DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2019

Soft-Subspace Clustering on a High-Dimensional Musical Dataset

EMIL JUZOVITSKI

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

Soft-Subspace Clustering on a High-Dimensional Musical Dataset

EMIL JUZOVITSKI

Master in Machine Learning
Date: October 11, 2019
Supervisor: Johan Gustavsson
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Host company: Soundtrack Your Brand AB
Swedish title: Soft-Subspace Clustering Applicerad på Högdimensionell Musikdata

Abstract

Clustering Analysis can be used to solve various tasks. In this thesis, we look at the possibility of using clustering techniques to help generate novel music playlists by clustering a high dimensional dataset of songs. We compare how a newer category of clustering methods called *Soft-subspace* clustering (SSC), which weighs features independently for each cluster, performs compared to the traditional *k-means* algorithm. The SSC algorithms of *EWKM* (Entropy Weighted k-means), *FSC* (Fuzzy Subspace-Clustering), and *LEKM* (Log- Transformed Entropy Weighting k-means) were tested on a $5 \cdot 10^4$ sample of the dataset. Parameters were tuned based on an external validation index. The best performing SSC algorithm, which ended up being LEKM, was then compared to the results of k-means through a committee of judges with professional music composing experience. The results show that both LEKM and k-means are capable to cluster the dataset and generate novel clusters. Both algorithms create clusters of high general quality, but there is no shown benefit of using LEKM over k-means on the given dataset. For a more conclusive result, a larger sample dataset would be needed.

Sammanfattning

Klusteranalys kan användas för att lösa varierade problem. I denna avhandling granskar vi specifikt möjligheten att använda klusteranalys för att skapa spellistor (playlists) med nya musikaliska teman. Detta görs genom att klustra ett högdimensionellt dataset bestående av låtar. Vi jämför hur en ny sorts klustringsmetod, *Soft-subspace Clusering* (SSC), som viktar attributen separat för respektive kluster, presterar jämfört den mer traditionella *k-means*-algoritmen. Tre olika SSC-algoritmer testades på en datamängd av $5 \cdot 10^4$ låtar: *EWKM* (Entropy Weighted k-means), *FSC* (Fuzzy Subspace-Clustering), och *LEKM* (Log- Transformed Entropy Weighting k-means). Dessa algortimers parametrar justerades systematiskt utifrån ett externt valideringsindex varefter den bäst presterande SSC-algoritmens förmåga att klustra sånger bedömdes av sakkunniga experter med kompositionserfarenhet och jämfördes mot k-means. Resultaten visar att både LEKM och k-means klustrar datamängden likvärdigt, och lyckas generera kluster med helt nya musikteman. Båda algoritmerna skapar kluster av allmänt hög kvalitet, men det går inte att fastslå att LEKM är bättre än k-means på den givna datamängden.

Contents

Chapter 1

Introduction

Retail industries use music in stores as a branding mechanism [1]. Substantial effort and employee hours (money) are needed for the creation (and navigation) of musical playlists that fit a company brand. To provide a cheaper and faster alternative, pre-curated playlists composed to work in various retail industries have been made available by a business-targeted music streaming platform. The work of composing playlists is time-intensive. A large amount of time is spent on the search of candidate songs, which currently requires expert knowledge — professional playlist composers. In order to lessen the burden on composers, we look at the option of using *Clustering Analysis* to automate the process of candidate song generation.

Clustering Analysis — the exploratory art of finding *clusters* (groups) in a dataset [2] — have been used to solve problems such as: Determining the life quality of cancer survivors based on cancer treatment and demographics [3], and classifying businesses by transaction tendencies [4]. Clustering analysis can similarly be used to algorithmically (without human interaction) find musical themes (clusters) within a musical dataset. Themes generated can then be used as candidate songs for the creation of playlists.

Various types of clustering methods exist, they vary in how the similarity between points are measured, and how clusters are defined. The properties of the dataset to be clustered is what decides the suitability of a method. The data-types (categorical, numerical or mixed), the size (the number of data objects), and the feature-dimensionality (the number of attributes describing an object) of a dataset are all critical factors when choosing a method. Choosing the right method can help find the relevant patterns in data that are otherwise

undetectable.

Clustering high-dimensional data suffers from the *Curse of dimensionality* [5, 6, 7], i.e. any two points with the same cluster membership are bound to have features in which there are large distances between the points [8]. As such, it is hard to assess which points are actually similar as the distance is dominated by dissimilar features which are often irrelevant.

