMU. Neurons that are close to each other in the grid (neighbours or second neighbours etc) should represent data points that are more similar to each other, and are thus more inclined to belong to the same cluster.

The variables used in a SOM is:

- The number of input vectors P (size of dataset)
- The number of features F (the dimensionality of the dataset)
- Input vector $X_p = [x_{p1}, ..., x_{pF}]$
- The number of map neurons I
- The weight vector $W_i = [w_{i1}, ..., w_{iI}]$

The algorithm is summarised by the following steps:

1. Initialisation: Assign map size and randomly initialise node weights
2. For each node in the map:
   (a) Calculate the (Euclidean) similarity between the input vector and the node weight vector.
   (b) Find the node that corresponds to the smallest distance, the BMU
3. Recalculate the weight vector of the neighbouring nodes of the BMU according to some update rule
4. Iterate steps (2-3) until convergence or maximal number of iterations is reached.

2.4 Evaluation methods

Clustering is an unsupervised method and there is no predefined label of what the correct division of the dataset is. Sometimes not even the number of clusters is known, making it highly non-trivial to determine the validity of the result of a clustering process.

2.4.1 Evaluation criteria

In general, there are three ways to evaluate clusters, namely *external*, *internal* and *relative* criteria [42]. External criteria are based on the existence of some predefined structure which is imposed on the data. This could be intuitive structures of the data or predefined class labels. Using an internal criterion, the result is evaluated based only on structures which are inherent in the data, such as compactness and separation of clusters [4]. The idea of relative criteria is to compare the result of one clustering to other results, obtained with the same algorithm but using different parameter values. A common example is varying the number of clusters when this is not known *a priori* and plotting the number of clusters against some validity index. A fast local change, a "knee", in the plot would then indicate the optimal number of clusters. The absence of a knee is an indication that the dataset possesses no clustering structure. All these approaches rely on some validity index, which will be discussed in section 2.4.3.

2.4.2 Hypothesis testing

Another approach, proposed for example by Bock [43], to evaluate the result of a cluster analysis is to investigate if the data points are randomly structured or not. The test is based on the null hypothesis H_0 that the dataset has a random structure (i.e. no clusters exist in the data), and a competing hypothesis A that clusters exist in the data. In some basic cases one can analytically calculate the test statistics and use statistical methods to determine if one can discard the null hypothesis or not. In most cases this is not possible, and a qualitative comparison can instead be made. The results of running the same algorithm on the real dataset and a set of random data are compared to determine if the dataset is randomly structured or not.

2.4.3 Validity indices

To evaluate the results of a clustering algorithm, some validity index is needed. Below are presented a few validity indices for numerical, categorical and mixed type data.

Indices for numeric data

The mean-square error of P data points x_i is defined as:

$$MSE = \sum_{i=1}^{P}(x_i - p_{k_i}) \tag{2.9}$$

where p_{k_i} is the cluster centre (or best matching unit) that x_i is assigned to.

Indices for categorical data

The notion of category utility (CU) originally proposed by [16], can be used as a validity index for categorical data [19]. The CU measure aims to maximise the probability of a feature belonging to a certain class and the probability of an instance in a certain class having a particular feature. Let $P(A_i = V_{ij}|C_k)$ be the conditional probability that attribute i (A_i) has the value V_{ij} given cluster C_k, and $P(A_i = V_{ij})$ the overall probability of the attribute i having values V_{ij} in the entire dataset.

Then the validity index can be defined as follows:

$$CU = \sum_{k} \frac{|C_k|}{D} \sum_{i} \sum_{j} \left[P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2 \right] \tag{2.10}$$

Where $|C_k|$ is the size of cluster k and D is the total size of the dataset. The higher the CU-value, the better the clustering result.

Indices for mixed type data

In [19], Hsu et al. propose a validity index based on the CU measure, which they call the CV value, and that works for mixed type attributes. The CV-value is defined as follows:

$$CV = \frac{CU}{1 + Variance} \tag{2.11}$$

The CU-value is the validity index for categorical data as defined above, and is used for the categorical attributes. Normal variance is used for the numerical attributes, and the final validity index is the combination of the two. The higher the CV value the higher the better the clustering result.

Chapter 3

Method

3.1 Data specification

3.1.1 Raw data

The raw data consists of customer information and can be both numerical and categorical. The data used consisted of $\sim 10^5$ companies, with around 30 features in the raw dataset. The categorical data have from two up to around 30 different categories, and all numerical data was normalised to range from zero to one.

3.1.2 Fabricated datasets

Two different synthetic datasets were generated. The first dataset consisted of four well defined clusters, which were generated by drawing points from four normal distributions with randomly chosen mean. This dataset is similar in dimensions to the real dataset, with seven numerical and six categorical features (see below). The second dataset consisted of completely random data, again of the same dimensions as the real dataset.

3.1.3 Feature selection

The feature selection was mainly done by a domain expert at Handelsbanken, not analytically. The features used in the clustering overlaps with the features used in one of the current risk models at use at Handelsbanken: some are the same others are features that are under investigation for future inclusion.

For the feature selection phase, we observed that some attributes, both numerical and categorical, had data quality issues that needed to be resolved first. To aid in the selection process, it was decided to use a set maximum threshold for the fraction of missing values for the attribute to be included.

After selecting only the relevant features and removing attributes with data quality issues seven numerical and six categorical attributes remained.

3.1.4 Null values

Even after removing the attributes with data quality issues, there was still missing values to be handled. Two different approaches were tried to handle the null data. The first one was to replace categorical missing values by the majority class of that specific feature, and missing numerical value by the mean value of that existing value of that feature. This approach was suggested in [9]. The resulting dataset is denoted "*Nulls to mean*" in the text and consisted of six categorical and seven numerical attributes.

A second approach only used the high-quality data points, i.e. only the ones without missing values. This meant removing some additional attributes of low quality and reducing the number of data points to about $1/3$ of the original size. The resulting dataset is denoted "*Filtered nulls*" and consisted of six categorical and five numerical attributes.

3.2 Clustering algorithms

3.2.1 k-prototypes

k-prototypes was originally proposed by Z. Huang in 1997 and was one of the earliest algorithms designed to handle mixed type data [7]. The algorithm is initialised by randomly choosing k cluster centres, so called *prototypes*. By iteratively reallocating these prototypes to better fit the data, the algorithm tries to minimise the total distance of points assigned to a cluster and its prototype, much like the k-means algorithm. The distance function d is defined as follows for a data point x_i

and a cluster prototype Q_l

$$d(X_i, Q_l) = \sum_{j=1}^{m_r}(x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c}\delta(x_{ij}^c, q_{lj}^c) \tag{3.1}$$

Here m_r is the number of numerical attributes, m_c the number of categorical objects, and δ is defined as in section 2.2.1. γ_l is a user specified parameter that describes the influence of the categorical versus numerical attributes. This distance measure is thus a linear combination of the Euclidean measure and the simple matching coefficient.

The algorithm can be described in a few steps:

1. Randomly select k initial prototypes from the dataset, one for each cluster.

2. Allocate each data point to the cluster whose prototype is nearest to it according to equation 3.1. Update the prototype of the corresponding cluster after each allocation to be the new centre of the data in the cluster.

