A Comparision Between K-means and Soft-Subspace Clustering Algorithms on a High-dimensional Musical Dataset

EMIL JUZOVITSKI

Master in Machine Learning
Date: June 6, 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

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Problem and Research Question . . . 2
- 1.3 Purpose . . . 3
- 1.4 Goal . . . 3
- 1.5 Ethics and Sustainability . . . 3
- 1.6 Delimitations . . . 3

2 Background **4**
- 2.1 Objective of Clustering . . . 4
- 2.2 Similarity/Distance Measures . . . 4
  - 2.2.1 Numerical Similarity Measures . . . 5
  - 2.2.2 Categorical Similarity Measures . . . 6
  - 2.2.3 Mixed Similarity Measures . . . 8
- 2.3 Clustering Algorithms . . . 8
  - 2.3.1 Hierarchical Clustering . . . 9
  - 2.3.2 Partition Clustering . . . 9
  - 2.3.3 Density-Based Clustering . . . 13
- 2.4 Feature-Weighted Clustering . . . 14
  - 2.4.1 Automated Feature-Weighting . . . 14
  - 2.4.2 Soft-subspace Clustering . . . 16
- 2.5 Evaluation . . . 20
  - 2.5.1 Inner Criteria . . . 20
  - 2.5.2 External Criteria . . . 21
- 2.6 Summary and Method Justification . . . 25

3 Method **26**
- 3.1 Method Design Overview . . . 27
- 3.2 Dataset Properties . . . 27

3.3 Pre-processing 28
3.4 Implementation 28
3.4.1 EWKM 28
3.4.2 FSC 29
3.4.3 LEKM 29
3.4.4 K-means 30
3.5 Parameter Selection 30
3.5.1 Convergence Rate 30
3.5.2 Purity 30
3.5.3 Amount of Clusters 30
3.5.4 EWKM 31
3.5.5 FSC 31
3.5.6 LEKM 31
3.5.7 K-means 32
3.5.8 Ranking songs and clusters 32
3.6 Expert Evaluation 32

4 Results **35**
4.1 General Results 35
4.1.1 EWKM 35
4.1.2 LEKM 39
4.1.3 FSC 42
4.1.4 K-means 42
4.1.5 Summary 42
4.2 External Evaluation 43

5 Discussion **46**
5.1 Quantitative results 46
5.2 Qualitative results 47
5.3 Future Work 48

6 Conclusions **50**

Bibliography **51**

Chapter 1

Introduction

Clustering analysis is the exploratory art of finding clusters(groups) in a dataset [1]. Informally, data objects are grouped by their similarity, with the most similar objects being categorized into the same group, a *cluster*.

There are many applications for clustering. Two examples are: Deciding what items to put in the same aisle in a shop in order to increase revenue, and, categorizing insurance holders to into risk groups, to provide the best price to each customer. The items and the insurance holders are the objects of the dataset that are the subjects for clustering. Topics that help describe a data object e.g. *cost*, *brand*, and *origin*, are often referred as features or attributes.

Data objects with similar feature properties are partitioned into the same cluster. For datasets with one- or two-features (dimensions) it may be possible to determine by-eye, which objects are similar and should be grouped together. Other datasets could need tens- or hundreds- of features to describe a single object. Here, it is almost impossible to tell by-eye how to cluster the dataset as it becomes hard to represent feature-spaces with a degree higher than three.

In clustering analysis, methods are instead able to automatically determine partitions by expressing the similarity of objects as a numerical value.

1.1 Background

There are various methods on how to cluster data. The properties of the dataset to be clustered, 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 [2, 3, 4]. Any two points with the same cluster membership are bound to have features in which there are large distances between the points [5]. 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 [6]. Newer algorithms made for the specific purpose of categorizing high-dimensional data have been mentioned to perform better.

In this thesis we study how the traditional *K-means* algorithm compares to *soft-subspace* clustering(SSC) algorithms. The two types of clustering algorithms are compared on the new real-world dataset of songs.

There are two problem statements of the thesis.

To answer how $K - means$ compare to SSC algorithms, an SSC algorithm has to be chosen and so the first problem statement becomes:

What is a suitable soft-subspace algorithm for the given dataset?

With the first statement answered the second and the main problem statement can be answered:

How does the performance of a soft-subspace clustering algorithm 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 — 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 more sophisticated, soft-subspace algorithms. 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 focuses 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 out of the scope of the project as the core features of the dataset are numerical.

Chapter 2

Background

The fundamentals of clustering are introduced in Chapter 2. Distance measurements of varying data-types are shown. Hierarchical-, Partition- and, Density-clustering are all described. The pre-study goes into detail on how the popular *K-means* algorithm functions. It continues by mentioning different feature-weighted clustering algorithms including *Soft Subspace Clustering* (SSC) algorithms. The chapter ends with a method justification.

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 $C = \{C_1, C_2, ..., C_k\}$ [7].

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) [8].

2.2 Similarity/Distance Measures

The relative closeness between two points X_i, X_j is quantified numerically through a similarity measure $s(X_i, X_j)$ [9]. 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 [10]. 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 [1, 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 [2, 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 [2]. 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[3].

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

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 [1].

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 [1]) 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? [1, 11]

To define the similarity between a point and a cluster representation, the number of mismatches can be used as shown in 2.8 [7, 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 [11, 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 [7]. 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 [2, 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[2], BIRCH[17] and, ROCK (categorical) [11] 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 [2, 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 [2, 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* [7], *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 [6].

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[7, 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* [7].

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 [7, 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 [2, 8, 14, 16]. DBSCAN [8] 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 [8].

2.4 Feature-Weighted Clustering

K-means assumes that all attributes/features are of the same importance [1]. 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 [1]. 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 [4]. 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. β 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 $P(U^{\hat{t+1}}, C^{\hat{t+1}}, W^t)$
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 $P(U^{\hat{t+1}}, C^{\hat{t+1}}, W^t)$ (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^β is replaced with w_{lj}^β — 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 ϵ 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 [6]. Below is the cost function of *FSC*, where β 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[5]. 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 shannon entropy to create the cost function shown in 2.30. Here, the shannon entropy cab be rergarded as 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. γ 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 (log-transformed entropy weighting *K*-means), has modified EWKM [27]. Two problems of EWKM are adressed: EWKM is very sensitive to γ, 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.

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

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

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

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

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

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

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]. The types are sometimes referred as cluster validation indices [32]. 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 data points 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. The measure is common to use when tuning the parameters of traditional partitioning algorithms. There is limited information on how to implement and use the measurement or any other internal index for subspace clustering methods. Unlike, *k-means* soft-subspace clustering have asymmetric distances between two points of different clusters.

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

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

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

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

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

2.5.2 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.

β 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, β is a weight between P and R, a $\beta > 1$ emphasizes the R.