A traditional choice for clustering is *k-means*. k-means is a numerical partition-based algorithm that iteratively partitions data points into clusters [9]. The algorithm does not manage feature weighting. Instead, features have to be weighed by the user beforehand. In the case of high-dimensional data, it becomes time-consuming and often impossible to manually determine feature weights. This forces the option of weighing features uniformly, which leads to the curse of dimensionality.

To avoid clustering on a high-dimensional dataset feature reduction techniques have been used. Reduction techniques try to find the smallest set of features that can represent a dataset. The drawback of using such techniques is that they lead to a loss of information [10], and disregard the fact that different clusters can depend on different features.

Soft-subspace clustering (SSC) [4, 10, 11] is a newer clustering method type designed specifically to tackle the problems of high-dimensionality. SSC algorithms are unique for the usage of cluster-specific feature-weights — determined automatically during the process of clustering. This enables the reduction in the dimensionality of feature-weights to a subset of features deemed relevant by the algorithm, independently for all clusters.

1.1 Problem Statement

In this thesis, we study how soft-subspace clustering algorithms compare to traditional approaches — represented through the k-means algorithm — on the given musical dataset. In order to answer how SSC algorithms compare to k-means, the following problem statement has been created:

How does the performance of soft-subspace clustering algorithms compare to traditional methods on the given high-dimensional dataset, from the perspective of novelty and general quality, when validated by a committee of judges consisting of playlist composers?

1.2 Objectives and Scope

The objective is to determine how k-means (a traditional algorithm) and *SSC*-algorithms compare in regard to the generation of thematic musical clusters. More specifically, clusters of the different sources are compared on the musical *novelty*, and the musical quality generated. In addition to the comparison between algorithms, it is also of interest to compare algorithmic cluster sources with existing playlists based on the same criteria.

Only three SSC-algorithms will be tested. They are *EWKM* (Entropy Weighted k-means), *LEKM* (Log- Transformed Entropy Weighting k-means), and *FSC* (Fuzzy Subspace-Clustering) — which are all introduced in the background. They will be parameter tuned based on genre purity. Each candidate parameter will only be tested once. The best performing tuned algorithm in terms of genre purity will then be compared to k-means.

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 consists of features — where different features describe the song through different song properties. There are two main features, general metadata — such as *Duration* and *Artists* — and audio-based features — describing songs through an extraction of audio features of the song.

This thesis focuses on the problem of clustering the audio-based features, which are made out of numerical features. The audio feature-set includes 160 features that make the data high-dimensional. Other properties such as *mixed data* are mentioned, but not addressed. For this thesis, algorithms are compared on a $5 \cdot 10^4$ sized sample of the original $5 \cdot 10^7$ sized dataset. Algorithms chosen were numerical in nature and restricted from using the categorical features of the dataset.

The dataset is given as is, the only feature manipulation done to the dataset are preprocessing techniques common for cluster analysis such as sampling a subset of the dataset and normalizing features. The process of picking and sampling audio features before the preprocessing is not within the scope of the project.

Chapter 2

Background

The fundamentals of clustering are introduced in this chapter. Distance measurements of varying data-types are shown. Hierarchical, partition and density-based clustering are all described. The chapter details how the popular k-means algorithm functions and continues by mentioning different feature-weighted clustering algorithms including soft-subspace clustering (SSC) that are based on k-means. The chapter ends with a method justification for the research design.

2.1 Objective of Clustering

Given 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 (features) – and the input parameter k (a positive integer), the objective of partition clustering is to divide objects into k disjoint clusters $C = \{C_1, C_2, ..., C_k\}$, where similar data points are partitioned into the same cluster [12].

The objective varies between clustering types, the above objective is as stated just for (hard-membership) partition-based clustering (see section 2.3.2), but the general idea of partitioning similar points to the same cluster is shared by all clustering methods. Another type of clustering methods is e.g. *density*-based clustering where the objective is to separate local dense areas (with high similarity) from noise (see section 2.3.3 for more details) [13].

2.2 Similarity and Distance Measures

The similarity between points has to be quantified numerically in order to be used in an algorithm. The relative closeness between two points X_i and X_o is quantified numerically through a *similarity measurement* $s(X_i, X_o)$ [14]. A high value indicates that the points are close, while a low measure indicates that the points are dissimilar. Values of $s(X_i, X_o)$ go between 0 and 1. Another solution is to quantify the dissimilarity of two points instead. The dissimilarity is described through a *distance measurement* $d(X_i, X_o)$ — where $d(X_i, X_o) \geq 0$.