3. When all data points are assigned to a cluster, recalculate the similarity of all objects against the current prototypes. If an object is found to be nearer another prototype than the one of the cluster it belongs to, reallocate to that cluster and update the prototypes of both clusters.

4. Repeat steps (2-3) until convergence (i.e. no data point changes cluster).

3.2.2 Extended self-organising maps

Different methods for incorporating mixed type data for the self-organising map have been proposed, for example [18] and [20]. For this degree project a method based on [9] has been implemented. This is an extension of the batch version of the self-organising map [29]. The idea of this approach is to separate the weight vector into two parts, one for categorical data and one for numerical. This gives us a new definition of the following variables:

1. The number of categorical features K

2. The number of numerical features N

3. The weight vector $W_i = [w_{i1}, ..., w_{in}, w_{in+1}, ..., w_{ik}]$

The similarity measure is defined as:

$$d(X_p) = D_n(X_p, W_i) + D_c(X_p, W_i) \tag{3.2}$$

where D_n is the Euclidean distance as before for numerical attributes. The categorical dissimilarity function is defined as:

$$D_c(X_p, W_i) = \sum_{z=n+1}^{k} (1 - W_{iz}[X_{pz}])^2 \tag{3.3}$$

This measure can be interpreted as the sum of the partial dissimilarities obtained for each categorical feature, where the partial dissimilarities are the probability of the reference vector not holding the category present on the input vector.

This modified algorithm is then described by the following pseudo code:

1. Initialise the map by randomly assigning neuron weights
2. For each data point X_p and neuron *i* in map:
   (a) Calculate the distance between X_p and W_i
   (b) Find the BMU as the neuron with the smallest dissimilarity according to equation 3.2
3. Update the weight vectors (equations 3.4 - 3.5)
4. Check for convergence:
   (a) If no BMUs changed or maximum iterations: **break**
   (b) Else reset weight update vectors, repeat from step 2

The weight vectors are updated according to the following equations, for numerical and categorical data respectively:

$$W_{in}(s+1) = \frac{\sum_{p=1}^{P} h(s, c(X_p), i) * X_{pn}}{\sum_{p=1}^{P} h(s, c(X_p), i)} \tag{3.4}$$

$$W_{ik}(s+1) = \left\{F(\alpha_k^1, W_{ik}(s)), F(\alpha_k^2, W_{ik}(s)), ..., F(\alpha_k^r, W_{ik}(s))\right\} \tag{3.5}$$

where F is defined as follows

$$F\left(\alpha_k^r, W_{ik}(s)\right) = \frac{\sum_{p=1}^{P} h(s, c(X_p), i) | X_{pk} = \alpha_k^r}{\sum_{p=1}^{P} h(s, c(X_p), i)} \tag{3.6}$$

The Gaussian neighbourhood function h is defined as:

$$h(BMU, i) = exp\left(\frac{-d^2}{2\sigma(s)^2}\right) \tag{3.7}$$

where d is the distance in the lattice from the winner neuron (BMU) to the neuron i, and $\sigma()$ is defined as

$$\sigma(s) = \sigma(1) exp\left(\frac{-s}{T}\right) \tag{3.8}$$

where s is the current iteration, $\sigma(1)$ is an initial radius and T is a shrinkage factor. These parameters need to be set by the user and some initial experiments were conducted to find appropriate values, as described in the following sections.

3.3 Parameter selection

3.3.1 k-prototypes

The k-prototypes algorithm has a parameter γ that controls the relative effect that the numeric and categorical attributes have on the total distance, as follows:

$$d_{tot} = d_{numeric} + \gamma d_{categorical} \tag{3.9}$$

In the k-prototypes algorithm, the number of clusters k can also be considered a parameter as it has to be specified by the user. As described in chapter 2.4.1, a knee-plot is a common way to find the optimal number of clusters. This approach was therefore taken in this project.

Since the results of k-means can vary greatly depending on initialisation, the average over four trials for each value of γ was used for the knee plot (giving a total of 16 trials). This was done using the values $\gamma = 0.1, 1, 5, 20$.

3.3.2 Self-organising map

The batch version of the self-organising map implemented for this thesis has two parameters T and $\sigma(1)$, that together control the speed of convergence of the algorithm. $\sigma(1)$ is the initial radius of the Gaussian neighbourhood, meaning that it controls how many neurons around the BMU that will be affected. In [9], the authors recommend that this parameter is set depending on the number of neurons, such as the number of columns, as it should cover a large portion of the grid. This rule of thumb was therefore used also for the current project.

The grid size (number of neurons) n can also be considered a parameter. A larger number of neurons will allow for more detail in the representation of the dataset, but it also greatly increases the calculation time. A too large grid can also lead to results that are difficult to interpret.

A parameter search for the optimal number of neurons n and for the decay speed T was conducted. The values $n = 15, 20$ and $T = 10, 25$ was chosen based on the results in the original article [9]

3.4 Implementation

This section describes the software packages and hardware used for this thesis. All programming have been done in Python 2.7, using the libraries described in the following subsection.

3.4.1 Software

NumPy[1] is a package for computing in Python. It contains a powerful object for N-dimensional arrays, as well as numerous mathematical functions that operate on these arrays. The packages specified here are dependent on NumPy and mostly on its arrays. The SciPy package[2] builds on the NumPy array object and contains a large number of mathematical function that can operate on them. Matplotlib[3] is a widely used package for Python that allows plotting. The k-modes package [4] includes an implementation of k-prototypes that was used

[1] www.numpy.org/
[2] www.scipy.org/
[3] https://matplotlib.org/
[4] https://github.com/nicodv/kmodes

for this project.

3.4.2 Hardware

The experiments have been conducted on an Intel Xeon E3-1240 processor.

3.5 Evaluation approach

3.5.1 k-prototypes

The results of k-prototypes are evaluated using the relative-criteria approach (see section 2.4.3). The validity index used is the one built-in to the k-modes package, namely the total in-cluster distance. The distance between numerical attributes is measured with the Euclidean distance and the distance in categorical attributes is the sum of the simple matching coefficient multiplied with gamma. The CV value (see section 2.4.3) was also used to measure the results of the k-prototype algorithm.

3.5.2 Self-organising maps

The CV value was used for evaluation also in the case of the self-organising map. Each neuron was counted as a cluster centre for this calculation. A mean-square-error (MSE) was also used to measure the in-cluster distance, again having each neuron as a cluster centre, using the Euclidean distance for numerical attributes and the frequency distance metric defined in equation 3.3 for categorical attributes.

The self-organising map also provide a good visual representation of the results using a U-matrix. The U-matrix shows the distance between neuron weight vectors and the weight vectors of their neighbours. The shorter the distance between the weight vectors the more similar the neurons are, and thus the more similar are the data points having the neurons as BMUs. Regions of low distance in the U-matrix therefore suggest the existence of a cluster.

