# An Evaluation of K-means and EWKM on a High-dimensional dataset

EMIL JUZOVITSKI

Master in Machine Learning
Date: May 12, 2019
Supervisor: Johan Gustavsson
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Host company: Soundtrack Your Brand AB
Swedish title: Detta är den svenska översättningen av titeln

## Abstract

English abstract goes here.

## Sammanfattning

Träutensilierna i ett tryckeri äro ingalunda en oviktig faktor, för trevnadens, ordningens och ekonomiens upprätthållande, och dock är det icke sällan som sorgliga erfarenheter göras på grund af det oförstånd med hvilket kaster, formbräden och regaler tillverkas och försäljas Kaster som äro dåligt hopkomna och af otillräckligt.

# Contents

# Chapter 1

# Introduction

## 1.1 Background

Clustering analysis is an unsupervised method for categorizing a set of data objects. Informally, data objects are grouped by their similarity, with the most similar objects being categorized into the same group, a *cluster*.

There are various methods on how to cluster data. The properties of the to be clustered dataset, is what decides the suitability of a method. The data-types, the size, and, the feature-dimensionality of a dataset are big factors when choosing a method. Picking the right method can be the difference between inadequate clustering performance and stellar performance.

In this thesis we explore the problem of clustering a new real-world high-dimensional dataset consisting of songs, where each song is described through a vector of features. In its raw form the dataset consists of both numerical and categorical data. The dataset is also large, it consists of $5 \cdot 10^7$ songs. The dataset was given by a stakeholder, aiming to find new categorizations of their set of music.

## 1.2 Problem and Research Question

Clustering high-dimensional data suffers from the curse of dimensionality [1, 2, 3]. Any two points with the same cluster membership are bound to have features in which there are large distances between the points [4]. As such, it is hard to assess which points are actually similar as the distance is dominated by dissimilar features which are often irrelevant features.

Traditional methods such as *K-means* do not include any additional steps to cluster high-dimensional data and leaves things desired in terms of performance. The usage of feature reduction techniques can lower the dimensionality but leads to loss of information [5]. Newer algorithms made for the specific purpose of categorizing high-dimensional data have been mentioned to perform better.

In this thesis we compare how the traditional *K-means* algorithm compares to a *soft-subspace* clustering algorithm, *EWKM* chosen specifically for its ability to handle high-dimensional data. The two algorithms are compared on the new real-world dataset of songs.

*How does the performance of EWKM compare to K-means on the given high-dimensional dataset?*

## 1.3 Purpose

The purpose is twofold: Show how soft-subspace clustering methods fair in terms of performance — Convergence Rate and Clustering Accuracy — compared to *K-means*, and determine whether novel song themes can be found through the usage of the clustering methods.

## 1.4 Goal

To adhere to the purpose a traditional clustering algorithm *K-means* is evaluated against a more sophisticated, soft-subspace algorithm *EWKM*. Convergence speeds are compared, The accuracy and novelty of the generated clusters are evaluated by judges — playlist composers.

## 1.5 Ethics and Sustainability

Ethical concerns are commonplace in automatic categorization. In the context of the song dataset, songs from a specific cluster could be used to generate a playlist for a popular streaming platform. That cluster might have it pivotal that an artist's country of origin is a specific country. Artists outside of that country that otherwise fit the cluster, are disregarded due to a seemingly, artificial wall. As such, artists from less established markets can become invincible for that playlist resulting in loss of revenue. Ethically, it is questionable whether county of origin, should have merit or not. From a cluster quality standpoint excluding some songs, for a better precision is a worthy trade-off.

## 1.6 Delimitations

This thesis will focus on the problem of high-dimensionality. Other properties such as mixed data are mentioned however, the algorithms chosen were evaluated on a numerical feature subspace of the dataset. Tackling mixed data through soft-subspace clustering is seen as future work.

# Chapter 2

# Background

Chapter 2. intends to introduce the theory and relevant related work that is necessary to understand the research topic.

## 2.1 Objective of Clustering

Given the inputs of a Dataset $\mathcal{D} = \{X_1, X_2, ..., X_n\}$ — where $X_i = [x_{i1}, x_{i2}, ..., x_{im}]$ is a single data point with $m$ attributes — and $k$ is a positive integer, the objective of partition clustering is to divide objects into $k$ disjoint clusters $C = \{C_1, C_2, ..., C_k\}$ [6].

The above objective is as stated just for partition-based clustering (See 2.3.2) . For density-based clustering the objective is to separate local dense areas from noise for more details (See 2.3.3) [7].

## 2.2 Similarity/Distance Measures

The relative closeness between two points $X_i, X_j$ is quantified numerically through a similarity measure $s(X_i, X_j)$ [8]. A high value indicates that the points are close, while a low measure indicates that the points are dissimilar. Values of $s(X_i, X_j)$ go between 0 and 1.

The dissimilarity between the same two points can be described through a distance measure $d(X_i, X_j)$ — Where, $d(X_i, X_j) \geq 0$. Distance measures are often used with quantitative data i.e. numerical data. Similarity measures are frequently used when data is qualitative e.g. categorical and mixed data [9]. Both measures are symmetric i.e. $s(X_i, X_j) = s(X_j, X_i)$ (An exception is subspace clustering methods - See 2.4 for more details).

Distances can be measured in many ways and are often explicitly created for one data type. Different algorithms use different distance measures. What distance measure an algorithm uses is based on the types of datasets the algorithm is specialized for.

A point $X_i$ can be compared with another point $X_j$ or a cluster representation $C_l$ — frequently used in partitioning algorithms. A representation can be the mean of all points in the cluster, or a centroid — the most central point in the cluster [10, 11]. For categorical data, the *mode* can represent the cluster — The cluster is represented by the most frequent value of each attribute in the cluster [12]. $C_l$ in these representations can be handled as a point.

The sections below introduce some popular approaches for numerical-, categorical-, and, mixed-data.

### 2.2.1 Numerical Similarity Measures

The Euclidean distance is a common measurement for the distance between two vectors, the measurement is shown in 2.1. A frequent choice for distance measurement in clustering analysis is the squared Euclidean [1, 13, 14], shown in 2.2.

$$d(X_i, C_l) = \sqrt{\sum_{j=1}^{m} (x_{ij} - c_{lj})^2} \tag{2.1}$$

$$d(X_i, C_l) = \sum_{j=1}^{m} (x_{ij} - c_{lj})^2 \tag{2.2}$$

The Euclidean distance is a special case of the Minkowski distance where $q = 2$. The Euclidean and other Minkowski distances often involve a preprocessing step of normalizing of features [1]. Features with higher variance will otherwise have a higher contribution in deciding the clusters. Similar to many other measurements, The *curse of dimensionality* diminseshes the ability to diffrentiate distances between points in high-dimsional data[2].

$$d(X_i, C_l) = \sum_{j=1}^{m} \left| x_{ij} - c_{lj} \right|^{q^{\frac{1}{q}}} \tag{2.3}$$

Using a Euclidean distance it is assumed that the covariance matrix is an identity matrix, that is, in a 2-D space you can only cluster points into perfect circles. To avoid this problem Mahalanobis distance can be used, e.g allowing for elliptical shapes in a 2-D plane. It introduces a covariance matrix $S^{-1}$. The algebraic equation is shown in 2.4. Like the Euclidean counterpart, the square root is often discarded.

$$d(X_i, C_l) = \sqrt{(X_i, C_l)^T S^{-1} (X_i, C_l)} \tag{2.4}$$

$$\tag{2.5}$$

The mentioned distances can be converted to a similarity distance — with values between 0 and 1 — by using the Gaussian kernel as shown in 2.6 [15].

$$s(X_i, C_l) = \frac{\exp(-0.5 \cdot d(X_i, C_l))}{\sum_{t=1}^{k} \exp(-0.5 \cdot d(X_i, C_t))} \tag{2.6}$$

### 2.2.2 Categorical Similarity Measures

Categorical data is not commonly referred as a datatype. It is instead a group of datatypes consisting of nominal- and ordinal-data. Unique properties separates the types from each other.