Table 2.1: Properties of algorithms discussed in this chapter [10, 18, 21, 25, 27]

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]
	LEKM	$\approx O(tnkm)$**	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
** Exact complexity is not given, but mentioned to be slower than EWKM due to how cluster centers are updated.

2.6 Summary and Method Justification

Figure 2.1 summarizes the properties of all the mentioned algorithms in this chapter.

The findings of the pre-study suggests that feature-weighted clustering can be used to attack the challenging problem of clustering high-dimensional data by reducing the weight of irrelevant features. Algorithms such as *Weighted K-means*, *FSC*, *Entropy Weighted K-means* and *LEKM* have all been introduced. The mentioned algorithms manage high-dimensional data clustering in different ways. The last three, 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 a time complexity (linear) similar to traditional partitioning algorithms.

Due to the previous mentioned benefits of SSC algorithms, three types of SSC methods will be tested for this thesis. They are *EWKM*, *FSC* and *LEKM*. *EWKM* is chosen as it takes into account the information gain in the cost function. FSC is chosen for the purpose of having an algorithm with different properties i.e. being a *Fuzzy* SSC algorithm that does not take into account information gain. *LEKM* is added for its use of logarithmic distances that could possibly help resolve problems with noisy data. To evaluate the performance, the algorithms will be compared to the traditional *K-means* algorithm.

The evaluation will consist of a supervised external index due to the difficulty of implementing a correct internal validation for *SSC* algorithms. Data was to be tested against a ground truth label through the measurement of Purity and a panel of judges.

The chosen methods, are possible to implement for mixed data, but are not originally implemented for the purpose. 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.

Chapter 3

Method

The method chapter subjects the reader to the framework used to qualitatively compare the different algorithms on the given dataset.

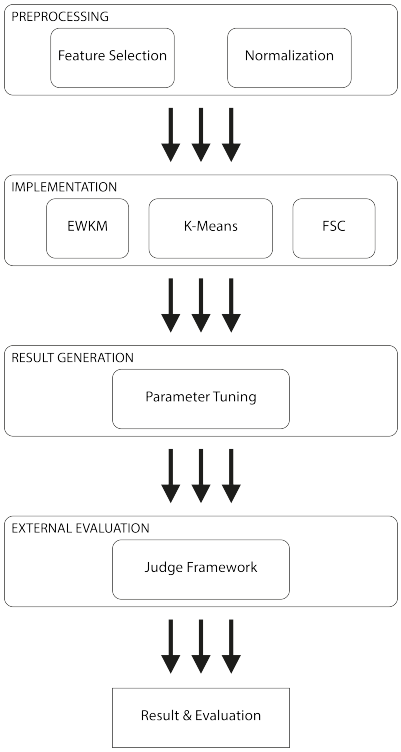


Figure 3.1: Method Design

3.1 Method Design Overview

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

3.2 Dataset Properties

The given dataset is a real-world, high-dimensional and, mixed dataset. It is a set of *song objects* extracted from an internal music database. Each song is a vector of features — where different features describe the song through different song properties. The vector consists of two types of features, general metadata and audio-based features.

The general metadata includes attributes that predominantly identify the song without resulting to audio analysis. Features of the type includes *Album* (string), *Artist* (string), *Title* (string) and *Year of Origin* (ordinal). The type also includes *BPM* (numerical, zero to around 200) and *Energy-feel* (ordinal, one to ten). These features include both numerical and categorical features — features stored as strings can be converted to categorical data.

The core features are the Audio-based features. They represent the song with features based on audio properties of a song. There are 160 of these features, together they can be seen as 160-dimensional vector where each dimension represents a feature.

The source of the features are a trained supervised neural network. The input of the network are audio embeddings of songs — devised from an audio spectrogram — that have a predetermined genre. The genre is utilized as a label to train the network for the task of classifying the genre of an audio embedding. In total there are 33 different genres a song can be labeled as.

Audio features of our dataset are a representative sample of neuron weights in the trained supervised neural network. The genre label is also a feature that exist in a smaller subset of our given dataset.

3.3 Pre-processing

The amount of dataset objects and features were reduced in the preprocessing stage for various reasons.

For the ease of testing — lowering computation time, and validating — being able to use the *genre* feature as a validation label, the original dataset was sampled and reduced to a size of $5 * 10^4$.

With a focus on tackling the problem of high-dimensionality, the choice was to use the numerical methods *EWKM*, *LEKM*, *FSC*, and, *K-means*. That left a restriction to only select numerical features as input. Given that the audio vector was used to categorize songs in the neural network, the hypothesis was that clustering analysis could also be done exclusively on that feature-space. The dataset was therefore preprocessed to only include the subspace of audio-features.

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

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

3.4 Implementation

3.4.1 EWKM

An implementation of the EWKM algorithm is available in *R and CSRAN* through the *wskm* package [34]. The code is written in C and wrapped for R. All of the source code is obtainable from Github[1].

There are modifications made in the implementation that differ from the original research paper [18]. Additional normalization steps have been added when calculating w_{lj}. Equation (3.4) shows how w_{lj} is updated in the given implementation. Occurrences of empty clusters during partitioning are managed by re-sampling cluster centers i.e. restarting the algorithm; in the original algorithm empty clusters were not treated. Finally, a convergence delta can be

[1] https://github.com/SimonYansenZhao/wskm/

set by the user which decides the percentage of change needed between two iterations for convergence.

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 [35] — 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*.

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

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

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

3.4.2 FSC

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 eq. (2.24), and eq. (2.26). γ is replaced by β, and the small value of ϵ was set to 0.001 as proposed in [6].

3.4.3 LEKM

LEKM like *FSC* was created by the author, it too was based on the source code of EWKM [34]. The same steps are necessary for this algorithm as EWKM, both are based on negative entropy, the difference was how the costs, weights etc. were updated. How they were updated corresponds to the equations of LEKM shown in eqs. (2.35) to (2.39). A slight modification is that the right-hand side i.e. the negative entropy is dropped from when updating partitions.

The modification is based on a discussion with one of the authors of the original *LEKM* paper. The gist of the discussion was that the entropy should not be used for updating clusters, and could result in negative distances.

3.4.4 K-means

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*. As a popular clustering algorithm, *K-means* is widely available for Python. The PyClustering implementation of *K-means* was used for this report [36]. The implementation was created in *C++* and wrapped for Python.

All algorithm specific parameters are ran with all k candidates. The best performing combination of parameters are deemed the best candidates for the algorithm.

3.5 Parameter Selection

Results generated by an algorithm, on the sample dataset, for different parameter candidates, were compared in order to select the best parameters. The selection of best parameter is based on an external validation, a *purity* score.

3.5.1 Convergence Rate

For generating results, convergence was set to occur if the cost delta between previous iteration and current was less than 0.5%.