Chapter 4

Results

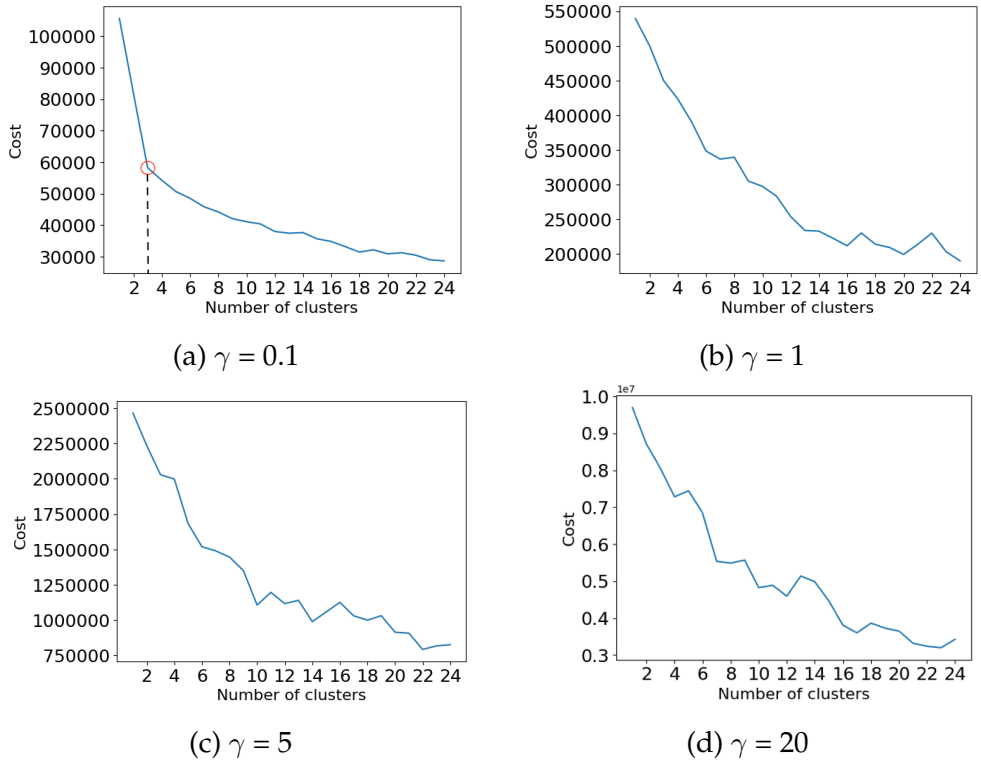
4.1 k-prototypes

4.1.1 Parameter selection

Figures 4.1 and 4.2 show knee-plots for different values of γ ($\gamma = 0.1, 1, 5, 20$), for the two different approaches to handle null values, i.e. "Nulls to mean" and "Filtered nulls", respectively. Figure 4.3 shows the same plots for random data. In the random data the cost is smoothly decreasing with a larger number of clusters, it does not appear to be a knee for any cluster number. Similarly, for both the "Nulls to mean" and "Filtered nulls" datasets, $\gamma = 1, 5, 20$ (Figures 4.1b-4.1d and 4.2b -4.2d) does not show any clear knees. For $\gamma = 0.1$ however, both datasets show a clear knee for 3 and 4 clusters respectively, marked with a red circle in Figure 4.1a and 4.2a. The value $\gamma = 0.1$ was therefore chosen as the optimal value of this parameter.

4.1.2 Cluster centres

Table 4.1 shows the resulting prototypes (cluster centres) when running the k-prototypes algorithm with $\gamma = 0.1$ and $k = 3$ on the "Nulls to mean" dataset. The results are representative also for the "Filtered nulls" datasets, on the relevant attributes. Table 4.1 also shows the proportion of data points assigned to each respective cluster. The k-prototypes algorithm is unstable, meaning that different initial choices of the centres will result in different final prototypes. The cluster centres shown here are the result of one specific run.



(a) $\gamma = 0.1$

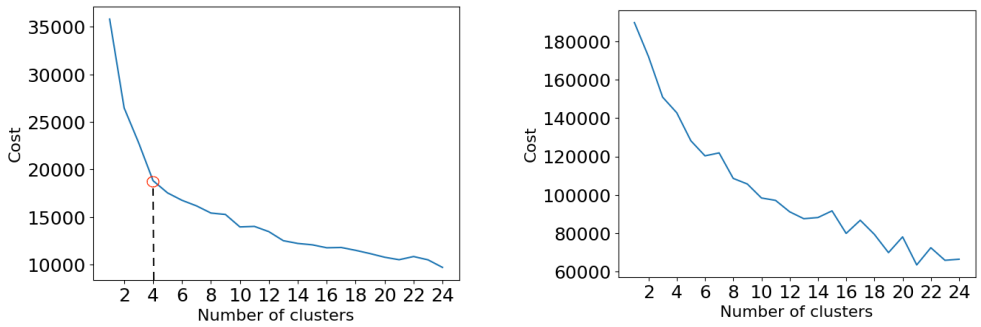
(b) $\gamma = 1$

(c) $\gamma = 5$

(d) $\gamma = 20$

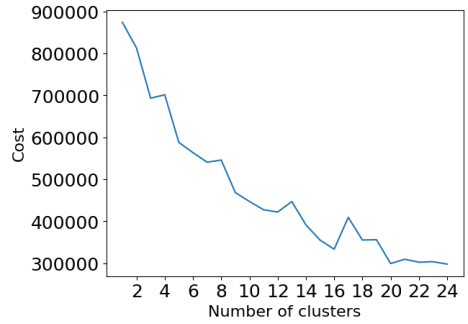
Figure 4.1: Cost as a function of the number of clusters, for different values of γ. Here on the dataset "Nulls to mean".

In Table 4.1 it can be observed that the different numerical attributes have a spread over different values in different clusters. In contrast, many of the categories have the same value over some or all of the clusters. The corresponding validity indices for these results was a cost (total in-cluster distance) of 58000 and the CV value of 1.25.