Nominal data — e.g. chemical elements in a periodic table — are either identical or different. Ordinal data — e.g. shirt sizes — can be more or less similar. Additionally, nominal data does not have to be information balanced either — some values are more important than others — e.g. if two songs share the same artist it is more information than if they do not [11].

Most clustering algorithms that cluster categorical data do not take account the differences between the data types mentioned above. There are distance measures that do (see *daisy function* in [11]) but implementing them in a clustering algorithm would increase the complexity of the algorithms. The rest of the similarity measures below considers ordinal data nominal. From this point onward, both nominal and ordinal will be referred as categorical data, and treated as nominal data.

In its purest form a categorical similarity is simply yes or no, as shown in 2.7, i.e. Are two attribute values the same? [10, 11]

To define the similarity between a point and a cluster representation, the number of mismatches can be used as shown in 2.8 [6, 12]. Where a cluster representation is defined by the most frequent attribute values of the points in the cluster. This method is used in *K-modes* and the cluster representation is referred as a *mode*.

Another way is to compare the values of $X_i$ and the values of all data points $\forall X_k \in C_l$ for each attribute as shown in 2.10, 2.11 [10, 15]. In this method a cluster representation is not necessary. To allow fast computations of distances between points and clusters the frequency of each value of each cluster is stored in a frequency matrix. $w_j$ is an attribute weight defining how important each attribute is. The weight is created by calculating the entropy of the attribute. How to calculate the entropy is mentioned in section 2.4.

$$\delta(x_{ij}, x_{kj}) = \begin{cases} 1 & \text{if } x_{ij} \neq x_{kj} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

$$d(X_i, C_l) = \sum_{j=1}^{m} \delta(x_{ij}, C_{lj}) \tag{2.8}$$

$$s(x_{ij}, x_{kj}) = 1 - \delta(x_{ij}, x_{kj}) \tag{2.9}$$

$$s(x_{ij}, C_{lj}) = \frac{\sum_{X_k \in C_l} s(x_{ij}, x_{kj})}{\sum_{X_k \in C_l} [x_{kj} \neq null]} \tag{2.10}$$

$$s(X_i, C_l) = \sum_{j=1}^{m} w_j s(x_{ij}, C_{lj}) \tag{2.11}$$

### 2.2.3 Mixed Similarity Measures

Mixed data measures are simply a combination of numerical and categorical measures. That is, measure categorical attributes with the categorical measure, and measure the numerical data with numerical measure. The bigger question is rather how to weigh the different measures to create a single similarity measure between $X_i$ and, $X_j$ or $C_l$. Weighing specifics is mentioned in section 2.4.

Huang proposes the similarity measure shown in 2.12 [6]. $w_l$ determines how important categorical data is compared to its numerical counterpart for cluster $l$. In *K-Protoypes* a simplification is made local $w_l$ for each cluster $l$ with a single global weight $w$. The weight $w$ is a user-defined input.

Cheung and Jia [15] propose representing numerical data as an vector $X_i^r$ where numerical data is denoted by $r$, while letting each categorical attribute have a single weight . The categorical weights are automatically weighed through information gain, as such the algorithm is a *feature weighting* algorithm. See section 2.4 for more. By defining the numerical attributes as a single numerical vector the amount of attributes is defined as the number of categorical attributes, plus 1. $X_i$.

$$d(X_i, C_l) = \sum_{j=1}^{m_r} (x_{ij}^r - c_{lj}^r)^2 + w_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, c_{lj}^c) \tag{2.12}$$

$$s(X_i, C_l) = \frac{1}{m_f} s(X_i^r, C_l^r) + \frac{m_c}{m_f} \sum_{j=1}^{m_c} w_j s(x_{ij}^c, c_{lj}^c) \tag{2.13}$$

Where:
$m_c$ is the number of categorical attributes
$m_f = m_c + 1$

## 2.3 Clustering Algorithms

Clustering algorithms are often divided into categories. Hierarchical-, Partition- and Density-based clustering are the largest categories [16]. The algorithms below are exclusively numerical if not stated otherwise.

### 2.3.1 Hierarchical Clustering

A Hierarchical algorithm generates a set of nested clusters, also known as a dendogram [1, 16]. There are two approaches two hierarchical clustering.

Agglomerative- and divisive-hierarchical clustering.

In agglomerative clustering, each point starts being its own cluster. In the next step it merges with the most similar cluster. This repeats until all points are in one cluster.

Divisive clustering does instead the opposite: every point starts in the same cluster. In the next step the cluster gets split up into two clusters. This repeats until every point is in its own cluster.

CURE[1], BIRCH[17] and, ROCK (categorical) [10] are popular algorithms of hierarchical clustering. The creation of a dendogram in these clustering algorithms results in a time complexity that make the algorithms less suitable for larger datasets however, with sampling techniques as used in CURE and ROCK bigger datasets can be managed [1, 16]. In addition to creating a dendogram a positive is the possibility to cluster with any arbitrary shape.

### 2.3.2 Partition Clustering

In partition clustering a partition of clusters $U$ is found by iteratively optimizing a cost function [1, 14, 16]. The algorithms include two iterative steps of assigning each point to the most similar cluster center representation, and, updating a cluster center representation — which is a mean in *K-means* and the most central point in *K-medoids*.

The algorithms converge when no data points switch cluster association or on a user defined convergence delta for the cost function. The time complexity is dependent on the amount of iterations. The number of iterations is not known beforehand and could vary depending on the chosen initial cluster centers — A usual approach is to randomly choose $K$ clusters from the dataset as initial clusters centers. Partitioning algorithms handle larger datasets better than most hierarchical approaches.

#### 2.3.2.1 K-means Family

K-means is arguably the most common clustering algorithm. The cost function of which K-means is optimized on is shown in 2.14, 2.15. Either 2.14 is minimized or 2.15 is maximized [14]. There are many algorithms based on K-means, they are referred as the K-means family of clusters [13]. The family includes *K-Modes* [12], *K-Prototypes* [6], *W-K-means* [14], *E-W-K-means* [18].

$$P(U,C) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} u_{il} d(x_{ij}, c_{lj}) \tag{2.14}$$

$$P(U,C) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} u_{il} s(x_{ij}, c_{lj}) \tag{2.15}$$

$U$ is a partition matrix of size $N \times K$. The Partition matrix shows in which clusters point $X_i$ is in. In hard clustering one point can only exist in a single cluster $C_l$. Thus, $u_{il} = [0,1]$ and $(\sum_{l=1}^{k} u_{il}) = 1$. Otherwise, points can be members of multiple clusters with different probabilities that sum up to 1. This can be referred as *Fuzzy-memberships* and often includes a fuzziness index, a variable, deciding the importance of the generated weights [5].

Optimizing the cost function $P(U,C)$ is done by iteratively optimizing the function on one variable and treating the other one as a constant. Our suboptimization problems becomes:

**P1.** Assign the each point to the most similar cluster.

Fix C = $\hat{C}$, Optimize $P(U, \hat{C})$

**P2.** Update the mean for each cluster.

Fix U = $\hat{U}$ Optimize $P(\hat{U}, C)$

For **P1.** we update $U$ by equation 2.16.

$$u_{il} = \begin{cases} 1 & d(X_i, C_l) \leq d(X_i, C_t) \quad \text{for} \quad 1 \leq t \leq k \\ 0 & \text{for} \quad t \neq l \end{cases} \tag{2.16}$$

For **P2.** we update $C$ — updating each center representation — through the equations below:

- For numeric values solved by obtaining the average:

$$c_{lj} = \frac{\sum_{i=1}^{n} u_{il} x_{ij}}{\sum_{i=1}^{n} u_{il}} \tag{2.17}$$

- For categorical values, one option is defining the center as mode [12], which is solved by:

$$c_{lj} = a_j \tag{2.18}$$

  Where: $a_j$ is the most frequent value of attribute $j$ in $C_l$.