Distance measures are often used with quantitative data, i.e. numerical data. Similarity measures are frequently used when data is qualitative, i.e. categorical and mixed data [15]. Both measurements are symmetric, i.e. $d(X_i, X_o) = d(X_o, X_i)$. An exception is subspace clustering methods, see section 2.4 for more details.

It is not always the similarity between points that are measured. A point X_i can also be compared to a cluster representation C_l — frequently used in partitioning algorithms. A cluster representation is a synthetic point that summarizes the characteristics of a cluster [2, 16, 17]. The representation of a cluster varies between algorithms. In k-means for example, the mean of each feature is used as the cluster representation.

Distances (and similarities) can be measured in many ways and are often explicitly created for a dataset with attributes of one specific data type. The sections below introduce some popular distance and similarity approaches for different data types, starting with the most common numerical data, continuing with categorical data, and ending with approaches that combine the two in *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 eq. (2.1). A frequent choice for distance measurement in clustering analysis is the squared Euclidean [5, 9, 18], shown in eq. (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 (eq. 2.3) where $q = 2$. The Euclidean and other Minkowski distances often involve a preprocessing step of normalizing the dataset features [5]. Features with higher variance will otherwise have a higher contribution in deciding the clusters. Similar to many other measurements, the *curse of dimensionality* diminishes the ability to differentiate distances between points in high-dimensional data [6].

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

The mentioned distances can be converted to a similarity measurement — with values between 0 and 1 — by using the Gaussian kernel as shown in eq. (2.4) [19].

$$s(X_i, C_l) = \frac{\exp(-0.5 \cdot d(X_i, C_l))}{\sum_{t=1}^{k}\exp(-0.5 \cdot d(X_i, C_t))} \tag{2.4}$$

2.2.2 Categorical Similarity Measures

Categorical data is not commonly referred to as a single datatype. It is instead a group of datatypes consisting of nominal and ordinal data. Unique properties separate 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 [2].

Most clustering algorithms that cluster categorical data do not take into account the differences between the data types mentioned above. There are distance measures that do (see *daisy function* in [2]) 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 to as categorical data, and treated as nominal data.

In its purest form a categorical similarity is simply a statement of *yes* or *no*, as shown in eq. (2.5), i.e. answering if two attribute values are the same or not [2, 16].

$$\delta(x_{ij}, x_{oj}) = \begin{cases} 1 & \text{if } x_{ij} \neq x_{oj} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

To define the similarity between a point and a cluster representation, the number of mismatches can be used as shown in eq. (2.6) [12, 17]. A cluster representation in this case, 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 to as a *mode* [17] (see section 2.3.2.1).

$$d(X_i, C_l) = \sum_{j=1}^{m} \delta(x_{ij}, C_{lj}) \tag{2.6}$$

Another way to define the similarity is to compare the values of X_i and the values of all data points $\forall X_o \in C_l$ for each attribute as shown in eqs. (2.8) and (2.9) [16, 19]. 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. The weight w_j is an attribute weight defining how important an attribute is. The weights of each attribute is based on the Shannon entropy (for more on entropy see section 2.4).

$$s(x_{ij}, x_{oj}) = 1 - \delta(x_{ij}, x_{oj}) \tag{2.7}$$

$$s(x_{ij}, C_{lj}) = \frac{\sum_{X_o \in C_l} s(x_{ij}, x_{oj})}{\sum_{X_o \in C_l} [x_{oj} \neq null]} \tag{2.8}$$

$$s(X_i, C_l) = \sum_{j=1}^{m} w_j s(x_{ij}, C_{lj}) \tag{2.9}$$

2.2.3 Mixed Similarity Measures

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

Huang proposes the similarity measure shown in eq. (2.10) [12], w_l determines how important categorical data (denoted with ^{c}) is compared to its numerical counterpart (denoted with ^{r}) for cluster l. In a popular partition-based algorithm *K-Protoypes*, a simplification is made, local weights w_l for each cluster l is replaced with a single global weight w. The weight w is a user-defined input.

$$d(X_i, C_l) = \sum_{j=1}^{m_r} (x_{ij}^r - c_{lj}^r)^2 + w_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, c_{lj}^c) \tag{2.10}$$