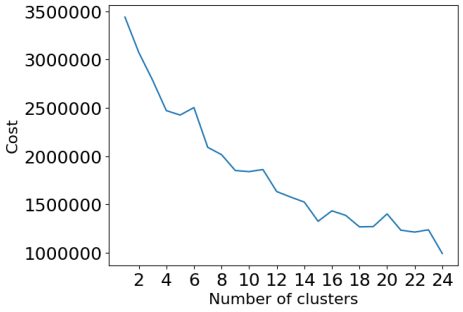


(a) $\gamma = 0.1$

(b) $\gamma = 1$



(c) $\gamma = 5$

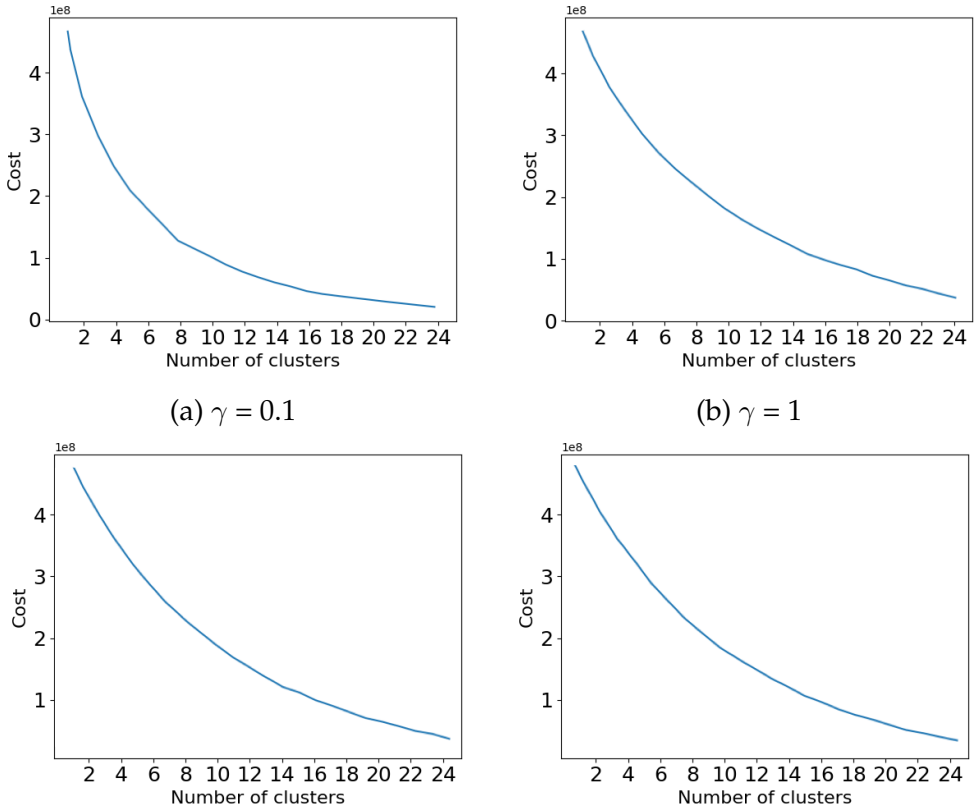


(d) $\gamma = 20$

Figure 4.2: Cost as a function of the number of clusters, for different values of γ. Here on the dataset "Filtered nulls".

Cluster / Attribute	0	1	2
N0	32.8	7.8	5.1
N1	34.7	8.8	5.2
N2	13619477.7	614416.5	88057.3
N3	71755.7	27335.8	1229.3
N4	1311.8	119190.9	121.5
N5	23.3	11.2	15.2
N6	343.3	244.9	470.4
C0	1	2	3
C1	5	5	7
C2	1	1	1
C3	49	49	53
C4	1	1	1
C5	0	1	1
Portion of data points	10%	67%	23%

Table 4.1: Cluster centres from k-prototypes for the seven numerical (N0-N6) and six categorical (C0-C5) features, using $\gamma = 0.1$. The categorical feature names have been replaced by integers due to secrecy. The last column shows the proportion of the data points that have been assigned to each cluster. Here on the "Nulls to mean" dataset.



(a) $\gamma = 0.1$ (b) $\gamma = 1$

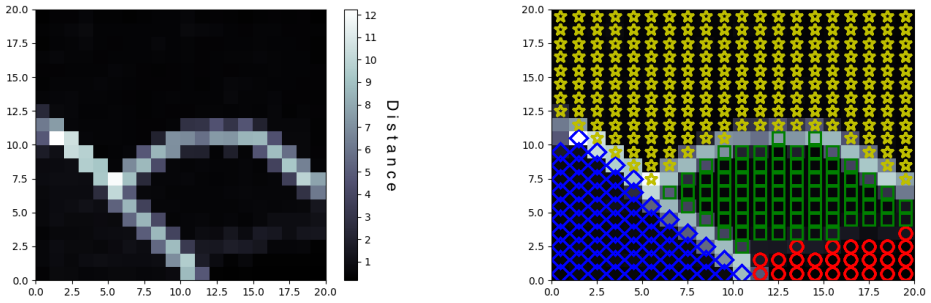
(c) $\gamma = 5$ (d) $\gamma = 20$

Figure 4.3: Cost as a function of the number of clusters, for different values of γ. Here on the random data.

4.2 Self-organising maps

4.2.1 Implementation validation

Figure 4.4 shows the resulting U-matrix from the synthetic cluster dataset after the self-organising map. Labels showing which normal distribution each data point is drawn from is shown for comparison in Figure 4.4b for the data points mapped to each neuron. As described in section 2.4.1, the U-matrix show the distance between the weight vector and its neighbours for all dimensions. A darker colour indicates a low distance. The clear boundaries between the yellow and blue, yellow and green and green and blue labels suggest that these clusters are very well separated. The darker boundary between the red and the green labels suggests that these clusters are more similar to each other



(a) U-matrix for a synthetic dataset with 4 clear clusters.

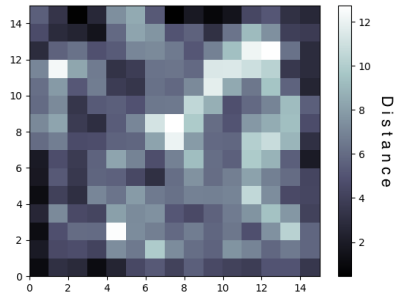
(b) U-matrix and labels for a synthetic dataset with 4 clear clusters.

Figure 4.4: Result of the self-organising map applied to a synthetic dataset with four distinct clusters.

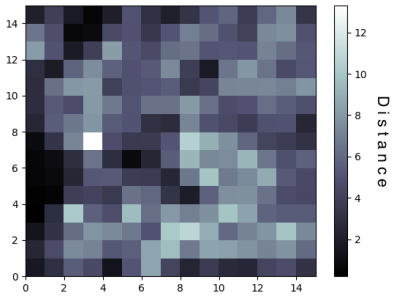
than the clusters, where the separation is marked with a very light colour. It is clear that points originating from the same normal distributions are mapped to neurons that are very similar, and that the algorithm clusters data in a satisfying way.

4.2.2 Parameter selection

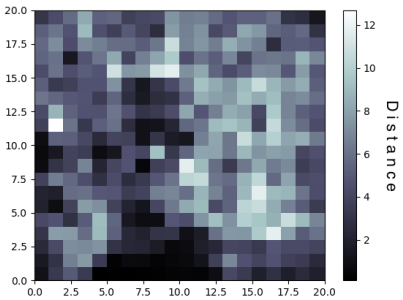
Figures 4.5 and 4.7 show the U-matrices for different parameter settings for the two different approaches to handle null values. Figures 4.6 and 4.8 show the MSE and CV values from the same experiments. In both datasets, the MSE is lowest for $n = 20$ and $T = 10$. The CV values is however highest for $n = 15$ for both datasets, with $T = 10$ for "Nulls to mean" and $T = 25$ for "Filtered nulls". As the MSE is more reliable than the CV value, $n = 20$ and $T = 10$ was concluded to be the best parameter values for both datasets.