- For mixed values, one option is representing the cluster as a Prototype[6, 13]. The solution is simply to use 2.17 for numerical attributes and 2.18 for categorical.

The steps of clustering k-means family algorithms, thus becomes:

1. Choose initial cluster representatives $C^0$
2. Fix $C^t$ = $\hat{C}$, Optimize $P(U^t, \hat{C})$, Obtain $P(U^{t+1}, \hat{C})$

   if $P(U^t, \hat{C}) = P(U^{t+1}, \hat{C})$

      return $P(U^t, \hat{C})$ (Convergence)
3. Fix $U^{t+1}$ = $\hat{U}$, Optimize $P(\hat{U}, C^t)$ Obtain $P(\hat{U}, C^{t+1})$

   if $P(\hat{U}, C^t) = P(\hat{U}, C^{t+1})$

      return $P(\hat{U}, C^t)$ (Convergence)
4. Repeat 2. and 3.

Different similarity functions determine what K-means family algorithm the optimization represents [14]. Euclidean distance would result in *K-means*. The categorical similarity shown in 2.8 results in *K-modes* [12]. If the similarity is the mixed type shown in 2.12 the optimization represents *K-Prototypes* [6].

#### 2.3.2.2 K-medoid Family

In *K-medoids* a cluster is represented by the most central object in a center — A medoid. The cost function looks at the total deviation between points in the cluster and the medoid [19]. Compared to 2.14 the difference is that the distance function compares a point $X_i$ with the medoids $C_l^m$ of that cluster. Using medoids instead of a mean results in a more robust clustering performance. The tradeoff is *K-medioids* runtime. *K-medioids* is usually implemented through the *PAM* algorithm which has a time complexity of $O(k(n -$

$k)^2)$ per iteration, whereas the basic *K-means* has an iteration complexity of $O(nk)$.

Extensions of *PAM*, such as, *CLARA*, and, *CLARANS* are approaches to improve the scalability of the clustering type by clustering on a sample representation which improves the run time [19]. *CLARANS* improves the simple sampling procedure in *CLARA*. It defines a sample of cluster medoids as a node in a graph which is the whole dataset. Neighbouring nodes differ by one medoid. *PAM* traverses all neighbours for each node it traverses to find the node with minimum distance between points. CLARA can be seen as only finding the minimum of a sub-graph containing only the sample points. *CLARANS* samples dynamically neighbours while traversing the graph not restricting the traversal to a subgraph.

#### 2.3.2.3 Determining K and K-initialization

One aspect of using a clustering algorithm that often gets overlooked, is the inputs of the algorithm. In order to use a partitioning algorithm the input of K has to be determined.

A way to determine the K is by running the algorithm with different values of K and then through an internal criterion measure (see 2.5.1) decide the best K for the dataset [6, 20]. Running an algorithm multiple times makes the process of clustering multiple times slower. When clustering a large dataset this is something to avoid.

Sampling is used in CLARA, CLARANS [19] to find medoids for the whole dataset. The idea is that a small($40 + 2k$ points) randomly uniform sample of the dataset should represent the whole dataset. The same sampling process can be used to find *K* and so, finding a $k$ for a sample would be finding an approximation of *K* for the whole dataset. The algorithm still has to be ran on the sample multiple times and evaluated against the inner criterion, but the time to compute the clusters for a sample rather than the whole dataset is significantly less.

Cheung and Jia propose another approach [15, 21]. Here, competitive learning is used — giving every cluster a weight that together with the distance function determines the chance of a datapoint becoming a member of the cluster. When a datapoint is asssigned to the cluster, that cluster's weight and its neighbour's weight increase, and the rest of the clusters' weights decrease. Eventually, some clusters disappear. The positive aspect of this approach is determining the amount of clusters occurs during the clustering process removing the need to cluster the dataset multiple times. Note: A user input of maximum number of clusters $\mathbf{k}^*$ is required.

In addition to choosing $k$, picking good initial points is also a problem that should be considered. Good initial clusters allow for a faster convergence while bad can result in convergence on a suboptimal result[21, 22], forcing multiple clusterings to obtain a result of confidence. While random uniform sampling is a way to initialize $K$ centers there are more robust solutions, that can exclude picking e.g. outliers allowing the result to be less random.

*K-means*++[22] proposes replacing uniformly at random sampling with the following steps:

1. Pick one center $C_l$ uniformly at randomly
2. Pick a new center $C_i$, choosing point $X_i \in \mathcal{D}$ with probability $\frac{d(X_i,C_q)^2}{\sum_{t=1}^{k} d(X_t,C_q)}$, where $C_q$ is the already choosen point with the minimum distance to $X_i$,$X_t$.
3. Repeat 1. and 2. until $K$ points have been picked.

In [21] a specific approach for mixed data is mentioned.

### 2.3.3 Density-Based Clustering

Density-based clustering algorithms define clusters to be higher-density areas — a local area with a relative high number of data points i.e. higher density than noise [1, 7, 14, 16]. DBSCAN [7] is the most well known clustering algorithm where each data point in a cluster must have a user defined *MinPts* in its $\mathcal{E}$-neighborhood, where $d(X_i, X_j) < \mathcal{E}$ and $\mathcal{E}$ is user-defined. The shape of the created clusters is defined by the chosen distance measure.

The algorithms of this type perform well on larger low-dimensional datasets especially on spatial data [7].

## 2.4 Feature-Weighted Clustering

K-means assumes that all attributes/features are of the same importance [11]. If one were to scale the range of one attribute by two, it would become twice as important for the clustering result. To allow attributes of naturally smaller value ranges the same importance as attributes with larger value ranges, attributes are often normalized.

After normalizing the attributes, attributes can be given a weight based on the perceived importance of the attribute. Weighting attributes is necessary for good performance on most datasets. Using irrelevant attributes damages the clustering performance [11]. It is worse than not using the attribute at all. Moreover, increasing the importance of a relevant attribute allows the clustering algorithm to perform better.

Deciding on how to weight attributes is hard. A simple solution is using technical expertise to assign attribute weights. In e.g. data-mining, a dataset is often of high-dimensionality as it may be generated from a database with hundreds of tables and columns [18]. The high degree of dimensions makes manually determining the weights of the attributes almost impossible.

A fairly popular way to handle high-dimensional data is through the use dimensionality-reduction techniques e.g. PCA [23, 24]. This is a possible first step in clustering analysis, occurring before the actual clustering. In short, dimensionality-reduction techniques try to find the minimum set of representative attributes that account for the properties of the dataset. It can be hard to interpret the results from PCA. PCA also assumes that all clusters care about the same features [3]. An assumption that often is false in high-dimensional datasets.

### 2.4.1 Automated Feature-Weighting

Feature-weighting can also be done automatically. Automatic feature-weighting is commonly referred as *feature weighting* and is often achieved by extending a *k-means* like algorithm. An additional *attribute weight* variable is added to the cost function given in 2.14. How the cost function changes from 2.14 depends on what concepts we use to automate the weighting.