Cheung and Jia [19] propose representing numerical data as an vector X_i^r — where numerical data is denoted by r — while letting each categorical attribute (denoted by ^{c}) have its own weight. The amount of categorical attributes is denoted by m_c, and the total amount of attributes is denoted by $m_f = m_c + 1$ as all numerical values share one weight. The similarity measure is seen in eq. (2.11). Categorical weights are automatically weighed through information gain, as such the algorithm is a *feature weighting* algorithm. See section 2.4 for more details.

$$s(X_i, C_l) = \frac{1}{m_f} s(X_i^r, C_l^r) + \frac{m_c}{m_f} \sum_{j=1}^{m_c} w_j s(x_{ij}^c, c_{lj}^c) \tag{2.11}$$

2.3 Clustering Algorithms

Clustering algorithms are often divided into categories. Hierarchical, partition and density-based clustering are the largest categories [20]. Each of the given types are introduced in the sections below.

2.3.1 Hierarchical Clustering

A hierarchical algorithm generates a set of nested clusters, also known as a *dendogram* [5, 20]. There are two approaches for 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 [5], BIRCH [21] and ROCK (categorical) [16] are popular algorithms of hierarchical clustering. The creation of a dendogram in these clustering algorithms results in a high time complexity ($O(n^2 \cdot log(n))$) that make the algorithms less suitable for larger datasets however, by running the algorithms on a sample representation of the dataset, as used in CURE and ROCK bigger datasets can be managed [5, 20]. In addition to creating a dendogram, the type includes methods which are able to partition clusters with an arbitrary shape.

2.3.2 Partition Clustering

In partition clustering a partition of clusters U is found by iteratively optimizing a cost function [5, 9, 20]. 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* (see section 2.3.2.1).

The algorithms converge either when no data points switch cluster association or on a user-defined convergence delta for the cost function. The run time 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 cluster 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 (also known as objective function) of which k-means is optimized on is shown

in eqs. (2.12) and (2.13), n refers to the amount of objects, m to the amount of attributes (features) and k is the amount of clusters. The cost function is either minimized (eq. 2.12) or maximized (eq. 2.13) [9].

$$P(U, C) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} d(x_{ij}, c_{lj}) \tag{2.12}$$

$$P(U, C) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} s(x_{ij}, c_{lj}) \tag{2.13}$$

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, $u_{il} = 0$ *or* 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 to as *Fuzzy-memberships* and often includes a fuzziness index, a variable deciding the importance of the generated weights [11].

There are many algorithms based on k-means, they can be referred to as the k-means family of clustering [18]. Different similarity functions determine what k-means family algorithm the optimization represents [9]. Euclidean distance would result in k-means. The categorical similarity shown in eq. (2.6) results in *K-modes* [17]. If the similarity is the mixed type shown in eq. (2.10) the optimization represents *K-Prototypes* [12]. Additional algorithms in the class (*Weighted-k-means*, *EWKM*, *FSC* etc.) [4, 9, 10] tweak the distance function (measurement) to include some sort of weighting (see section 2.4 for more).

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. The sub-optimization problems (**P1**, **P2**) become:

P1. Assign 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 eq. (2.14).

$$u_{il} = \begin{cases} 1 & d(X_i, C_l) \leq d(X_i, C_t) \quad \text{for} \quad 1 \leq t \leq k \\ 0 & \text{for} \quad t \neq l \end{cases} \tag{2.14}$$

For **P2.** we update C — the center representations — through the equations below:

- For numeric values solved by obtaining the average:

$$c_{lj} = \frac{\sum_{i=1}^{n} u_{il} x_{ij}}{\sum_{i=1}^{n} u_{il}} \tag{2.15}$$

- For categorical values, one option is defining the center as mode [17], which is solved by:

$$c_{lj} = a_j \tag{2.16}$$

  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[12, 18]. The solution is simply to use eq. (2.15) for numerical attributes and eq. (2.16) 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 steps 2. and 3.

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 [22]. Compared to eq. (2.12) 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* run time. *K-medioids* is usually implemented through the *PAM* algorithm (an efficient implementation of *K-medioids*) which has a time complexity of $O(tk(n–k)^2)$, whereas the basic k-means (Lloyd's implementation) has an complexity of $O(tnk)$ [22].