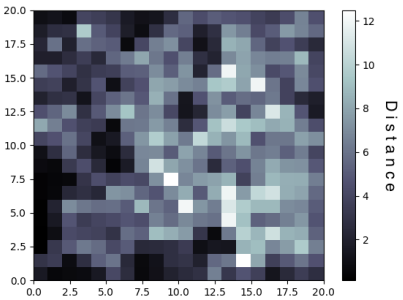
(a) 15 neurons, $T = 10$



(b) 15 neurons, $T = 25$



(c) 20 neurons, $T = 10$



(d) 20 neurons, $T = 25$

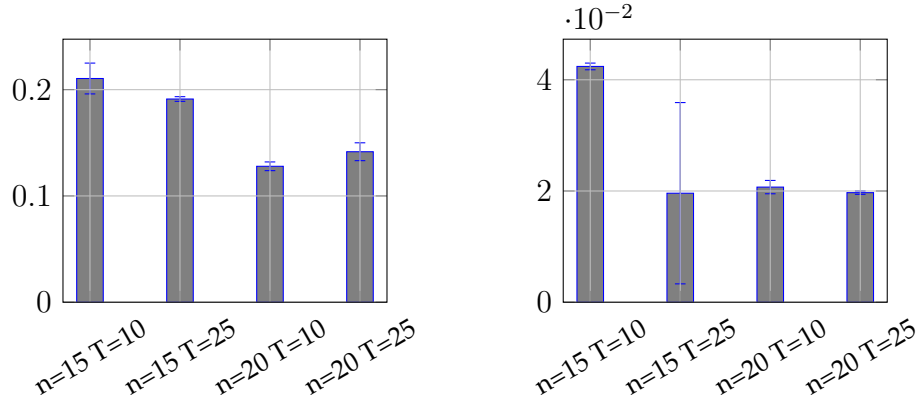
Figure 4.5: U-matrices for the dataset "Nulls to mean" using different values of T and number of neurons

4.2.3 Results

Using the parameter values found in the previous section, the "Nulls to mean" and the "Filtered nulls" dataset was compared to random data. The results can be seen in Figure 4.9. Table 4.2 shows the corresponding MSE and CV values, as well as the results for the synthetic cluster data used in section 4.2.1.

Figure 4.10 shows neuron weight vector in some selected dimensions, corresponding to four different features (two numerical and two categorical). The categorical feature per neuron was taken to be the feature the neuron had highest probability for, and the numerical feature is just the value of the weight for that neuron. The actual class of the categorical values have been replaced by integers for secrecy.

Figure 4.11 shows the cluster centres from running k-prototypes with $k = 3$, plotted as "labels" onto a U-matrix of the "Nulls to mean"



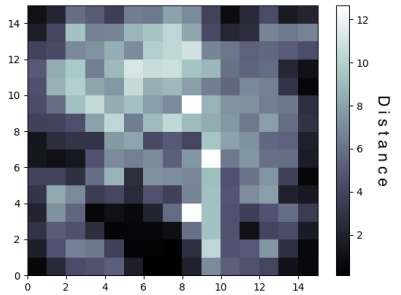
(a) MSE (b) CV

Figure 4.6: The two validity indices for different values of T and number of neurons n, for the dataset "Nulls to means". Average of two runs, with maximum and minimum values plotted in blue.

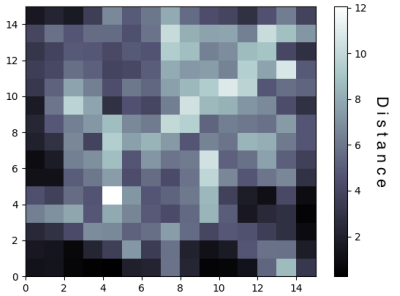
Dataset	MSE	CV
Nulls to mean	0.13 +- 0.004	0.021 +- 0.001
Filtered nulls	0.11 +- 0.004	0.026 +- 0.001
Random	0.44	0.0032
Synth. cluster	0.0062	0.045

Table 4.2: Table with results for the different datasets. All datasets were evaluated using $T = 10$ and $n = 20$.

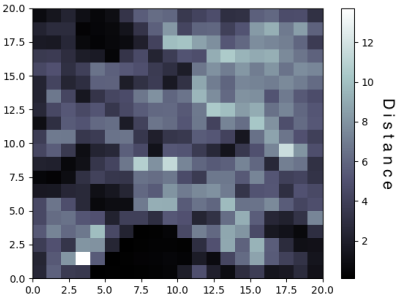
dataset. The U-matrix comes from running the self-organising map algorithm with $n = 20$ and $T = 10$ on the same dataset.



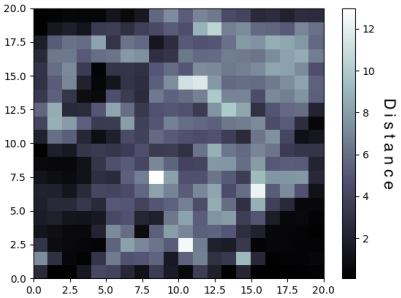
(a) 15 neurons, $T = 10$



(b) 15 neurons, $T = 25$

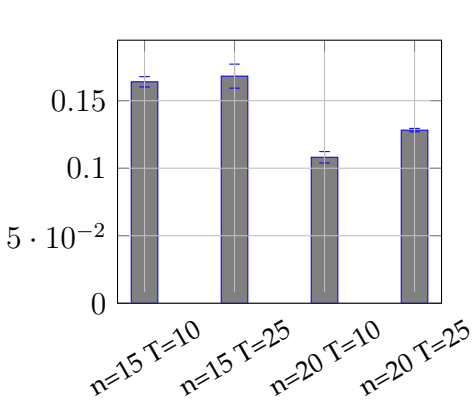


(c) 20 neurons, $T = 10$

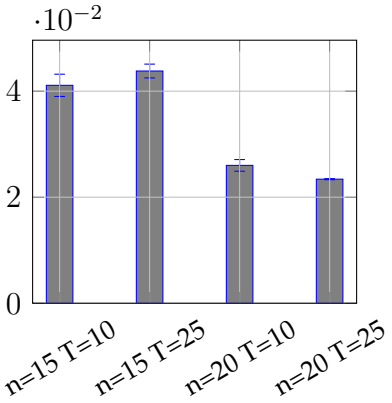


(d) 20 neurons, $T = 25$

Figure 4.7: U-matrices for the dataset "Filtered nulls" using different values of T and number of neurons

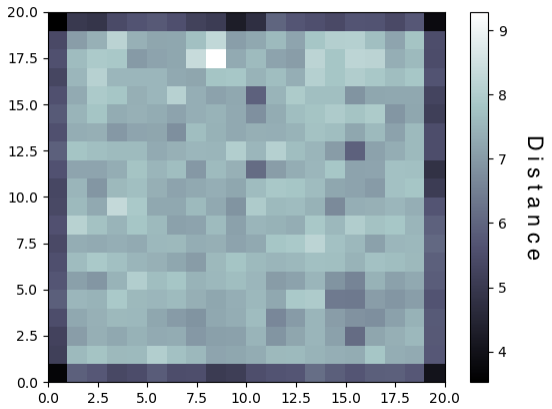


(a) MSE

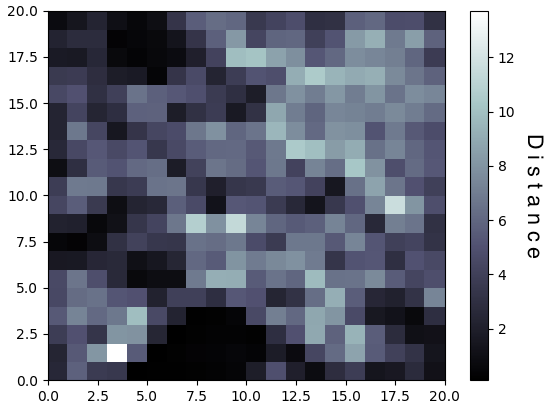


(b) CV

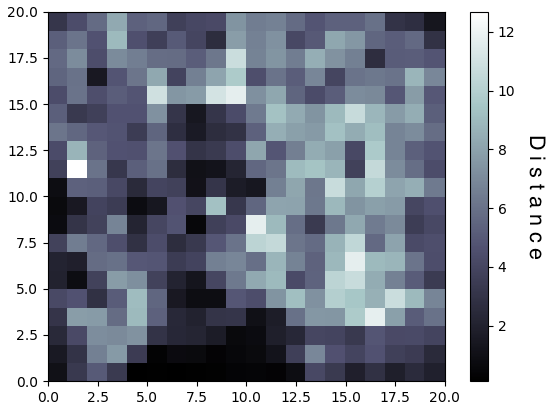
Figure 4.8: The two validity indices for different values of T and number of neurons n, for the dataset "Filtered nulls". Average of two runs, with maximum and minimum values plotted in blue.