3.5.2 Purity

An external validation index was created using the *genre* feature as a ground truth in combination with the measurement of *purity*. To create the purity score, the most dominant genre of each cluster was aggregated and divided by the total amount of songs in the dataset.

3.5.3 Amount of Clusters

The problem of deciding the amount of clusters i.e. k is global as all the clustering algorithms chosen are partition based. This section describes how k was chosen for all of the given algorithms.

For the choice of k, it was decided to not use a competitive learning strategy. The strategy would add another layer of complexity on the algorithms and a possible point of error. k was instead based on the *purity* results of the whole sample.

Candidate values of k were restricted to 50, 100, and, 500 in order to reduce the time needed to generate results. The values were chosen to give an approximation on how large k should be. 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. 500 was decided as the maximum amount for k to keep the average cluster size to be above 100.

3.5.4 EWKM

EWKM introduces the γ parameter. The recommended range of $\gamma \approx (0.5, 3)$ mentioned [18, 34] was extended with smaller values $(0.0005, 0.001, 0.005, 0.01, 0.05, 0.1)$ as candidate values. The smaller values were added to avoid possible problems with the objective function being negative due to a large negative entropy.

Results of gamma values with immediate convergence were discarded as the behaviour was deemed unwanted, and not what was expected from the algorithms.

3.5.5 FSC

The unique parameters of *FSC*, β — The fuzzyness variable, and ϵ — A small constant, were set to 2.1 resp. 0.00001 in [6]. In our tests β was varied between 1.5 and 30 while ϵ was kept to the value mentioned in the report.

3.5.6 LEKM

In [27] values of β was varied between 1 and 16 with a decreasing score after a value of 2.0. Our candidate values ranged between 0.5 and 10. Similarly to EWKM, results of immediate convergence were disregarded.

γ is a hyper-parameter of EWKM and LEKM. In short, γ determines the likeliness to cluster on more or less features. For any given dataset the ideal γ can

vary. An ideal γ is found (in this scenario) when there is no other γ value that can result in a better purity.

3.5.7 K-means

K-means does not have any hyper-parameter that needs to be tweaked. The only thing that was necessary for the result generation was to run K-means on all values of k.

3.5.8 Ranking songs and clusters

Songs were sorted in ascending order based on their distance to their respective cluster center. The distance is equivalent to the distance used for assigning a partition to an object. For *k-means* that was the regular euclidean distance. For SSC algorithms the distance included the feature weights of the cluster. The distances differ from SSC algorithm to another as e.g. *LEKM* uses log-transformed distances.

Clusters were similarly sorted in ascending order. The distance for a cluster was set as the average distance of songs with a membership in the cluster.

The sorting was used as a ranking, based on the idea that ”better” clusters would have smaller distances than ”worse” clusters. The reason for not using a genre accuracy for the ranking was to allow clusters that have novel themes that are not genre-specific a chance to be high ranked.

The ranking allowed for a cutoff in songs and clusters i.e. the top five clusters could be chosen. To not have the same genre type of clusters again and again in the top 5, each genre could only appear as dominant in one cluster (except for pop and rock, as they were regarded as larger genres with more diversity). As such, after sorting the top five cluster were picked from small to large distance with the condition stated above.

3.6 Expert Evaluation

To answer how performance varies between traditional and SSC algorithms in terms of novelty and cohesion, a test was created. In this test professional composers were used as evaluation judges.

The platform of the evaluation was a webpage created by the author. In here, different clusters could be browsed. Each cluster was represented by 10 30-second song previews. The chosen songs were sampled based on a half-normal distribution, i.e. songs with a relative small distance had a higher chance to be picked than those with a high distance. A picture of the webpage is shown in fig. 3.2. Django [2] and Bootstrap 4[3] were used to build the page.

The top five ranked clusters of each algorithm was chosen for the test. In addition to the ten clusters sourced from the algorithms, five more clusters were added that were sourced from existing playlists on the stakeholder platform.

The test was conducted as blind test to remove any bias playlist clusters might have. The composers did not know that the clusters had different sources. In their minds they were all from the same algorithm machine learning algorithm.

Initially, the idea was that composers would rate a cluster on *similarity* and *novelty* (from a scale of 1-10). After showing a mock test and discussing it with the head of product experience, head of machine learning, and a number of composers at the stakeholder, it was conducted that the parameters had to be modified. There were two reasons: the terms were unclear for the composers, and, a general quality was hard to transcribe from the scores.

To adhere to the problems, *similarity* was divided into *audio similarity*, and *cultural similarity*. *novelty* was changed to a term familiar with the composers — *playlist uniqueness*, i.e. how unique the cluster would be as a playlist compared to already existing playlists on the platform. Additionally, a *general quality* field was added to solve the second problem of scores not translating to a general quality. A text box was also added where the composers could add any description they thought summarized the cluster.

The composers and author were situated in the same room. All composers evaluated the clusters together — The composers had mentioned that they believed they can give a more accurate score together if they were allowed to discuss the clusters in as a team. Discussions the composers had during the evaluation was recorded. The author also answered questions the composers on how to navigate the webpage.

[2] https://www.djangoproject.com/
[3] https://getbootstrap.com/

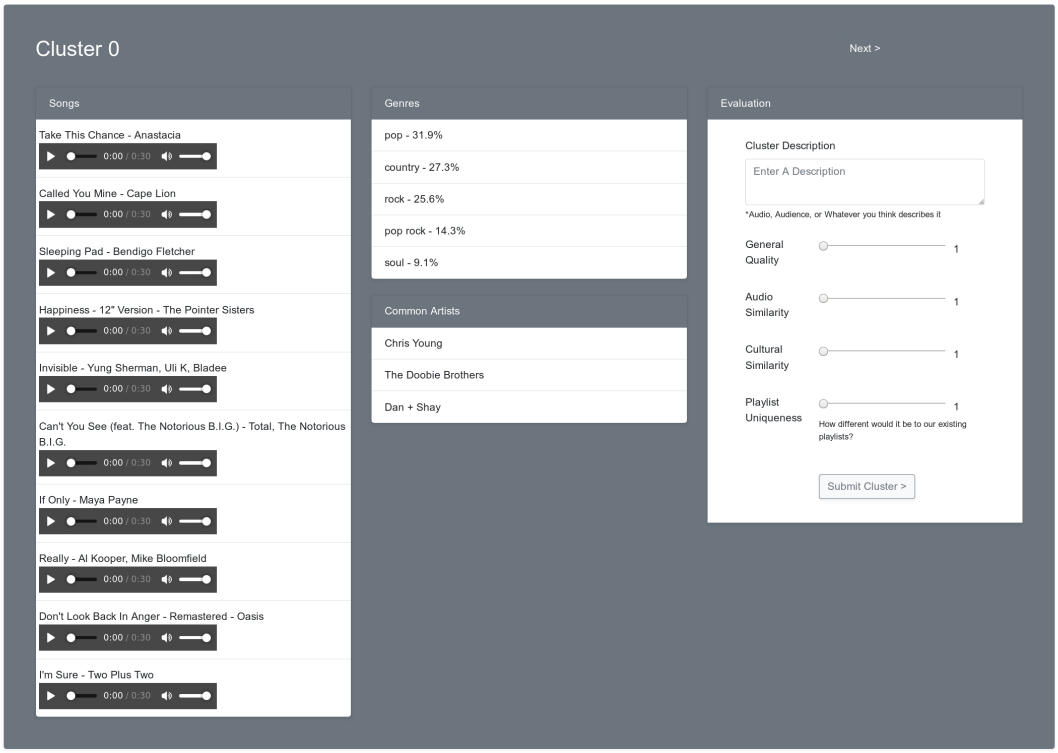


Figure 3.2: Screenshot of the evaluation website.
From the left-hand-side song samples are shown and played, In the middle statistics about genres and common artists. On the right-hand-side the evaluation is situated with a general description text area, and sliders for the rest of the evaluation categories