Extensions of *PAM*, such as *CLARA* and *CLARANS* are approaches to improve the scalability and run time of PAM by clustering on a sample representation of the dataset [22]. The sample is defined to be of size $40 + 2k$ by the authors, where 40 is a constant, and k is the wanted amount of clusters. To reduce sampling bias, CLARA is clustered on multiple sample representations. The sample resulting in the smallest cost is used to define the resulting medoids of the algorithm 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 neighbours dynamically 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 parameters chosen as inputs for the algorithm. When using hierarchical and partition-based clustering the value of *k* has to be decided.

A way to determine *k* is by running the algorithm with different values of *k* and then through an internal or external criterion measure (see section 2.5.1) decide the best k for the dataset [12, 23]. Running an algorithm multiple times makes the process of clustering many times slower, and should be avoided for larger datasets.

Sampling is used in CLARA and CLARANS [22] to improve the performance of PAM. The idea in these algorithms is to find medoids for the whole dataset

by clustering on a sample representation of the dataset. The same reasoning can be used to find an approximation of *k* for the whole dataset. The algorithm still has to be run on the sample multiple times and evaluated against a defined 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 [19, 24]. 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 assigned to a cluster, that cluster's weight is increased together with its neighbours' weights, while the rest of the clusters' weights decrease. Eventually, some clusters disappear. A benefit of competitive learning is that the amount of clusters are determined during the clustering process, removing the need to cluster the dataset multiple times. Note, however that a user input of maximum number of clusters 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 ones can result in convergence on a suboptimal result [24, 25], 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.

For numerical data, *k-means++* [25] proposes replacing uniform random sampling with the steps seen below. It is also possible to add an initialization step for categorical and mixed data as mentioned in [24].

1. Pick one center C_l uniformly at random.
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 chosen point with the minimum distance to X_i,X_t.
3. Repeat 1. and 2. until K points have been picked.

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 [5, 9, 13, 20]. DBSCAN [13] is a well known clustering algorithm of the category, where each data point in a cluster must have a user defined *MinPts* in its $\mathcal{E}$-neighborhood, where $d(X_i, X_o) < \mathcal{E}$ 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 [13].

2.4 Feature-Weighted Clustering

k-means assumes that all attributes/features are of the same importance [2]. 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, a weighting step can be added, where attributes are given a weight based on its perceived importance in relation to other attributes. The step is done to avoid giving high importance to features that are irrelevant — as irrelevant attributes damage the clustering performance [2] — and increase the bias towards relevant features. Assigning a large weight to an irrelevant feature is in fact, worse than not using the attribute at all.

Deciding on how to weigh 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 [4]. The high degree of dimensions make it almost impossible to manually determine the weights of the attributes.

A fairly popular way to handle high-dimensional data is through the use of dimensionality reduction techniques e.g. *PCA* [26, 27]. 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 [7]. An assumption that often is false in high-dimensional datasets.

Another possibility which is introduced in the next section, is to add steps to the clustering algorithm to automate the process of weighting features.

2.4.1 Automated Feature-Weighting

Feature-weighting can be done automatically. Automatic feature-weighting is commonly referred to as *feature weighting* and is often achieved by extending a k-means like algorithm. An additional *feature weight* (attribute weight) variable is added to the cost function given in eq. (2.12). The feature weights are updated based on the distance contribution of the feature. How the cost function changes from eq. (2.12) depends on what concepts we use to automate the weighting.

One algorithm of this class is *weighted-k-means*, which extends k-means by adding a weight to each attribute [9]. w-k-means uses the cost function in eq. (2.17). The cost function introduces two new variables: W — a weight vector containing the weights for each attribute — and β — a hyper-parameter defining the importance of the weight vector. **P3.** is the new subproblem below of updating W after C and U have been updated. The weight W^t is updated through eq. (2.18), and the feature distance D_j is defined in eq. (2.19).

P3. Update the weights of all attributes.

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

$$P(U, C, W) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} w_j^{\beta} d(x_{ij}, c_{lj}) \tag{2.17}$$

$$\hat{w}_j = \begin{cases} 0 & D_j = 0 \\ \frac{1}{\sum_{t=1}^{m} [\frac{D_j}{D_t}]^{\frac{1}{\beta-1}}} & D_j \neq 0 \end{cases} \tag{2.18}$$

$$D_j = \sum_{l}^{k} \sum_{1}^{n} \hat{u}_{il} d(x_{ij}, c_{lj}) \tag{2.19}$$

The algorithm can be modified to allow clustering on mixed data by using a distance measurement for mixed data [24].