One algorithm of this class is *w-k-means* which extends *k-means* by adding a weight to each attribute [14]. *w-k-means* uses the cost function in 2.19 which introduces a new varaible $W$ — the weights of each attribute — to be optimized. $\beta$ is a hyperparamater for the importance of weights. In *w-k-means* $W$ is optimized on the variance of the intra-cluster distances. P3. is the additional problem of optimizing $W$ and is shown in 2.21. An additional step in the k-means algorithm is added for the optimization:

**4.** Let $\hat{U}$ = $U^{t+1}$, $\hat{C}$ = $C^{t+1}$, Optimize $P(\hat{U^{t+1}}, \hat{C^{t+1}}, W^t)$
Obtain $P(\hat{U}, \hat{C}, W^{t+1})$

if $P(\hat{U}, \hat{C}, W^t) = P(\hat{U}, \hat{C}, W^{t+1})$.

return $P(\hat{U^{t+1}}, \hat{C^{t+1}}, W^t)$ (Convergence)

$$P(U, C, W) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} w_j^{\beta} d(x_{ij}, c_{lj}) \tag{2.19}$$

$$D_j = \sum_{l}^{k} \sum_{1}^{n} \hat{u}_{il} d(x_{ij}, c_{lj}) \tag{2.20}$$

$$\hat{w}_j = \begin{cases} 0 & D_j = 0 \\ \frac{1}{\sum_{t=1}^{m} [\frac{D_j}{D_t}]^{\frac{1}{\beta - 1}}} & D_j \neq 0 \end{cases} \tag{2.21}$$

The function can be modified to allow clustering on mixed data by using a suitable distance measurement [21].

Instead of optimizing the weights on only intra-cluster distances — which is tricky for categorical data — The weights can be optimized by information gain, more specifically the entropy. Entropy can be referred as the amount of ”disorder” in a system. Entropy is used in [15, 18]. Using the entropy works for both numerical and categorical data. In [15] the importance of an categorical attribute is defined as the average entropy of each value of the attribute. This is shown in 2.22. To allow weights in the range of $\{0, 1\}$ the importance is divided by the total importance of all attributes in the dataset as shown in 2.23.

$$H_j^c = -\frac{1}{m_r} \sum_{t=1}^{m_r} p(a_{tj}) log(p(a_{tj})) \tag{2.22}$$

$$w_j = \frac{H_j^c}{\sum_{t=1}^{d_c} H_t^c} \tag{2.23}$$

Where: $m_r$ is the number of different values of attribute $j$, and $a_{tj}$ is the t:th value of attribute j and $d_c$ is the number of categorical attributes.

### 2.4.2 Soft-subspace Clustering

The previous mentioned methods make an assumption that all cluster are interested about the same features. This is not inherently true, for example, given clusters as medical deceases, and patients as data-objects, the importance of a certain feature — Recently visited a tropical climate — differs. For Malaria it is of high relevance, while for Acne it is not.

Subspace-clustering allows clusters to have their own feature subspace [18, 21, 25, 26], hence the name. It is a cluster analysis-specific way to deal with high-dimensionality. Irrelevant features can be discarded per cluster resulting in reduce of dimensionality while losing less information than other techniques.

There are two types of subspace-clustering techniques. The fist type is Hard-subspace clustering. Hard-subspace clustering tries to find the exact subspaces [18, 21, 25, 26]. PROCLUS and CLIQUE are two algorithm's of the category. The second type is Soft-subspace clustering (SSC), SSC finds the approximate subspaces of all clusters. Each cluster is given a weight attribute vector, the vector determines the association probability of each feature for that given cluster[27]. Finding exact subspaces is computationally heavy, as such soft-subspace clustering is in general faster than it's counterpart, with a often linear time complexity.

Automated-Weighting Algorithm (*AWA*) [28] is a soft-subspace approach similar to *w-k-means*, the difference is that for the cost function of AWA: $w_j^\beta$ is replaced with $w_{lj}^\beta$ — a weight for an attribute in a cluster $C_l$. The algorithm does not work when the variance of an attribute in a cluster is zero as the learning rules denominator becomes zero[29]. *FWKM* [29] and *FSC* are two alternatives to *AWA* which solve the problem by adding a small value to the distance function, forcing the variance to not be zero. In *FWKM* that value is based on a formula and recalculated during each iteration, in *FSC* the small value is a constant $\epsilon$ determined beforehand. Otherwise the algorithms are equivalent. All three algorithms only look at the intra-cluster distance. The three algorithms are often referred as being *Fuzzy*-weighting-algorithms due to features having different degrees of association in different clusters, and a fuzziness index is used to decide the importance of the weight [5]. Below is the cost function of *FSC*, where $\beta$ is the fuzziness index.

$$P(U,C,W) = \sum_{l=1}^{k} \left[ \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} w_{lj}^{\beta} d(x_{ij}, c_{lj}) + \epsilon \sum_{j=1}^{m} w_{lj}^{\beta} \right] \tag{2.24}$$

$$\textit{where:} \quad \beta \geq 1 \tag{2.25}$$

$$w_{lj} = \frac{1}{\sum_{t=1}^{m} \left[ \frac{D_{lj}+\epsilon}{D_{lt}+\epsilon} \right]^{\frac{1}{\beta-1}}} \tag{2.26}$$

$$\textit{where:} \quad D_{lj} = \sum_{X_i \in C_l} d(x_{ij}, c_{lj}) \tag{2.27}$$

Entropy can also be used in soft-subspace clustering. The main idea is to weigh features in the cluster with respect to the variance of the data within the cluster[4]. The entropy of a weight can then be used to describe the certainty of a feature in the cluster. In Entropy Weighed K-means (*EWKM*) [18] the intra-cluster distance is combined with the negative entropy to create the cost function shown in 2.30. Here, the negative entropy is a regularization term that the algorithm tries to maximize while still minimizing the intra-cluster distance. Unlike, the *Fuzzy*-weighted algorithms, *EWKM* does not need another variable to handle variances of zero. As in *W-k-means* **P3.** and Step **4.** becomes optimizing $W$ for the function while fixing $U$ and $C$. $\gamma$ is a user-inputted variable used to control the size of the weights, For $\gamma > 0$ The smaller $D_{lj}$ the more important attribute $A_j$ is to cluster $C_l$. A modified version of *EWKM* is *IEWKM* [30]. It specifies a cost function for both numerical and categorical data.

$$P(U,C,W) = \sum_{l=1}^{k}\left[\sum_{i=1}^{n}\sum_{j=1}^{m} u_{il}w_{lj}d(x_{ij},c_{lj}) + \gamma\sum_{j=1}^{m} w_{lj}log(w_{lj})\right] \quad (2.28)$$

$$\textit{where:}\quad \gamma \geq 0 \quad (2.29)$$

$$c_{lj} = \frac{\sum_{X_i \in C_l} x_{ij}}{\text{Count}_{X_i \in C_l}} \quad (2.30)$$

$$u_{lj} = \min_{1\leq l\leq k}\Big(\sum_{j=1}^{d} w_{lj}d(x_{ij},c_{lj})\Big) \quad (2.31)$$

$$w_{lj} = \frac{\exp(\frac{-D_{lj}}{\gamma})}{\sum_{t=1}^{m}\exp(\frac{-D_{lt}}{\gamma})} \quad (2.32)$$

$$\textit{where:}\quad D_{lj} = \sum_{X_i \in C_l} d(x_{ij},c_{lj}) \quad (2.33)$$

A recent paper introdcing the *LEKM* algorithm, has modified EWKM [27]. Two problems of EWKM are adressed: The algorithm is very sensitive to $\gamma$, and noicy data. To solve the problems EWKM was modified to use log-transformed distances. The modification allows intra-cluster variance of different features to become smaller and more similar, decreasing the chance of one dominant feature of a cluster. Additionally, centers are set in such fashion that noicy data are less impactful. LEKM is shown below.