Chapter 4

Results

The results of this chapter has been divided into two sections: *General Results* and *Evaluation Results*. *General Results* show how different algorithms perform in terms of *purity*, *convergence speed* and *feature weight distribution*. The section ends with a summary showing the best parameters, and, purity scores for each algorithm. *Evaluation Results* presents the expert evaluation scores of *K-means*, the best performing *SSC-algorithm*, and playlists through table 4.2.

4.1 General Results

Below sections describe the results of K-means, EWKM, LEKM, and FSC with different parameters on the dataset. The chapter is summarized by table 4.1, which shows the *purity* of the best performing parameters for each algorithm.

To avoid repetitiveness algorithm-specific figures are shown only for $k = 500$. The choice of $k = 500$ is justified as it is the best performing k for algorithms in regards to purity (see the section summary).

4.1.1 EWKM

Figure 4.1 presents the purity given different γ's for EWKM. From the figure we see a trend of the purity slowly decreasing until a value of $\gamma = 2$, where a

steep increase occurs to a score of 0.44.

Figure 4.2 show the corresponding amount of iterations until convergence. The amount of iterations needed to converge is decreasing with the value of γ. Here, we see the immediate convergence of values of 0.05 and higher.

Figure 4.3 show the amount of restarts needed to generate non-empty cluster partitions. Most value selections needed zero or one restarts. The outlier was the choice of 0.01, which required 10 restarts. Selections larger than 2.0 resulted in zero restarts.

Immediate convergence (with no restarts) — as shown in the figures to be 2.0 and higher, were caused by the Shannon entropy being more negative than the total dispersion within clusters, making the objective function negative.

Figure 4.4 represents the feature weights of each cluster as colored grids (often referred to as a meshgrid) for various choices of γ. The grids in fig. 4.4 show the weight of a specific feature for a cluster as a colored rectangle. Yellow denotes a high feature weight, while purple denotes a low value. A cluster's feature weight vector is a horizontal line in the grid. For all the tested values of γ that did not result in immediate convergence, the weight distribution of the clusters were similar. One feature was often dominant i.e. had a high weight (around 1) while the rest had feature weights near zero. The grids do however, show a trend of less dominant features being produced with larger values of γ. The results of dominant features are unlike what is presented in [18]

.

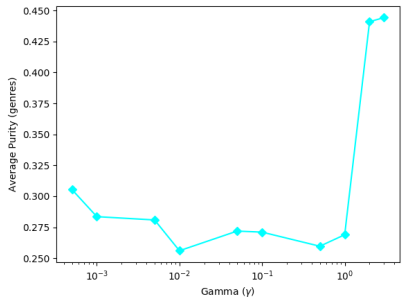


Figure 4.1: Purity accuracy of EWKM given various values of γ, $k = 500$

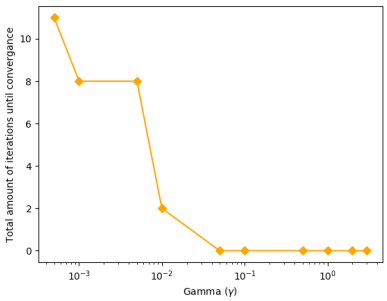


Figure 4.2: Iterations until convergence of EWKM given various values of γ, $k = 500$

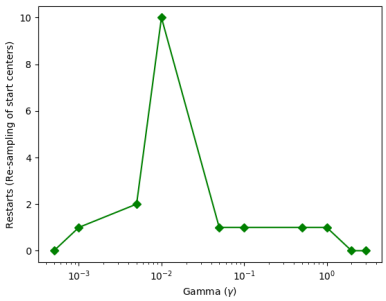
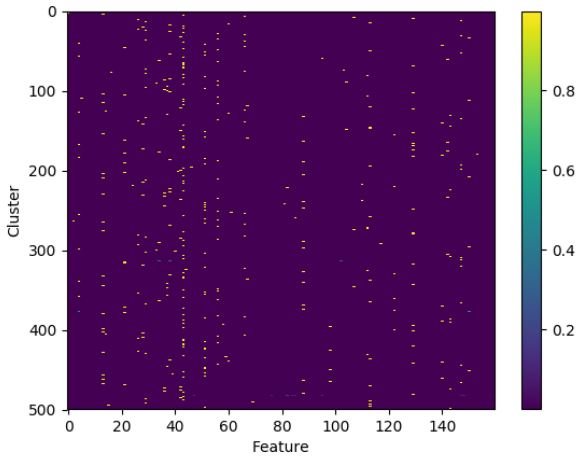
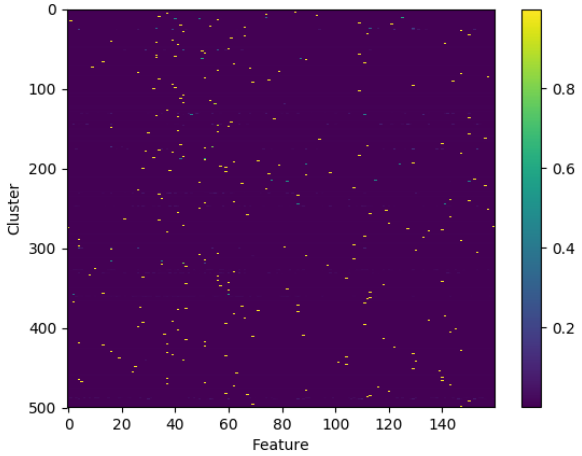


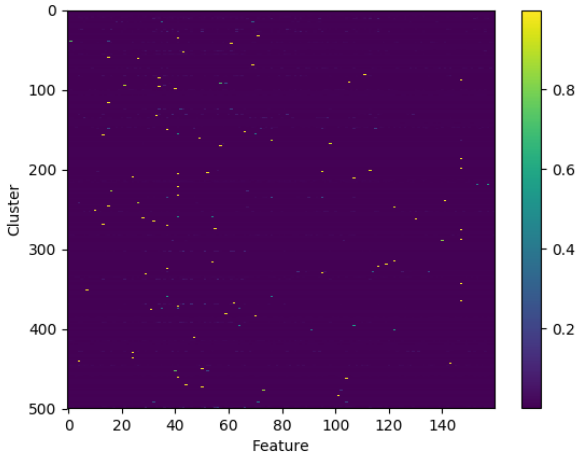
Figure 4.3: Restarts of EWKM given various values of γ, $k = 500$



(a) $\gamma = 0.0005$



(b) $\gamma = 0.05$



(c) $\gamma = 0.5$

Figure 4.4: Feature-Weight Meshgrid of EWKM

4.1.2 LEKM

The purity scores of LEKM is shown in fig. 4.5. With higher values the purity is increasing until a peak purity is reached at $\gamma = 1.4$ with a score of 45.5%. For values ≥ 2.6 the purity decreases steeply to values of 43.8%, these values correspond to values of immediate convergence.