For w-k-means and other feature weighting algorithms, k-means is extended with the additional step of updating feature weights. The steps of feature weighted algorithms are thus:

1. Choose initial cluster representatives C^0

2. Fix $C^t = \hat{C}$, $W^t = \hat{W}$, optimize $P(U^t, \hat{C}, \hat{W})$. Obtain $P(U^{t+1}, \hat{C}, \hat{W})$
   if $P(U^t, \hat{C}, \hat{W}) = P(U^{t+1}, \hat{C}, \hat{W})$
   return $P(U^t, \hat{C}, \hat{W})$ (Convergence)
3. Fix $U^{t+1} = \hat{U}$, $W^t = \hat{W}$, optimize $P(\hat{U}, C^t, \hat{W})$. Obtain $P(\hat{U}, C^{t+1}, \hat{W})$
   if $P(\hat{U}, C^t, \hat{W}) = P(\hat{U}, C^{t+1}, \hat{W})$
   return $P(\hat{U}, C^t, \hat{W})$ (Convergence)
4. Fix $\hat{U} = U^{t+1}$, $\hat{C} = C^{t+1}$, optimize $P(\hat{U^t}, \hat{C^t}, W^t)$. 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 $P(\hat{U}, \hat{C}, W^t)$ (Convergence)
5. Repeat steps 2., 3. and 4.

Instead of using a fuzzy weighting, i.e. having a input variable deciding the importance of the weight vector (β), another option is to decide the intra-cluster distance through information gain, in other words: Shannon entropy. Entropy can be said to be the amount of *disorder* in a system. Entropy is used in [4, 19]. Using the entropy works for both numerical and categorical data. In [19] the importance of a categorical attribute is defined as the average entropy of each value of the attribute. This is shown in eq. (2.20), 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.

$$w_j^{c*} = -\frac{1}{m_r}\sum_{t=1}^{m_r} p(a_{tj})log(p(a_{tj})) \tag{2.20}$$

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 eq. (2.21),

$$w_j = \frac{w_j^{c*}}{\sum_{t=1}^{d_c} w_t^{c*}} \tag{2.21}$$

The previously mentioned methods make the assumption that the same reduced features are of interest for all clusters. This is not inherently true, clusters usually have their own unique subspaces. In an example where clusters represent medical patient groups and patients (the objects of the dataset) are

represented by patient information (age, gender, traveling history etc.), the patient information (features) characteristics of each group is different, i.e. there are differences to why a patient is hospitalized between patient groups. One patient group could represent Malaria patients, and in that case *traveling history* is important, i.e. it does not vary much within the cluster. However, in a patient group which represents e.g. Parkinsons' decease, the traveling history is less important as it varies throughout the cluster, instead what is important here is the age of the patient, as shown by the smaller variance for the feature. The above example shows that subspaces are often different between clusters.

Subspace-clustering allows clusters to have their own feature subspace [4, 24, 28, 29], hence the name. It is a cluster analysis-specific way to deal with high-dimensionality. Irrelevant features can be discarded per cluster resulting in a reduction of dimensions while still losing less information than other techniques. There are two types of subspace-clustering techniques. The first type is *hard-subspace clustering* (HSC). The second type is *soft-subspace clustering*.

Hard-subspace clustering algorithms find the exact cluster subspaces of a dataset [4, 24, 28, 29]. CLIQUE [4], a bottom-up approach of HSC works by first splitting each dimension into equal sized parts and storing only the dense areas. Then intersections of dense areas of two dimensions are found. The process of finding dense intersections are repeated until dense areas are found for all feature dimensions of the dataset. The resulting dense areas are then picked as the clusters of the dataset, who have the property of having their own subspaces. This process of finding clusters is equivalent to what is used in the *Apriori* algorithm for the frequent itemset problem (a well known problem in data mining), and is linear in terms of objects in the dataset, but quadratic in terms of *k* thus, slow when a large quantity of clusters are to be found [20, 29].

Soft-subspace clustering algorithms find the approximate subspaces of clusters in a dataset. Each cluster is given a feature weight vector, where the vector determines the association probability (importance) of each feature for that given cluster [4, 10, 24]. The type is seen as an extension of feature weighting algorithms such as w-k-means with the same iteration steps, except that features have different weights in each cluster. Soft-subspace clustering is in general faster than its HSC counterpart, with an often linear time complexity. The paragraphs below introduce some notable SSC algorithms.