$$P(U,C,W) = \sum_{l=1}^{k}\sum_{i=1}^{n} u_{il}\left[\sum_{j=1}^{m} w_{lj}\ln\left[1+d(x_{ij},c_{lj})\right] + \gamma\sum_{j=1}^{m} w_{lj}\ln(w_{lj})\right] \tag{2.34}$$

$$\textit{where:}\quad \gamma \geq 0 \tag{2.35}$$

$$c_{lj} = \frac{\sum_{X_i\in C_l}\left[1+d(x_{ij},c_{lj})\right]^{-1}x_{ij}}{\sum_{X_i\in C_l}\left[1+d(x_{ij},c_{lj})\right]^{-1}} \tag{2.36}$$

$$u_{lj} = \min_{1\leq l\leq k}\left(\sum_{j=1}^{d} w_{lj}\ln\left[1+d(x_{ij},c_{lj})\right] + \gamma\sum_{j=1}^{d} w_{lj}\ln w_{lj}\right) \tag{2.37}$$

$$w_{lj} = \frac{\exp(\frac{-D_{lj}}{\gamma})}{\sum_{t=1}^{m}\exp(\frac{-D_{lt}}{\gamma})} \tag{2.38}$$

$$\textit{where:}\quad D_{lj} = \frac{\sum_{X_i\in C_l}\ln\left[1+d(x_{ij},c_{lj})\right]}{\text{Count}_{X_i\in C_l}} \tag{2.39}$$

WOCIL [21] is a Mixed-data soft-subspace *k-means* type clustering algorithm. To determine $W$ — which in this case is a matrix, due to the method being a subspace method — the inner criterion is used. To determine the inter cluster similarity, the distribution of attribute $A_r \in C_l$ is compared to the distribution of $A_r$ outside of $C_l$. In this case, Hellinger distance is used the quantify the dissimilarity between the two distributions. For categorical data the distribution is assumed as 2... and for numerical attributes a Gaussian distribution is assumed. The intra-cluster similarity is then found through 2.41.

$$M_{lj} = \frac{1}{\sum_{X_i\in C_l}1}\sum_{X_i\in C_l} s(x_{ij},C_l) \tag{2.40}$$

$$H_{lj} = F_{lj}M_{lj} \tag{2.41}$$

$$w_{lj} = \frac{H_{lj}}{\sum_{t=1}^{m}H_{lt}} \tag{2.42}$$

## 2.5 Evaluation

There are two main ways of which clustering are evaluated: Internal- and external-criterion.[31] In some research the categories are extended with the relative-criterion.[32]

### 2.5.1 Inner Criteria

Internal criterion is an unsupervised validation approach and can be described as evaluating the results without respect to external information [32]. An internal criterion tries to verify the objective of the clustering algorithm on the dataset. For example: Making sure that points assigned to the same cluster are in general more similar than points outside of the cluster.

#### 2.5.1.1 Average Silhouette Coefficient

The average silhouette coefficient upon all datapoints shown in 2.47, is one way to evaluate the internal criteria [33]. A single silhouette coefficient shown in 2.46, looks at a data points' intra-cluster similarity — similarity to points within the same cluster, and, inter-cluster similarity — similarity with point outside of the cluster. $\overline{s}_{co}$ goes between -1 and 1 where a high value (close to 1) indicates a natural clustered dataset.

When $s_{co}(X_i)$ is a high value (close to 1) a point is well clustered i.e. the intra-cluster similarity is high relative to the inter-cluster similarity [33] . The same reasoning is extended to $\overline{s}_{co}(\mathcal{D})$ where a high value is a well clustered dataset.

$$d_{avg}(X_i, C_l) = \frac{\sum_{X_k \in C_l} d(X_i, X_k)}{Count(X_k \in C_l)} \tag{2.43}$$

$$a(X_i) = d_{avg}(X_i, C_a)\text{ , } \textit{where } (X_i \in C_a) \tag{2.44}$$

$$b(X_i) = min_{C \neq C_a}(d_{avg}(X_i, C)) \tag{2.45}$$

$$s_{co}(X_i) = \frac{b(X_i) - a(X_i)}{max(a(X_i), b(X_i)} \tag{2.46}$$

$$\overline{s}_{co}(\mathcal{D}) = \frac{\sum_{i=1}^{N} s_{co}(X_i)}{N} \tag{2.47}$$

#### 2.5.1.2 Calinski-Harabasz Index

Calinski-Harabasz is for measuring the validity

### 2.5.2 External Criteria

External criterion is a supervised validation approach and can be described by the following sentence: Validation of the results by imposing a pre-defined structure on the dataset i.e. data not used for generating the clustering results [32].

The external criterion requires validation data in addition to an external measurement. Validation data could be a sample of the dataset that is labeled with a class i.e. a ground truth. If the dataset is already labeled by a feature and that feature is not used for clustering, then that dataset can be easily evaluated based on that feature through a external measurement.

In many cases, the clustered data is not labeled — A reason why, an unsupervised approach was chosen in the first place. Even if a label exists it might not be a goal for the research to exclusively produce results that evaluate well to the validation data. There could be a hope to find *new* classes. The validation data can in these cases be replaced by expertise — a panel of judges with knowledge of the data [31]. A sample of clusters can then be given to the judges to assess. Combining the judges with a measurement tool allows the dataset to be given a score that corresponds to the external validity of the dataset.

#### 2.5.2.1 External Measurements

Purity is one measurement for the external criterion. It measures the mean ratio of the most dominant class in each cluster. The data-points are labeled with a class beforehand [31]. Equation 2.48 defines the measurement. The measurement is easy to compute. A downside is that the equation does not penalize small clusters as such small clusters produces a high score.

$$P(C, \Psi) = \frac{1}{n} \sum_{l=1}^{k} \max_{1 \leq j \leq t} \left| C_l \cap \Psi_j \right| \tag{2.48}$$

Where:
$C = \{c_1, c_2, ..., c_k\}$ is the set of clusters, and, $\Psi = \{\Psi_1, \Psi_2, ..., \Psi_t\}$ is the set of *truth*-classes.

$F$*-measure* is another measurement for the evaluation of the external criteria. The measurement is shown in 2.51[31]. The resulting clusters of an algorithm are seen as a series of decisions on each pair of data-points in the set.

An ideal clustering in this reasoning would mean all similar points are within the same cluster i.e. only *True Positives* and no *False Positives*. In reality, some non-similar pairs are clustered together leading to *False Positives*. Similarly, *True Negatives* and *False Negatives* can exist.

The terms can be combined to create the *precision* and *recall*. The precision defines the ratio of *True Positives* on all positives — pairs in the same cluster. The precision is shown in 2.49. The recall on the other hand measures the ratio of how many similar pairs have been clustered together given all possible similar pairs.

$\beta$ in 2.51 determines in what ratio *precision* and *recall* should be taken account to. $\beta < 1$ results in precision being more important and $\beta > 1$ results in more importance to the recall. The balanced $F$*-measure* is called the $F_1$-*measure* and is defined in 2.52. It weighs the impact of precision and recall the same. The $F$*-measure* is common in information retrieval. It is however, more complex *Purity* and requires more effort to implement.

$$P = \frac{TP}{TP + FP} \tag{2.49}$$

$$R = \frac{TP}{TP + FN} \tag{2.50}$$

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R)}{\beta^2 \cdot P + R} \tag{2.51}$$

$$F_{\beta=1} = \frac{2 \cdot P \cdot R}{P + R} \tag{2.52}$$

Where:
$TP$ are true positives, $FP$ false positives, $FN$ false negatives, and, $\beta$ is a weight between $P$ and $R$, a $\beta > 1$ emphasizes the $R$.

## 2.6 Chapter Summary

The fundamentals of clustering have been introduced in this chapter, distance measures are described as well as how they correlate to different datatypes. Hierarchical-, Partition- and, Density-clustering are all described. Figure 2.1 summarizes the properties of all the mentioned algorithms in this chapter. The pre-study goes into detail on how the popular *K-means* algorithm functions as it is pivotal for understanding weighted clustering.