The amount of iterations until convergence is shown in fig. 4.6. Higher values resulted in increasingly more iterations needed, with the peak reached at 20 iterations with $\gamma = 2.2$, until a drop to zero — immediate convergence — occurs at values of 2.6 and larger. Compared to EWKM, the γ's of LEKM is larger when immediate convergence occurs.

The feature weight grids of LEKM is shown in fig. 4.7 (Note that the values colors represent, are different for each grid). We see that there are multiple features representing the cluster through the figures. Certain features are dominant in a very few clusters (shown as a purple-dominant vertical line in the grid) while others are dominant in most clusters (shown as a yellow-dominant vertical line in the grid). The highest weights are also very small compared to EWKM, but are still, larger than the least important features of the cluster. The dispersion between small and large weight values decrease with a higher γ.

Another perspective of how the weights differ given various γ is shown in fig. 4.8. The plots of the figure show the values of all feature weights in all clusters, summarized into a histogram. The resulting histograms show a distribution with properties of a normal distribution, similar to what is mentioned in [18]. From smaller- to larger-values, we see the deviation of the distribution decrease, leading to a more and more uniform distribution.

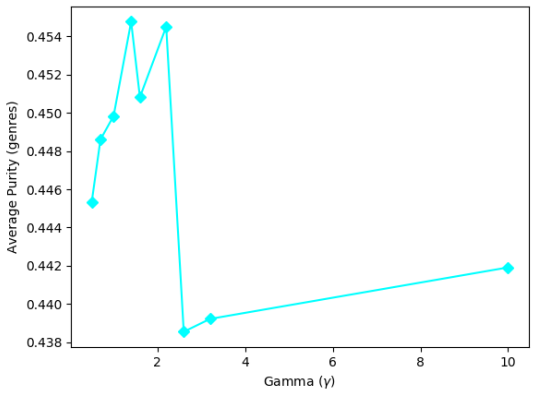


Figure 4.5: Purity accuracy of LEKM given various values of γ, $k = 500$

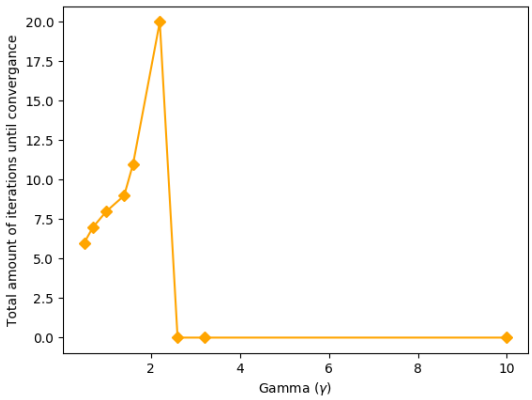
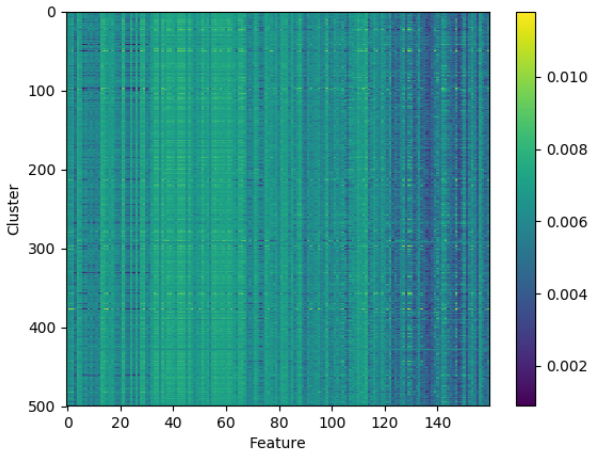
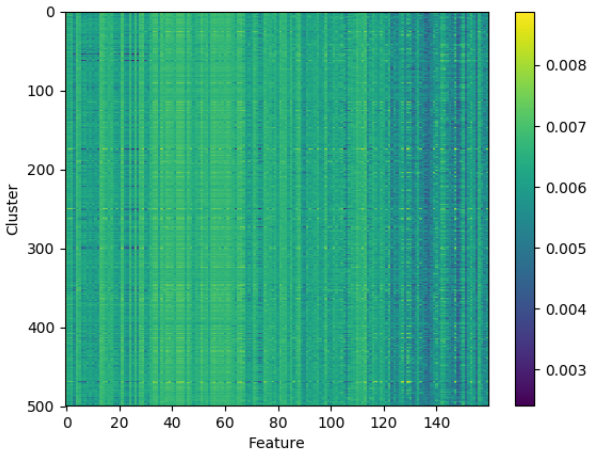


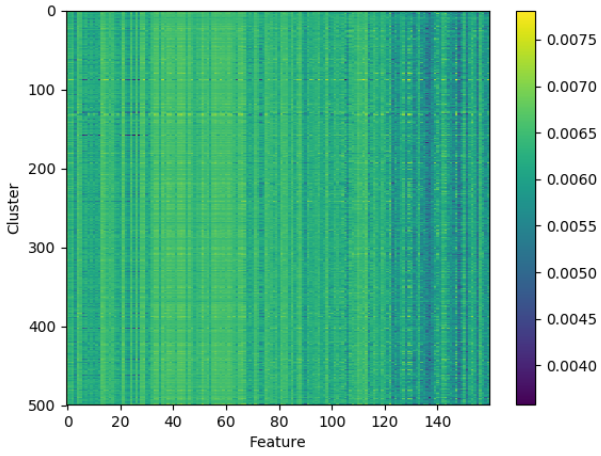
Figure 4.6: Iterations until convergence of LEKM given various values of γ, $k = 500$