Automated-weighting algorithm (AWA) [30] is a soft-subspace approach similar to w-k-means, the difference is that for the cost function w_j^β is replaced

with w_{lj}^{β} — a weight for an attribute in a cluster C_l, where $\beta \geq 1$. The algorithm does not work when the variance of an attribute in a cluster is zero as the learning rules denominator becomes zero [31]. *FWKM* (Feature weighted k-means) [31] and *FSC* (Fuzzy-subspace clustering) [11] 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 a priori. 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 (β) is used to decide the importance of the weight [11]. Below is the cost function eq. (2.22), and the equation for updating the feature weights eq. (2.23) of FSC, where D_{lj} is the total distance of feature j in cluster l.

$$P(U, C, W) = \sum_{l=1}^{k} \left[\sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} w_{lj}^{\beta} d(x_{ij}, c_{lj}) + \epsilon \sum_{j=1}^{m} w_{lj}^{\beta} \right] \tag{2.22}$$

$$w_{lj} = \frac{1}{\sum_{t=1}^{m} \left[\frac{D_{lj}+\epsilon}{D_{lt}+\epsilon} \right]^{\frac{1}{\beta-1}}} \tag{2.23}$$

$$D_{lj} = \sum_{X_i \in C_l} d(x_{ij}, c_{lj}) \tag{2.24}$$

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 [8]. The entropy of a weight can then be used to describe the certainty of a feature in the cluster. In *entropy weighted k-means* (EWKM) [4] the intra-cluster distance is combined with the Shannon entropy to create the cost function shown in eq. (2.25), where $\gamma(\geq 0)$ is a user-defined variable that controls the size of the weights. Here, the Shannon entropy can be regarded 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 with w-k-means, **P3.** becomes optimizing W for the function while fixing U and C. For EWKM the feature weight optimization occurs through eq. (2.28). The smaller D_{lj} (eq. 2.29), the more important attribute A_j is to cluster C_l. A modified version of *EWKM* is *IEWKM* [32]. It specifies a cost function for both numerical and categorical data.

$$P(U,C,W) = \sum_{l=1}^{k}\left[\sum_{i=1}^{n}\sum_{j=1}^{m} u_{il}w_{lj}d(x_{ij},c_{lj}) + \gamma\sum_{j=1}^{m} w_{lj}log(w_{lj})\right] \quad (2.25)$$

$$c_{lj} = \frac{\sum_{X_i\in C_l} x_{ij}}{\text{Count}_{X_i\in C_l}} \quad (2.26)$$

$$u_{lj} = \min_{1\le l\le k}\left(\sum_{j=1}^{d} w_{lj}d(x_{ij},c_{lj})\right) \quad (2.27)$$

$$w_{lj} = \frac{\exp(\frac{-D_{lj}}{\gamma})}{\sum_{t=1}^{m}\exp(\frac{-D_{lt}}{\gamma})} \quad (2.28)$$

$$D_{lj} = \sum_{X_i\in C_l} d(x_{ij},c_{lj}) \quad (2.29)$$

A recent paper introducing the *LEKM* algorithm (log-transformed entropy weighting *K*-means), has modified EWKM [10]. Two problems of EWKM are addressed — the algorithm is very sensitive to γ and noisy data. To solve the problems, EWKM was modified to use log-transformed distances. The modification allows intra-cluster variances 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 noisy data are less impactful. LEKM is shown below, the cost function is found in eq. (2.30), where $\gamma \ge 0$.

$$P(U,C,W) = \sum_{l=1}^{k}\sum_{i=1}^{n} u_{il}\left[\sum_{j=1}^{m} w_{lj}\ln\left[1+d(x_{ij},c_{lj})\right] + \gamma\sum_{j=1}^{m} w_{lj}\ln(w_{lj})\right] \quad (2.30)$$

$$c_{lj} = \frac{\sum_{X_i\in C_l}\left[1+d(x_{ij},c_{lj})\right]^{-1}x_{ij}}{\sum_{X_i\in C_l}\left[1+d(x_{ij},c_{lj})\right]^{-1}} \quad (2.31)$$

$$u_{lj} = \min_{1\le l\le k}\left(\sum_{j=1}^{d} w_{lj}\ln\left[1+d(x_{ij},c_{lj})\right] + \gamma\sum_{j=1}^{d} w_{lj}\ln w_{lj}\right) \quad (2.32)$$

$$w_{lj} = \frac{\exp(\frac{-D_{lj}}{\gamma})}{\sum_{t=1}^{m}\exp(\frac{-D_{lt}}{\gamma})} \quad (2.33)$$

$$D_{lj} = \frac{\sum_{X_i\in C_l}\ln\left[1+d(x_{ij},c_{lj})\right]}{\text{Count}_{X_i\in C_l}} \quad (2.34)$$