The findings of the prestudy suggests that feature-weighted clustering can be used to attack the challenging problem of clustering high-dimensional data. Algorithms such as *Weighted K-means*, *FSC*, and, *Entropy Weighted K-means* have all been introduced. The mentioned algorithms manage high-dimensional data clustering in different ways. The last two, are examples of soft subspace clustering — an extension of *feature weighting*, which allows each cluster an individual subspace. The soft-subspace methods have been shown to deal with high-dimensional data better than traditional algorithms while still allowing the performance similar to traditional partitioning algorithms.

As for evaluation, it is necessary to use both an internal and external criteria. While an internal criteria can give a score of inter- and intra-cluster performance through methods such as the silhouette coefficient it cannot replace a ground truth, and should instead be used as a tool for parameter tuning. The supervised method of using an external criteria should instead be used. Data can be tested against a ground truth label or a panel of judges through measurements like, the mentioned $F_1$-Score.

The clustering method for this thesis will be to use EWKM algorithm on the given dataset. The first reason for choosing the algorithm is that it is a soft-subspace clustering algorithm. Secondly, it takes into account the information gain in the cost function. To evaluate the performance, the algorithm will be compared with the traditional *K-means* algorithm. The evaluation will compare the average silhouette coefficient for a chosen *K-parameter* on the two algorithms results. The results will also be evaluated by an external criteria of $Purity$ basing the ground truth on the genre feature which is a given label for all songs in the dataset.

The chosen methods, are possible to implement for mixed data, but are available online as numerical methods. From a thesis standpoint, tackling the high-dimensionality of a dataset was the main priority. As such, only numerical features of the dataset can be used — See Dataset in Chapter 3. For more details.

Table 2.1: Properties of algorithms discussed in this chapter [9, 25]

| Type | Algorithm | Properties | | |
|---|---|---|---|---|
| | | Time Complexity | Data Type | Weighted |
| Hierarchical | CURE | $O(n^2 log(n))^*$ | Numerical | |
| | BIRCH | $O(n)^*$ | Numerical | |
| | ROCK | $O(n^2 \cdot log(n))^*$ | Categorical | |
| Density | DBSCAN | $O(n^2)^*$ | Numeric | |
| Partition | K-means (Lloyd) | $O(tnkm)$ | Numerical | |
| | PAM | $O(tk(n-k)^2)^*$ | Numerical | |
| | CLARA | $O(t(k(40+k)^2 + k(n-k)))^*$ | Numerical | |
| | CLARANS | $O(n^2)^*$ | Numerical | |
| | K-modes | $O(tnkm)$ | Categorical | |
| | K-Prototypes | $O(tnkm)$ | Mixed | Yes[a] |
| | OCIL | $O(tnkm)$ | Mixed | Yes[a] |
| | WKM | $O(tnk + tkm + tm)$ | Numerical[b] | Yes[c] |
| | FSC | $O(tnk + tk + tkm)$ | Numerical[b] | Yes[d] |
| | EWKM | $O(tnk + 2tkd)$ | Numerical[b] | Yes[e] |
| | WOCIL | $\approx O(tnkm)$ | Mixed | Yes |

[a] Not all features have independent weights.
[b] Has been modified to allow mixed data.
[c] Only Feature weighting.
[d] Fuzzy-Weighting, Within-distance
[e] Entropy-Weighting, Within-distance
* Dimensionality disregarded in complexity notation

# Chapter 3

# Method

The method chapter subjects the reader to the framework used to qualitatively compare the algorithms with regards to the given dataset.

## 3.1 Method Design Overview

In order to generate results and compare the two different algorithms, a method design was planned and executed. The overall design is shown in 3.1 it includes the step of Preprocessing, Implementation, Result Generation and External Evaluation.

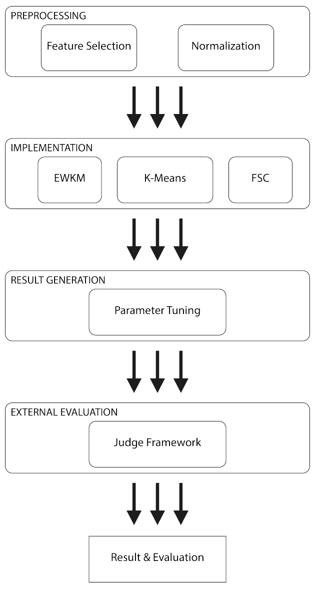


Figure 3.1: Method Design

## 3.2 Preprocessing

### 3.2.1 Dataset Properties

The given dataset is a real-world-, high-dimensional- and, mixed-dataset. It is a set of *songs* extracted from an company-internal music database. Each song is a vector of features — each feature describes the song in a unique way. The vector consists of two different types of features; General metadata and metadata generated from an audio embedding.

The general metadata, includes attributes that predominantly identify the song without resulting to audio analysis. Features of the type were e.g. *Album*, *Artist*, *Title* and *Year of Origin*. The type also includes *BPM* and *Energy-feel*. These features include both numerical and categorical features — features stored as strings can be converted to categorical data.

The second feature type is a 160-dimensional vector embedding — with numerical values ranging through -1 and 1 — that characterizes the song through audio analysis. The embedding is generated through a neural network of auto-encoders, trained on the Mel-frequency cepstrum coefficients(MFCC)[34]. The feature type is anonymous to humans, i.e. what exactly dimension 5 represents is unknown.

There is also feature named *Genre* which holds a list of human-tagged genres that a song fits to. In total 33 genre tags exist in the dataset. A minority of the songs do not hold a single genre.

### 3.2.2 MFCC

To understand MFCC we need to first understand how humans perceive sound and how speech is created. Humans are more sensitive to frequency differences at lower frequencies compared to higher frequencies. The Mel-scale takes the differences in sensitivity into account by creating a logarithmic scale. Human speech is generated by the shape of the vocal tract. The shape determines what sound is created. A power spectrum describes how much power is in the frequency component of a signal. The envelope of the power spectrum can be used to represent the shape of the vocal tract of the given audio. An MFCC represents the envelope [34]. With the shape of the tract represented, generated phonemes can be determined of the vocals of the audio.

### 3.2.3 Feature-Selection and Normalization

With a focus on tackling the problem of high-dimensionality, the choice was to use the numerical methods of *EWKM*, *FSC*, and, *K-means*. That left a restriction to only select numerical features as input. A hypothesis was that audio-embeddings can exclusively describe the songs in the dataset sufficiently. The dataset was therefore preprocessed to only include the subspace of embedding-features.

The *genre* feature was additionally kept as an approximate ground truth label for further use in the external criteria.

As a next step in preprocessing, all features were normalized to the same variance, a way to make sure that all embeddings have the same impact on the algorithm. For normalization Z-score was used ( See 3.1 ).

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

## 3.3 Implementation

### 3.3.1 EWKM Implementation

The chosen EWKM algorithm is available in *R and CSRAN* through the *wskm* package [35]. The source code is available for the public on github. The algorithm is implemented in $C$ and wrapped for $R$.

The implementation differs slightly in how $w_{lj}$ is calculated from the algorithm shown in 2.30. It includes additional normalization steps. Eq 3.4 shows how $w_{lj}$ is updated in the given implementation.

Another deviation from the original algorithm is how empty clusters are treated. In this implementation cluster center's are re-sampled if the originally picked centers result in empty clusters i.e. the algorithm is re-ran if empty clusters are occurring. Additionally a convergence value can be set by the user.