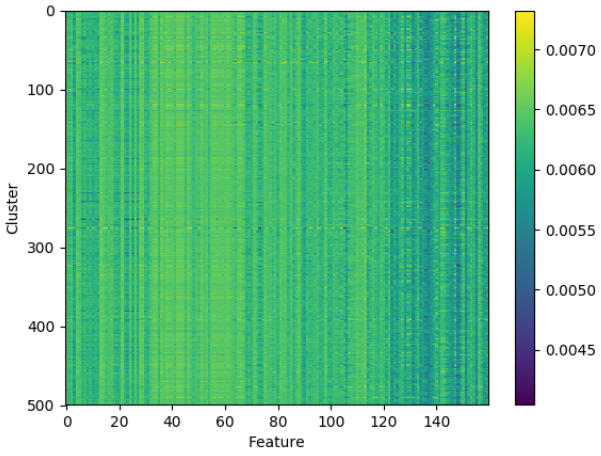
(a) $\gamma = 0.5$



(b) $\gamma = 1.0$

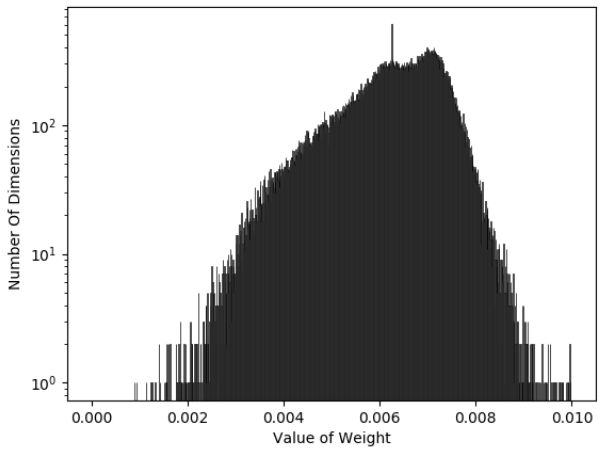


(c) $\gamma = 1.6$

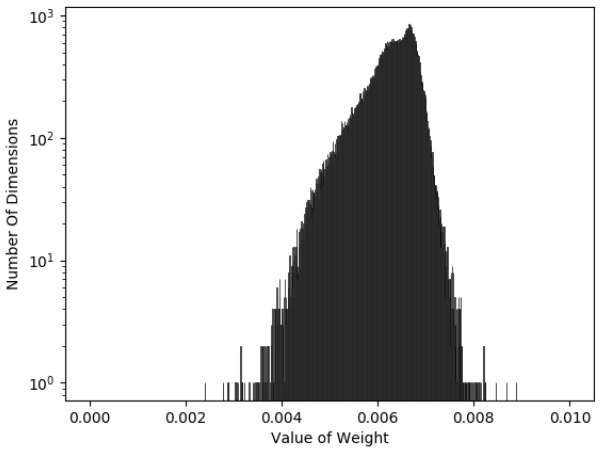


(d) $\gamma = 2.2$

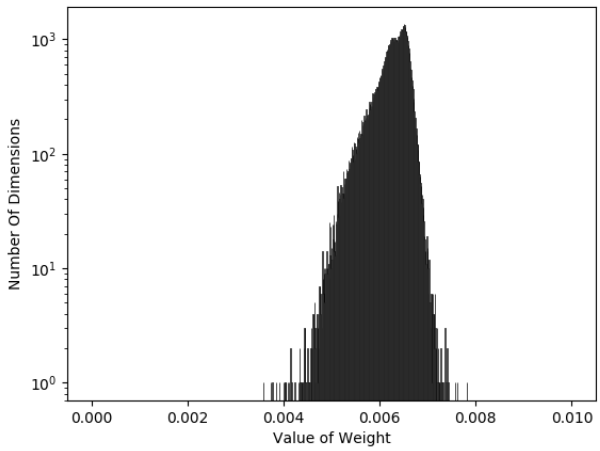
Figure 4.7: Feature-Weight Meshgrid of LEKM (k=500)



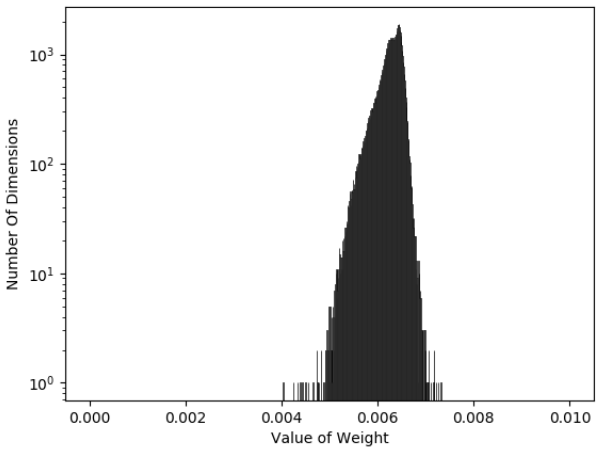
(a) $\gamma = 0.5$



(b) $\gamma = 1.0$



(c) $\gamma = 1.6$



(d) $\gamma = 2.2$

Figure 4.8: Distribution of feature weights values upon all clusters ($k = 500$)

4.1.3 FSC

Figure 4.9 shows how the algorithm performs based on purity on the dataset. From the figure we can see a tendency of $\lim_{\beta \to \infty}$ led to the best score. The score is slightly worse than *LEKM*

The amount of iterations, as shown in fig. 4.10, are high for smaller values of β and a constant of two for larger β.

Figure 4.11 show how feature weights are distributed along clusters. A selection of a lower β i.e $\beta = 1.5$ results with a single dominant feature weight for most clusters, similar to *EWKM*. As γ increased, more and more features were given importance in clusters. Some small amount of clusters had larger differences between feature weight values, thus, the pictures becoming more blue than yellow.

The feature weight distribution of different β's are shown through mesh-grids in fig. 4.11, and histograms in fig. 4.12. The figures show that a selection of a lower β i.e $\beta = 1.5$ results with a single dominant feature weight for most clusters.

A choice of $\beta = 1.5$ is shown to have feature weights let to a single dominant weight per feature. Higher values resulted in uniform more dominant features. Similar to LEKM, certain features appeared more as a dominant feature in clusters as well as certain feature weights being small for many clusters. Unilke LEKM, certain feature outlier existed, that had larger feature-weights. In terms of distribution as shown in fig. 4.12 from the resulting distribution is not normally distributed but like LEKM the distribution ended up being more sharp for larger values.

4.1.4 K-means

K-means converged after 7 iterations given $k = 500$. The purity score was 45.4% as shown in table 4.1.