(a) Random data

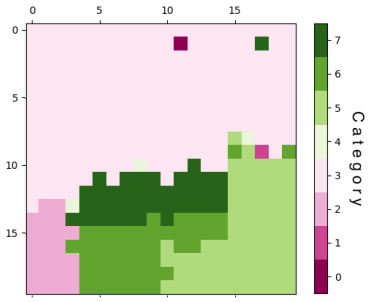


(b) Filtered nulls

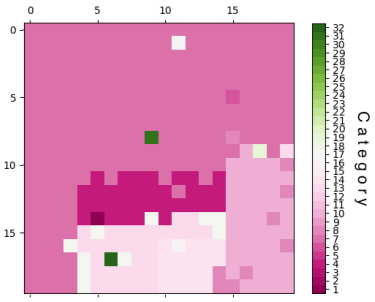


(c) "Nulls to mean" dataset

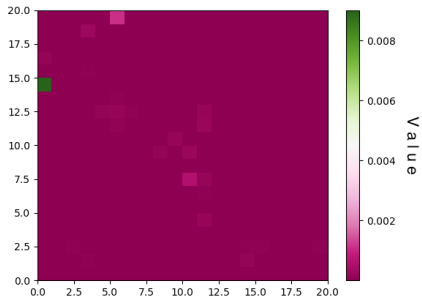
Figure 4.9: U-matrices from different datasets using the same parameters ($T = 10$, 20 neurons).



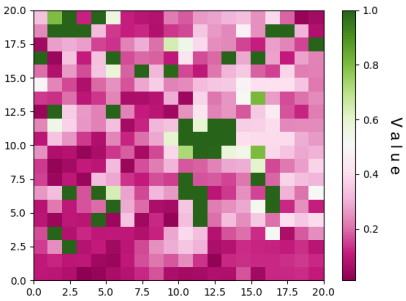
(a) Categorical feature 1



(b) Categorical feature 3

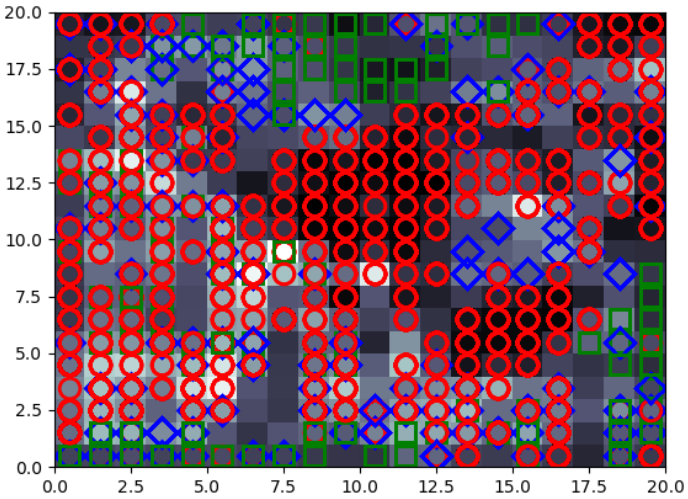


(c) Numerical feature 2

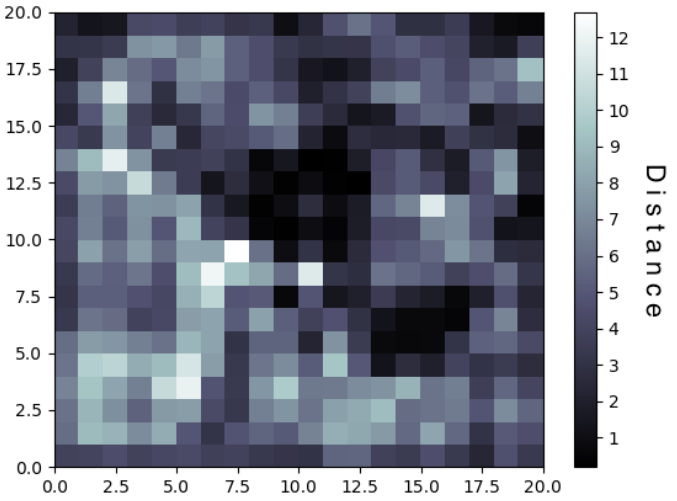


(d) Numerical feature 5

Figure 4.10: Weights per attribute for a two numerical and two categorical features.



(a) Three cluster centres from k-prototypes.



(b) Same U-matrix but without "labels"

Figure 4.11: Cluster centres from k-prototypes plotted as labels in self-organising map, using $n = 20$ and $T = 10$

Chapter 5

Discussion

5.1 Discussion of results

The clear knees in plots 4.1a and 4.2a suggest that there are clear clusters in the data, with 3 and 4 clusters respectively for the two ways of handling missing values. These images should be compared to Figure 4.3 where the error is decreasing very smoothly with the cluster numbers, indicating no existence of a cluster.

The optimal value for γ was found to be the lowest of the ones tried, $\gamma = 0.1$. This indicates that the clusters are more prominent in the numerical dimensions than in the categorical dimensions, as a low value of γ makes the impact of the categorical attributes lower. Since the optimal value for γ was found to be the lowest of the ones tried, it is possible that an even lower value for the parameter would be better.

Table 4.1 shows the cluster centres from the k-prototypes algorithm using $\gamma = 0.1$ on the "Nulls to mean" dataset. It can be observed that the cluster centres vary in all the numerical attributes for the different clusters. In contrast, many of the categorical attributes are the same over some, or all, of the cluster centres. These results confirm that the numerical attributes have had a bigger impact on the clusters, suggested by the knee-plots for different values of γ. This could be due to the natural clusters in the data foremost existing numerical dimensions. It could also be an effect of the 1-of-k encoding used for the categorical attributes in the k-prototypes algorithm. As discussed in section 2.2.3 this method suffers from an increase in dimensionality, and it has been shown that the Euclidean distance measure is not suitable for categorical data in large domains. Some of the categorical

attributes have up to 30 possible categories, for such large dimensions the Euclidean distance will yield extremely large distances even for intuitively close points (known as the curse of dimensionality). Another reason for the clusters being more prominent in the numerical dimensions could be that information is lost in this transformation, resulting in worse clustering in the categorical dimensions.

A domain expert was consulted about the results in Table 4.1. Some of the features confirmed with their prior belief of the data, for example categorical feature 0 and numerical feature 5. They were however much surprised by some of the numerical attributes, for example numerical feature 2. According to them, the raw data in this dimension mostly consists of low values (<10), with some extreme outliers having very high values. As discussed in the theory section, the k-prototypes algorithm is very sensitive to outliers, which could be one reason for these unexpected results. Another reason for the deviant values could be the "Nulls to mean" approach to handle null data. Because of the extreme outliers in the raw data, the median of the data will be much lower than the mean for this attribute. Therefore when the missing values are replaced with the mean, they will become outliers themselves, thus skewing the cluster centres even more.