Per request of the stakeholder along with the author's familiarity with Python and Pandas DataFrame, the code was re-wrapped for Python using numpy-C-API [36] — allowing for the dataset to be sent in to the algorithm as a numpy array. The C-code was tweaked to not be dependent on R's *unif_rand() function*. The altered C-code along with the Python wrapping can be found in the attachments....

$$\lambda_{lj} = \exp\left(\left(\frac{-D_{lj}}{\gamma}\right) - \max_{\forall t \in C_l}\left(\frac{-D_{lt}}{\gamma}\right)\right) \tag{3.2}$$

$$\lambda_{lj} = \max\left(\frac{\lambda_{lj}}{\sum_{t=1}^{m} \lambda_{lt}}, \frac{0.0001}{m}\right) \tag{3.3}$$

$$w_{lj} = \frac{\lambda_{lj}}{\sum_{t=1}^{m} \lambda_{lt}} \tag{3.4}$$

### 3.3.2 FSC Implementation

The FSC implementation was created by the thesis author and is based on the C-code of *EWKM*. The methods of calculating cost, updating weights and updating cluster memberships were modified to correspond to the FSC algorithm as shown in 2.24, and 2.26. $\gamma$ is replaced by $\beta$, and the small value of $\epsilon$ was set to $0.001$ as proposed in [5].

### 3.3.3 K-means Implementation

*EWKM* is equivalent to *K-means* when $\lim_{\gamma \to 0}$, although simply setting $\gamma = 0$ forces division by zero. As such it was simpler to use another package for *K-means*. *K-means* being a popular clustering algorithm, is widely available for Python. The PyClustering implementation of *K-means* was used for this report [37]. The implementation was created in *C*++ and wrapped for Python.

## 3.4 Result Generation: Parameter Tuning

### 3.4.1 Purity

As stated above the *genre* feature was kept. With it an external criteria could be created, the measurement of purity was combined with the feature to create a external criteria. The most dominant genre of each cluster was aggregated and divided by the total amount of songs in the dataset.

It is important to highlight the problem of only relying on this criteria to evaluate the results. A good song cluster does not have to be genre homogenic. A good cluster could hypothetically be Christmas music and hold multiple genres, this would then be discarded by the evaluation as a bad cluster. The evaluation was tehrefore not used, for the final evaluation. It was however, deemed as a suitable score for the processes of parameter tuning.

### 3.4.2 Average Silhouette

The internal criterion of average silhouette can be used for parameter tuning. Soft-subspace clustering requires the creation an asymmetric distance matrix, as the distance from *Object A* to *Object B* is dependent on the subspace weights of the cluster in which *Object B* resides in.

1000 objects were sampled from the result to generate the asymmetric distance matrix. From there, the *sklearn* package was used for average silhouette calculation.

### 3.4.3 Tuning K - Amount of Clusters

$k$ could have been decided on a smaller sample of the dataset, or through the use of a competitive learning strategy as discussed in the background. Due to the already relative small sample of the whole dataset and an algorithm that is linear in time it was decided to decide $k$ based on the results of the whole sample dataset with respect to the best $\gamma$ in regards to the cost function.

$k$ was checked for the values of 50, 100, 500, 1000 and compared through the average silhouette . It was not important for this project to get the exact best $k$ rather what approximately range $k$ should be picked. The minimum of $50$

clusters was based on that the number of genres in the dataset was 33, and generated clusters should at least be as specific as a genre label. On the other side of the scale, 1000 was decided as the maximum amount of clusters as more would result in an average less than 50 songs per cluster. Additionally, if you were to scale the dataset to $50 * 10^7$ songs, more cluster would translate into more expensive computations.

### 3.4.4 EWKM

$\gamma$ is a hyper-parameter of EWKM. In short, $\gamma$ determines the likeliness to cluster on more or less features. For any given dataset the ideal $\gamma$ can vary. An ideal $\gamma$ is found when there is no other $\gamma$ value that can result in a lower cost.

The EWKM algorithm does not try to maximize the inter-cluster distance between clusters. The best $\gamma$ in regards to a minimized intra-cluster distance and a maximized inter-cluster distance — Which can be expressed by the average silhouette — does not have to equal the $\gamma$ which produces the lowest cost function. To attest for the difference the result of the best $\gamma$ in regards to the average silhouette was compared with the best $\gamma$ in regards to the cost function. Discarding any of the two before comparison could lead to sub-optimal results.

$\gamma$ was chosen by iteratively testing different values of the parameter as an input for the algorithm, from small ($1 * 10^{-3}$) to large ($\gamma = 3$). The results on the sample dataset – of the different $\gamma$ values — was compared in regards to silhouette and purity.

### 3.4.5 Ranking songs and clusters

A distance array — based on the subspace distance measure of the algorithm —- between each point and the cluster center in which the point is a member in was generated. Songs within a cluster could then be ranked by the ascending distance to the center. Clusters could also be ranked by representing the cluster *"goodness"* through the ascending average distance of the songs. Here, it was deemed that good clusters should not be picked based on genre purity.

The ranking allowed for a cutoff in songs and clusters i.e. the top 10 clusters could be chosen and the clusters and from there a cluster could be shrank by only picking songs with a certain distance to the cluster center.

### 3.4.6 Judge Evaluation

Purity using the genre as benchmark, assumes that a good cluster is genre coherent. While it could be true, a good cluster could be cross-genre such as Christmas music. To adhere

A website page was created where judges could listen to songs of a cluster and rank it. Each cluster could be ranked by cohesion (similarity between songs) and novelty (if the cluster was generating a new type of mini-genre) on a scale 1-10.

Ten songs were sampled on a half-norm distribution — based on the songs distances. A blind test was created, in which five songs of each algorithm was picked. To allow a reference point, three clusters were generated from previously created playlists. Judges were to choose the novelty and cohesion of each cluster.

The website was built in django and bootstrap. A screenshot can be found in appendix....

# Chapter 4

# Results

## 4.1 Comparision Between SSC Algorithms

EWKM, LEKM, and, FSC were compared on their best hypterparameters according to the cost function on different values of *k* (50, 100, 500). Figure **??** shows the weight grid of EWKM on different values of *k*. Figure **??** shows the weight grid of EWKM on different values of *k*. Figure **??** shows the weight grid of EWKM on different values of *k*.

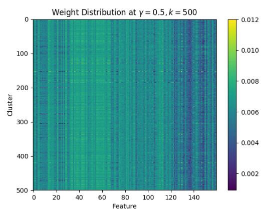
Table ... shows the different purity of the different algorithms on different *k*

### 4.1.1 EWKM

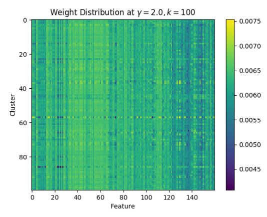
EWKM was first ran on between values of $\gamma = [0.5 - 3]$, the recommended range as described in [18, 35]. The negative entropy ended up being larger than the cost function resulting in immediate convergence.

On the next iteration of tests, $\gamma$ was lowered until immediate convergence did not occur, the best $\gamma$ parameter and purity is shown in . The resulting Purity is lower than other algorithms for all $k$. All tested $\gamma$'s resulted in a weight distribution, in which all clusters only had one significant dimension.

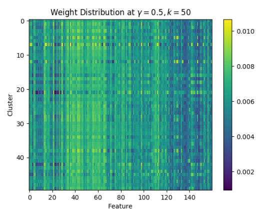
Table ... shows diff



(a) $k = 500$, $\gamma = 2$

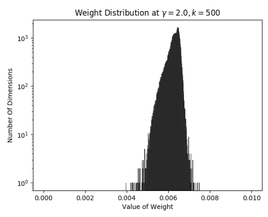


(b) $k = 100\gamma = 2$

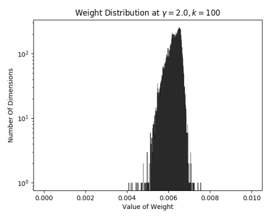


(c) $k = 50\gamma = 0.5$

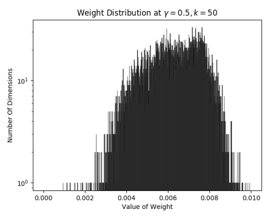
Figure 4.1: Three simple graphs



(a) $k = 500$, $\gamma = 2$



(b) $k = 100\gamma = 2$



(c) $k = 50\gamma = 0.5$

Figure 4.2: Three simple graphs

### 4.1.2 LEKM

LEKM was ran on $\gamma$'s ranging from 0.5 to 3. The LEKM did not have the same problem as EWKM on the given range. The best performing gamma is shown in ... . The scores were comparable to FSC. All $\gamma$'s resulted in normal distribution of feature weights. Higher values of $\gamma$ resulted in a smaller standard deviation, leading to a more uniform distribution.