4.1.5 Summary

The best parameters, together with their *purity* scores are shown for $k = 50, 100, 500$ in table 4.1 for each algorithm. The table shows that $k = 500$

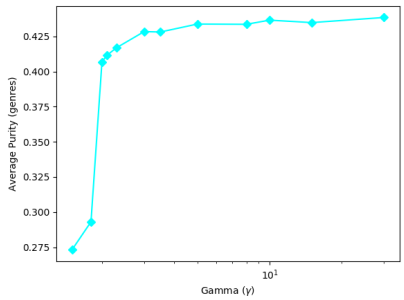


Figure 4.9: Purity accuracy of FSC given various values of γ $k = 500$

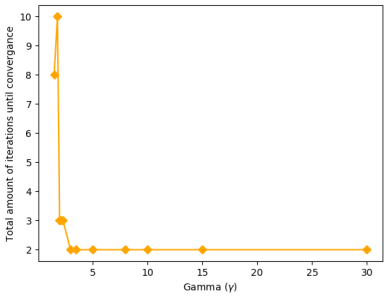


Figure 4.10: Iterations until convergence of FSC given various values of γ $k = 500$

achieves the best purity scores for all algorithms except EWKM (where $k = 100$ results in the greatest purity). For $k = 50$ and $k = 100$, *K-means* is the best performing algorithm, but only marginally. For $k = 500$ *LEKM* performs marginally better than *K-means* with the best overall score of (45.5%). In all values of k, EWKM is the worst performing algorithm.

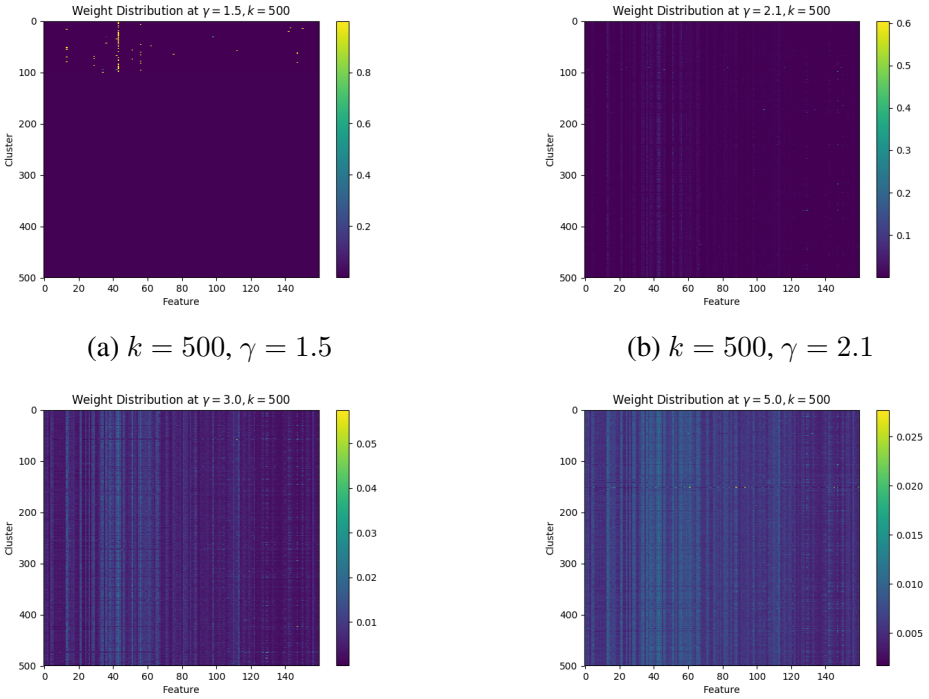
.

	K-means	EWKM		LEKM		FSC	
k	Purity	γ	Purity	γ	Purity	β	Purity
50	39.0%	0.005	28.2%	0.0	38.8%	30.0	38.0
100	40.8%	0.005	28.9%	2.1	40.6%	15	39.6
500	45.4%	0.001	28.6%	1.4	**45.5%**	30.0	43.8%

Table 4.1: Best parameter- and purity-score (γ, β) given k

4.2 External Evaluation

The results of the blind test can be found in table 4.2 and figure **??**. The table and figure show how the different sources was rated on average on the clusters of the source. The clusters sourced from playlists perform better than K-means and LEKM on all parametrs as expected. In general quality there is only as 0.4 respectivly 0.6 difference between playlist, K-means and LEKM. The algorithms perform better in terms of audio similarity than cultural similarity.



(a) $k = 500$, $\gamma = 1.5$

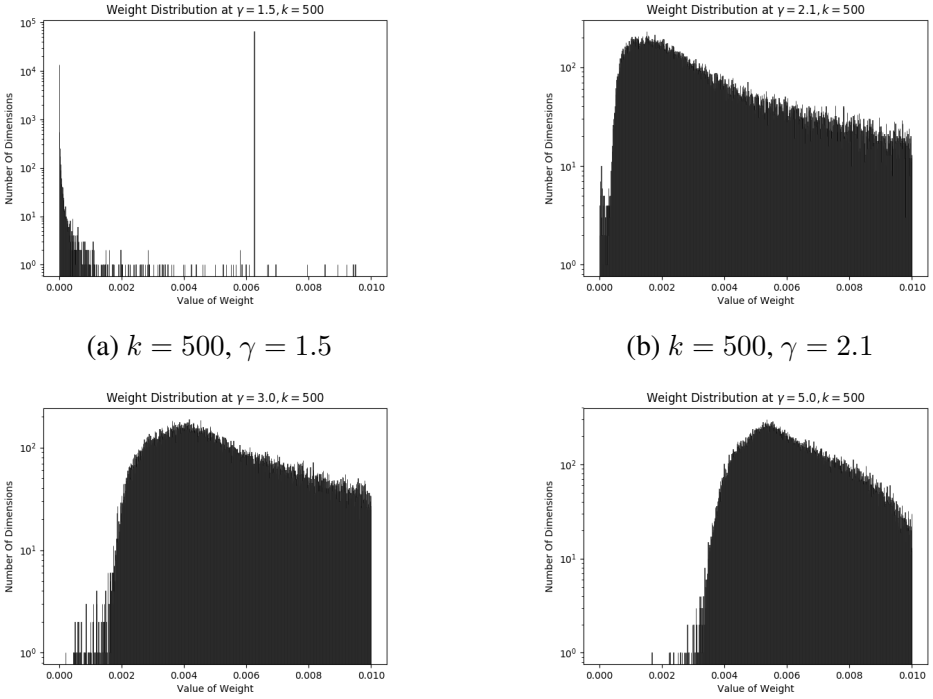
(b) $k = 500$, $\gamma = 2.1$

(c) $k = 500$, $\gamma = 3.0$

(d) $k = 500$, $\gamma = 5.0$

Figure 4.11: Feature-Weight Meshgrid of FSC

There is a great difference between the cultural similarity rating between the playlist clusters and the algorithmic clusters. Regarding K-means vs LEKM, K-means performs marginally better than LEKM in all categories, where the biggest difference is in novelty. Listening to the discussions during the blind test, clusters from the algorithms were mentioned to have a core of songs that were an obvious theme with an additional 2-3 songs that were culturally different. For well performing clusters that difference added depth to the playlist in a positive way. For bad performing clusters the mixins were offputting for the theme of the cluster, and made the cluster worse in terms of general quality.