Figure 4.5 and 4.7 show a parameter search over the two different parameters T and n. Unsurprisingly, the corresponding MSE values in 4.6a and 4.8a suggest that the biggest increase in performance comes from increasing the number of neurons. However, increasing the number of neurons greatly increases the time complexity. It also makes the results harder to interpret and use.

The parameter search raises some questions about the CV-validity index. Its results are often unintuitive, for example giving better results for fewer neurons in the self-organising map. The results are also opposite to that of the MSE in Figures 4.6 and 4.8.

Figure 4.9 shows that both datasets are clearly more structured than the random dataset in figure 4.9a. It is however clear that the clusters are not as crisp as the synthetic cluster dataset in Figure 4.4a. This is confirmed by the summary in Table 4.2, where both "Nulls to mean" and "Filtered nulls" get better values than the random data, but worse values than the synthetic cluster data. This result is expected, it suggests that clusters exists but with a considerable amount of noise.

Figure 4.10 shows the neuron weight vectors per parameter in four different dimensions. It is clear that the two categorical features 4.10a

and 4.10b are correlated. They also show a nice segmentation of the grid space, with very little noise. The results of the numerical features in 4.10c and 4.10d do not show the same nice segmentation, but they still provide valuable information. Numerical feature 2 in 4.10c contains only very low values, potentially with one neuron (coloured green) allowing for some higher values. This indicates that the self-organising map was robust to the outliers present in the data, as suggested by theory (see further discussion below). Numerical feature 5 in Figure 4.10d show some structure (for example the bottom neurons all span the lower range), but overall the segmentation is quite poor. This could indicate that the self-organising map clustered better on the categorical features than the numerical features.

Two different approaches to handle null data were tried in this project. One of these approaches was to simply remove low-quality data points and the second was to substitute missing values for the majority class and mean value for categorical and numerical features respectively. Not surprisingly does the "Filtered nulls" approach yield the lowest MSE value and the highest CV value as seen in Table 4.2. The lower MSE value could simply be a result of the dataset containing fewer dimensions, but it could also reflect the loss of information from replacing nulls with means.

5.2 Comparison of the algorithms

The results from k-prototypes are in one way easier to interpret, it yields k cluster centres from the data, that could be interpreted as k type customers. However, the results from the self-organising map suggests that this is a great simplification, that there is not a "simple" way to cluster the dataset. The self-organising map suggests that clusters exists in the data, but really visualises the complexity of the structure in the dataset.

Figure 4.11 shows the cluster centres from k-prototypes plotted as labels onto the U-matrix from the self-organising map, for the "Nulls to mean" dataset. The regions in the U-matrix that are very dark are all occupied with red circles, with little overlap with the other two classes. Lighter regions (for example lower left corner) have a bigger mix of classes, with overlaps and much variety in label. In the upper middle section where the U-matrix is also very dark, k-prototypes yields

a green area. As discussed above, the U-matrix tells us a more complex story about the data than the cluster centres from k-prototypes. The data points that k-prototypes have put into one cluster are instead mapped to an area of bigger variation in the self-organising map.

As discussed above, the k-prototypes algorithm performed badly in the dimension with outliers. As a comparison, Figure 4.10c is the results for the same parameter but from the self-organising map. However normalised, all values span the extreme low range of the possible values, corresponding to the values of the majority of the points in the raw data. It is clear that the self-organising map was much better than the k-prototypes algorithm at handling the outliers in this attribute, as predicted by the theory.

5.3 Context

Interpreting the results from an unsupervised learning scheme is a very difficult task, and much work in the field has been focuses on this issue (for example [36]). Interpreting, and evaluating, the results from the cluster analysis have been difficult also in this project.

The first part of this thesis work was to determine the most suitable method. According to the literature, the self-organising map should be a better algorithm for the current problem than k-prototypes and give better results. As discussed above, it is hard to conclude which algorithm has performed better. For example, the self-organising map was better at handling outliers (according to the example with numerical feature 2), but the results from the k-prototypes algorithm are easier to interpret. Because of the higher transparency and interpretability of the results, the k-prototypes algorithm seems more useful in the context of anti-money laundering, however the clusters will not tell the whole story.

In Alexandre and Balsa [15], the authors manage to find clusters that correspond to known customer groups. Their results are however heavily dependent on the numeric data, and the impact on the results from the categorical data is less significant. In this project, using the self-organising map, the clustering is more balanced between the numeric and categorical data. In [15] the authors use the 1-of-k encoding, which as discussed above, might result in the loss of some information from the categorical dimensions.

5.4 Future work

For this thesis project, little focus was put on how to handle missing values in the data. For future work, more sophisticated methods to handle missing values in the data should be tried. As discussed earlier, for example trying the median instead of the mean could have benefits in some of the numerical dimensions where outliers are present. One could also try clustering in the logarithm space of the values as the order of magnitude varies greatly. It could also be interesting to try (supervised) machine learning methods for predicting missing values.

As discussed above, it is hard to interpret and evaluate results from an unsupervised learning algorithm. One way of overcoming this difficulty would be to compare the found clusters with, for example, known cases of money laundering (e.g. applying an external criteria).

In the literature study, it was found that either one of the self-organising maps or one of the fast, hierarchical clustering algorithms fulfilled the criterion for the current problem. The self-organising map was chosen due to it providing a nice visualisation of the results, for example via the U-matrix. The U-matrix, together with the weight vectors per parameter (such as in Figure 4.10) provides useful information for someone with knowledge of the subject. However, for someone with limited knowledge about self-organising maps or mathematics in general, they are very hard to interpret. For actual use in work against money laundering, the results must be understandable also for someone with deep domain knowledge but limited mathematical knowledge. For this purpose, the results of a hierarchical clustering could be better. For future work, it would therefore be interesting to see what results one of the fast, hierarchical clustering algorithms that were found suitable in the literature review would yield (i.e. [32] and [33]).

5.5 Conclusions

The purpose of this master's thesis was to explore a selection of features in Handelsbanken's customer database to determine if there exists clusters in it, more specifically if it yielded higher in-cluster compactness than random data. To do this, a literature study was conducted to decide which of the clustering methods available in the liter-

ature was most suitable. The most important constraints that the data imposed on the algorithm was its need to handle mixed type data, handle outliers and being able to find clusters of arbitrary shape. An extension to the self-organising map as well as the k-prototypes algorithm were chosen for this purpose. The algorithms were applied to the real dataset as well as to two synthetic datasets, one consisting of random data and one of synthetic data with clear clusters. The real datasets were shown to be more prone to clustering than the random data, but less than the synthetic cluster data, indicating the existence of clusters in the presence of outliers. It is confirmed that the self-organising map is a more suitable algorithm from a theoretical point of view than the k-prototypes algorithm for the current problem due to its ability to handle outliers and better performance in the categorical dimensions. However, in sense of usefulness for anti-money laundering purposes, the k-prototypes algorithm have advantages based on its transparency and easily interpretable results.