Table ... shows diff

### 4.1.3 FSC

$\beta$ was recommended to be set to 2.1 and $\epsilon$ to 0.0001 [5]. In our tests, we tested $\beta$'s between 1.5 and 30 with $\epsilon$ set to 0.0001. Based on purity $\lim_{\beta}$ inf led to the best score. The score of FSC was better than EWKM and similar to LEKM. $\beta = 1.5$ led to a single dominant feature, similar to EWKM. For

Table ... shows diff

| | EWKM | | LEKM | | FSC | |
|---|---|---|---|---|---|---|
| $k$ | $\gamma$ | Purity | $\gamma$ | Purity | $\beta$ | Purity |
| 50 | 0.005 | 28.2% | 0.0 | 0.0 | 2.1* | 0.0 |
| 100 | 0.005 | 28.9% | 2.1 | 40.6% | 2.1* | 0.0 |
| 500 | 0.001 | 28.6% | 2.2 | 45.2% | 2.1* | 43.8 |

Table 4.1: Best hyperparameter ($\gamma$, $\beta$) given $k$

## 4.2 K-means

## 4.3 External Evaluation

8 contestents were

# Chapter 5

# Discussion

# Chapter 6

# Conclusions

# Bibliography

[1] A K Jain, M N Murty, and P J Flynn. “Data Clustering: A Review”. In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. ISSN: 0360-0300.

[2] Lance Parsons, Ehtesham Haque, and Huan Liu. “Subspace Clustering for High Dimensional Data: A Review”. In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 90–105. ISSN: 1931-0145.

[3] Zhaohong Deng et al. “Enhanced soft subspace clustering integrating within-cluster and between-cluster information”. In: *Pattern Recognition* 43.3 (2010), pp. 767–781. ISSN: 0031-3203.

[4] Carlotta Domeniconi et al. “Locally adaptive metrics for clustering high dimensional data”. In: *Data Mining and Knowledge Discovery* 14.1 (Feb. 2007), pp. 63–97. ISSN: 1573-756X.

[5] Guojun Gan, Jianhong Wu, and Zijiang Yang. “A Fuzzy Subspace Algorithm for Clustering High Dimensional Data”. In: *Advanced Data Mining and Applications*. Ed. by Xue Li, Osmar R Zaïane, and Zhanhuai Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 271–278. ISBN: 978-3-540-37026-0.

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

[7] Martin Ester et al. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Tech. rep. 1996.

[8] Edwin Diday and J C Simon. “Clustering analysis”. In: *Digital pattern recognition*. Springer, 1976, pp. 47–94.

[9] D Wunsch. “Survey of clustering algorithms”. In: *IEEE Transactions on Neural Networks* 16.3 (May 2005), pp. 645–678. ISSN: 1045-9227.

[10] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. “Rock: a robust clustering algorithm for categorical attributes”. In: *Information Systems* (2000). ISSN: 03064379. arXiv: `arXiv:1011.1669v3`.

[11] Leonard. Kaufman and Peter J. Rousseeuw. “Introduction”. In: *Finding Groups in Data*. John Wiley & Sons, Ltd, 1990. Chap. 1, pp. 1–67. ISBN: 9780470316801.

[12] M K Ng. “A fuzzy k-modes algorithm for clustering categorical data”. In: *IEEE Transactions on Fuzzy Systems* 7.4 (Aug. 1999), pp. 446–452. ISSN: 1063-6706.

[13] Zhexue Huang. “Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values”. In: *Data Mining and Knowledge Discovery* 2.3 (Sept. 1998), pp. 283–304. ISSN: 1573-756X.

[14] Joshua Zhexue Huang et al. “Automated variable weighting in k-means type clustering”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2005), pp. 657–668.

[15] Yiu-ming Cheung and Hong Jia. “Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number”. In: *Pattern Recognition* 46.8 (2013), pp. 2228–2238. ISSN: 0031-3203.

[16] Dongkuan Xu and Yingjie Tian. “A Comprehensive Survey of Clustering Algorithms”. In: *Annals of Data Science* 2.2 (June 2015), pp. 165–193. ISSN: 2198-5812.

[17] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012, pp. 135–156. ISBN: 9781439830031.

[18] L Jing, M K Ng, and J Z Huang. “An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.8 (Aug. 2007), pp. 1026–1041. ISSN: 1041-4347.

[19] Raymond T Ng and Jiawei Han. “CLARANS : A method for clustering objects for”. In: *IEEE Transaction on Knowledge and Data Engineering* (2002).

[20] Catherine A Sugar and Gareth M James. “Finding the Number of Clusters in a Dataset”. In: *Journal of the American Statistical Association* 98.463 (2003), pp. 750–763.

[21] Hong Jia and Yiu Ming Cheung. “Subspace clustering of categorical and numerical data with an unknown number of clusters”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3308–3325. ISSN: 21622388.

[22] David Arthur and Sergei Vassilvitskii. *k-means++: The Advantages of Careful Seeding*. Technical Report 2006-13. Stanford InfoLab, June 2006.

[23] Ian Jolliffe. “Principal Component Analysis”. In: *Encyclopedia of Statistics in Behavioral Science*. American Cancer Society, 2005. ISBN: 9780470013199.

[24] Laurens Van Der Maaten, Eric Postma, and Jaap den Herik. “Dimensionality reduction: a comparative”. In: (2009).

[25] Zhaohong Deng et al. “A survey on soft subspace clustering”. In: *Information Sciences* 348 (2016), pp. 84–106. ISSN: 0020-0255.

[26] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. “Subspace clustering”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.4 (2012), pp. 351–364.

[27] Guojun Gan and Kun Chen. “A soft subspace clustering algorithm with log-transformed distances”. In: *Big Data and Information Analytics* 1.1 (Sept. 2015), pp. 93–109. ISSN: 2380-6966.

[28] Elaine Y Chan et al. “An optimization algorithm for clustering using weighted dissimilarity measures”. In: *Pattern Recognition* 37.5 (2004), pp. 943–952. ISSN: 0031-3203.

[29] Liping Jing et al. “Subspace Clustering of Text Documents with Feature Weighting K-Means Algorithm”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Tu Bao Ho, David Cheung, and Huan Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 802–812. ISBN: 978-3-540-31935-1.

[30] T Li and Y Chen. “A Weight Entropy k-Means Algorithm for Clustering Dataset with Mixed Numeric and Categorical Data”. In: *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 1. Oct. 2008, pp. 36–41.

[31] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. “Introduction to information retrieval”. In: *Natural Language Engineering* 16.1 (2010), pp. 100–103.

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

[33] Peter J Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427.

[34] Kuldip K Paliwal and Kaisheng Yao. "Chapter 6 - Robust Speech Recognition Under Noisy Ambient Conditions". In: *Human-Centric Interfaces for Ambient Intelligence*. Ed. by Hamid Aghajan, Ramón López-Cózar Delgado, and Juan Carlos Augusto. Oxford: Academic Press, 2010, pp. 135–162. ISBN: 978-0-12-374708-2.

[35] Graham Williams et al. *wskm: Weighted k-Means Clustering*. 2015.

[36] *NumPy C-API — NumPy Manual*.

[37] Andrei Novikov. "PyClustering: Data Mining Library". In: *Journal of Open Source Software* 4.36 (Apr. 2019), p. 1230.

# Appendix A

# Something Extra