(a) $k = 500$, $\gamma = 1.5$

(b) $k = 500$, $\gamma = 2.1$

(c) $k = 500$, $\gamma = 3.0$

(d) $k = 500$, $\gamma = 5.0$

Figure 4.12: Distribution of feature weights values upon all clusters

	Evaluation			
Source	General Quality	Audio Similarity	Cultural Similarity	Playlist Uniqueness
Playlist	7	7.6	8	5.6
K-means	6.6	7	6.2	5.4
LEKM	6.4	6.4	6	3.8

Table 4.2: Average rating of clusters given source type

Chapter 5

Discussion

5.1 Quantitative results

That K-means would marginally beat out all soft-subspace clustering algorithms was unexpected and surprising. There could be various reasons for the occurrence, they all start with the dataset.

The properties of the dataset could have made it less suitable for SSC algorithms. One possibility is that the dataset had in a sense already had its features selected beforehand, the 160 neuron weights selected as features were already picked from a larger quantity of neurons based in the neural network. SSC algorithms could theoretically still boost the performance of clusters after the feature sampling, by finding the unique subspaces of each cluster, but they did not. There was no significant improvement of purity performance with LEKM, and a downgrade in performance occurred with the usage of EWKM.

The fairly popular soft subspace clustering algorithm of EWKM performed poorly on the dataset. One reason was due to the single feature dominance. However, another problem that arose was the restriction of values γ that could be used in order to avoid immediate convergence. This problem was not mentioned in the paper introducing the algorithm [18], in the paper the weights were supposed to be of a normal distribution, with some features more and less important per cluster.

FSC had the accuracy scores increase with a larger β. A larger β did mean more uniform feature weights. The result would for that reason, indicate that important features are disregarded as irrelevant when smaller values of β is

used. The algorithm failed in that sense, to find the important features, and could not reduce the dimensionality of the dataset. Where *FSC* did not need to reduce the dimensions the score improved.

LEKM, which created feature weights that were normally distributed on each cluster, performed similarly to K-means. It had one advantage to K-means. The generated feature weights of each cluster had given less features importance when calculating the distance between data-object and cluster center. That indicates that clusters can be dimensionally reduced to a smaller subspace without impacting the accuracy negatively unlike *FSC*. This shows that LEKM can still be used for the problems high-dimensionality creates in cluster analysis and could potentially cluster even larger datasets.

Finding the correct parameter for the SSC algorithms ended up being non-trivial with an external evaluation of purity based on the genre label. *Genres* is not the ideal candidate for an external validity measure, and so tuning the parameters could have led to the right parameter for the SSC-algorithms being missed. The problem this thesis had with tuning the parameters of the algorithm show the problem of tuning soft subspace clustering algorithms on a new dataset. Internal indices such as silhouette have to be manipulated with the cluster specific asymmetric distance measure, while an external criterion used here, leave things left to be desired.

That the highest value of k would have the highest purity value was not unexpected as smaller clusters tend to allow for higher purity accuracy. The F_1 score could have been more suitable for the task.

5.2 Qualitative results

The Evaluation scores of the blind-test indicates that the clusters generated by the algorithms could be used to create playlists. This is reflected by the general quality scores of the algorithms. The audio similarity was through discussions and metrics fairly high along all clusters. While, as mentioned some songs do not fit the general theme of the cluster due to cultural differences such occurrences could be tuned out by a composer generating a playlist through the use of filters.

It can therefore be stated that the audio features selected in the preprocessing were indeed enough to cluster to songs, and the algorithms were indeed capable of clustering the dataset. However, to improve the cultural similarity

score one should consider adding features such as origin year, and, country of origin.

Generated clusters from the algorithms performed much worse than the playlist counterpart for the *Playlist uniqueness* index. All three sources saw a drop-off compared to the *General Quality* category. Notable was that playlists were rated over a value of five, when they were in fact, already available on the platform. The ratings given do not reflect the discussion the team of playlist composers had as they indicated that some clusters had a good blend and hard to replicate without hearing the cluster. Based on this, it could be stated that there was a misunderstanding of what the index meant. Based on the discussions alone, the belief is that the novelty criteria had been met.

K-means performed slightly better than LEKM in every category. The results indicate that K-means perform at least as good as LEKM, which for the most part corresponds to the results of purity. For statistical confidence that K-means performs better on the dataset than LEKM, a larger cluster sample size would have been needed.

5.3 Future Work

Adding additional features to the feature-space of the given dataset could have seen the results of the algorithms improve. The cultural similarity — where the algorithms struggled compared to the playlists — could have been helped by metadata such as origin year. To use more of the metadata e.g. *artist* (represents songs with the same artist), categorical features would have been needed to be used. There are two main ways to tackle the problem of mixed data with soft-subspace clustering. Modify the given algorithm's distance measurement to adhere to mixed data or use a mixed data clustering algorithm. The former is not a trivial task, and so the latter could be a better choice. The WOCIL algorithm mentioned in [21] would then be a suitable choice the for mixed data clustering on the dataset.

All the given results are based on the $50k$ sample of the larger dataset. Scaling the clustering process to the larger 50-million set should impact *k-means* negatively. Feature reduction could at that point become more essential for the clustering process, and so, SSC algorithms could become more useful.

Moving to a larger dataset requires more processing. Using a single thread on a single core is not efficient. An idea would instead to scale the algo-

rithms across multiple computer nodes using a distributed framework. The distributed implementation of the algorithms would require the code to be rewritten to fit the capabilities of the framework.

Algorithms based on k-means are non-deterministic due to the sampling of initial centers. Finding and using suitable sampling techniques as mentioned in [27] specifically made for subspace-clustering could improve the results of the soft-subspace clustering methods.

Many of the internal validation indices that exist for clustering analysis takes account both intra-cluster distance and inter-cluster distance — An ideal clustering should have clusters far between. The chosen SSC algorithms were all based on intra-cluster distance. No account was taken on the distance between clusters. A natural step to improve performance would be to find an algorithm which takes both conditions to account.

Chapter 6

Conclusions

The given high-dimensional musical dataset — sampled to a size of $50k$, and processed to only hold numerical features — had LEKM as the best performing *SSC*-algorithm. The comparisons done between *K-means* and LEKM show that there is no significant performance difference between the algorithm, feature-weighting did not improve the performance. Similar *Purity* scores were found for both algorithms. Based on a panel of judges, any of the two algorithms could be used to produce clusters of high general quality, audio similarity and, novelty. The results indicate that the generated clusters could be used as a tool when composing new playlists based on the dataset.

Bibliography

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

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

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

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

[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.

[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.

[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.

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

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

[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.

[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.

[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.

[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.

[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.

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

[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.

[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.

[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.

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

[35] *NumPy C-API — NumPy Manual*.

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