Bibliography

[1] Polismyndigheten och Samverkansradet. *Myndighetsgemensam lägesbild om organiserad brottslighet*. 2017. URL: https://polisen.se/Aktuellt/Rapporter-och-publikationer/Organiserad-brottslighet/Publicerat-Organiserad-brottslighet/Lagesbild-om-organiserad-brottslighet/ (visited on 02/12/2018).

[2] Europol. *Serious and Organised Crime Threat Assessment*. 2017. URL: https://www.europol.europa.eu/socta/2017/ (visited on 02/12/2018).

[3] 2017:630. *Åtgärder mot penningtvätt och finansiering av terrorism*. URL: http://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/lag-2017630-om-atgarder-mot-penningtvatt-och_sfs-2017-630 (visited on 03/13/2018).

[4] Guojun Gan, Ma Chaoqun, and Wu Jianhong. *Data clustering theory, algorithms, and applications*. ASA-SIAM series on statistics and applied probability ; 20. Philadelphia, Pa.: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2007. ISBN: 0-89871-834-1.

[5] J. Macqueen. "Some methods for classification and analysis of multivariate obser- vations". *In Proceedings of the 5th Berkeley symposium on mathematical statistics and probability* 1.2 (1967), pp. 281–297.

[6] Dongkuan Xu and Yingjie Tian. "A Comprehensive Survey of Clustering Algorithms". *Annals of Data Science* 2.2 (2015), pp. 165–193. ISSN: 2198-5804.

[7] Zhexue Huang. "Clustering large data sets with mixed numeric and categorical values". *In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining* (1997), pp. 21–34.

[8] Peter Cheeseman et al. "AutoClass: A Bayesian Classification System". *In Machine Learning Proceedings* (1988), pp. 54–64.

[9] Carmelo Del Coso et al. "Mixing numerical and categorical data in a Self-Organizing Map by means of frequency neurons". *Applied Soft Computing* 66 (2015), pp. 246–254.

[10] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis". *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619.

[11] Saeed Tavazoie et al. "Systematic determination of genetic network architecture". *Nature genetics* 22.3 (1999), p. 281.

[12] JA Saunders. "Cluster analysis for market segmentation". *European Journal of marketing* 14.7 (1980), pp. 422–435.

[13] A. Khashman. "Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes." *Expert Systems with Applications* 37 (9 2010), pp. 6233–6239.

[14] Nhien An Le Khac and M-Tahar Kechadi. "Application of data mining for anti-money laundering detection: A case study". *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*. IEEE. 2010, pp. 577–584.

[15] C. Alexandre and J. Balsa. "Client Profiling for an Anti-Money Laundering System". *arXiv preprint arXiv:1510.00878* (2015).

[16] James E Corter and Mark A Gluck. "Explaining basic categories: Feature predictability and information." *Psychological Bulletin* 111.2 (1992), p. 291.

[17] Kathleen McKusick and Kevin Thompson. "Cobweb/3: A portable implementation" (1990).

[18] Chung-Chain Hsu and Chien-Hao Kung. "Incorporating unsupervised learning with self-organizing map for visualizing mixed data". *Natural Computation (ICNC), 2013 Ninth International Conference on*. IEEE. 2013, pp. 146–151.

[19] Chung-Chian Hsu and Yu-Cheng Chen. "Mining of mixed data with application to catalog marketing". *Expert Systems with Applications* 32.1 (2007), pp. 12–23.

[20] Chung-Chian Hsu and Shu-Han Lin. "Visualized analysis of mixed numeric and categorical data via extended self-organizing map". *IEEE transactions on neural networks and learning systems* 23.1 (2012), pp. 72–86.

[21] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases". *ACM Sigmod Record*. Vol. 25. 2. ACM. 1996, pp. 103–114.

[22] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "Rock: A robust clustering algorithm for categorical attributes". *Information Systems* 25.5 (2000), pp. 345–366.

[23] Jörg Sander et al. "Density-based clustering in spatial databases: The algorithm gdbscan and its applications". *Data mining and knowledge discovery* 2.2 (1998), pp. 169–194.

[24] Wei Wang, Jiong Yang, Richard Muntz, et al. "STING: A statistical information grid approach to spatial data mining". 1997.

[25] Carl Edward Rasmussen. "The infinite Gaussian mixture model". *Advances in neural information processing systems*. 2000, pp. 554–560.

[26] Douglas Fisher. "Knowledge Acquisition Via Incremental Conceptual Clustering". *Machine Learning* 2.2 (1987), pp. 139–172.

[27] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering". *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1 (2009), p. 1.

[28] Sandro Vega-Pons and José Ruiz-Shulcloper. "A survey of clustering ensemble algorithms". *International Journal of Pattern Recognition and Artificial Intelligence* 25.03 (2011), pp. 337–372.

[29] Teuvo Kohonen. "The self-organizing map". *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[30] Gail A. Carpenter and Stephen Grossberg. "The ART of adaptive pattern recognition by a self-organizing neural network". *Computer* 21.3 (1988), pp. 77–88.

[31] Amir Ahmad and Lipika Dey. "A k-means type clustering algorithm for subspace clustering of mixed numeric and categorical datasets". *Pattern Recognition Letters* 32.7 (2011), pp. 1062–1069.

[32] Tom Chiu et al. "A robust and scalable clustering algorithm for mixed type attributes in large database environment". *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2001, pp. 263–268.

[33] Erendira Rendón and José Salvador Sánchez. "Clustering based on compressed data for categorical and mixed attributes". *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer. 2006, pp. 817–825.

[34] Cen Li and Gautam Biswas. "Unsupervised learning with mixed numeric and nominal data". *IEEE Transactions on Knowledge and Data Engineering* 14.4 (2002), pp. 673–690.

[35] Xiao Han et al. "An Improved ART 2-A Model for Mixed Numeric and Categorical Data". *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*. IEEE. 2009, pp. 1–4.

[36] Claudia Plant and Christian Böhm. "INCONCO: interpretable clustering of numerical and categorical objects". *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining*. KDD '11. ACM, 2011, pp. 1127–1135. ISBN: 9781450308137.

[37] Christian Böhm et al. "Integrative parameter-free clustering of data with mixed type attributes". *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2010, pp. 38–47.

[38] Zengyou He, Xiaofei Xu, and Shengchun Deng. "Clustering mixed numeric and categorical data: A cluster ensemble approach". *arXiv preprint cs/0509011* (2005).

[39] Jerome H. Friedman and Jacqueline J. Meulman. "Clustering Objects on Subsets of Attributes". *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 66.4 (2004), pp. 815–849. ISSN: 13697412, 14679868. URL: `http://www.jstor.org/stable/3647651`.

[40] Amir Ahmad and Lipika Dey. "A k-mean clustering algorithm for mixed numeric and categorical data". *Data & Knowledge Engineering* 63.2 (2007), pp. 503–527.

[41] Gail A. Carpenter and Stephen Grossberg. "A massively parallel architecture for a self-organizing neural pattern recognition machine". *Computer vision, graphics, and image processing* 37.1 (1987), pp. 54–115.

[42] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. "Cluster validity methods: part I". *ACM SIGMOD Record* 31.2 (2002), pp. 40–45. ISSN: 0163-5808.

[43] Hans-Hermann Bock. "On some significance tests in cluster analysis". *Journal of classification* 2.1 (1985), pp. 77–108.