2.5 Evaluation

Internal and external criteria, are the two main criteria categories in the evaluation of clustering performance [33]. They are sometimes referred as cluster validation indices [34].

2.5.1 Inner Criteria

Internal criteria is a validation type that can be described as evaluating the results without respect to external information [34]. An internal criterion tries to verify the objective of the clustering algorithm on the dataset. That is, making sure that points assigned to the same cluster are in general more similar than points outside of the cluster.

One way to evaluate the internal criteria is to use the *average silhouette coefficient*, and is shown in eq. (2.39) [35]. A single silhouette coefficient shown in eq. (2.38), looks at a data point's intra-cluster similarity (i.e. similarity to points within the same cluster), and inter-cluster similarity (similarity with points outside of the cluster).

$s_{co}(X_i)$ has a range between -1 and 1, where a high value (close to 1) indicates that a point is well clustered, i.e. the points of the same cluster are similar, and those who are in different clusters are dissimilar, far away from the point. 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 has asymmetric distances between two points of different clusters.

$$d_{avg}(X_i, C_l) = \frac{\sum_{X_o \in C_l} d(X_i, X_o)}{Count(X_o \in C_l)} \tag{2.35}$$

$$a(X_i) = d_{avg}(X_i, C_a) \text{ , } \textit{where } (X_i \in C_a) \tag{2.36}$$

$$b(X_i) = min_{C \neq C_a}(d_{avg}(X_i, C)) \tag{2.37}$$

$$s_{co}(X_i) = \frac{b(X_i) - a(X_i)}{max(a(X_i), b(X_i)} \tag{2.38}$$

$$\overline{s}_{co}(\mathcal{D}) = \frac{\sum_{i=1}^{N} s_{co}(X_i)}{N} \tag{2.39}$$

2.5.2 External Criteria

External criterion is another validation type that can be described as *validation of the results by imposing a pre-defined structure on the dataset*, i.e. data not used for generating the clustering results [34].

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 — often the reason to why an external validation approach was used 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 [33]. 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.

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 [33]. Equation (2.40) defines the measurement, $C = \{c_1, c_2, ..., c_k\}$ is the set of clusters, and $\Psi = \{\Psi_1, \Psi_2, ..., \Psi_t\}$ is the set of *truth*-classes. The measurement is easy to compute. A downside is that the equation does not penalize small clusters as such small clusters produce a high score.

$$P(C, \Psi) = \frac{1}{n} \sum_{l=1}^{k} \max_{1 \leq j \leq t} \left| C_l \cap \Psi_j \right| \tag{2.40}$$

F-measure is another measurement for the evaluation of the external criteria. The measurement is shown in eq. (2.43) [33]. 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 with this reasoning would mean that all similar points are within the same cluster, i.e. only *True Positives* (TP) and no *False Positives* (FP). In reality, some non-similar pairs are clustered together leading to *False Positives*. Similarly, *True Negatives* (TN) and *False Negatives* (FN) can occur.

The variables — TP, FP, TN, FN — are used to create the *precision* and *recall* measurements. The precision defines the ratio of *True Positives* on all positives — pairs in the same cluster. The precision is shown in eq. (2.41). 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 eq. (2.43) determines whether precision or recall should be emphasized. $\beta < 1$ results in precision being more important and $\beta > 1$ results in emphasis on the recall. The balanced F-measure is called the F_1*-measure* and is defined in eq. (2.44). It weighs the impact of precision and recall the same. The F-measure is common in information retrieval. It is however, more complex than purity and requires more effort to implement for clustering analysis.

$$P = \frac{TP}{TP + FP} \tag{2.41}$$

$$R = \frac{TP}{TP + FN} \tag{2.42}$$

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R)}{\beta^2 \cdot P + R} \tag{2.43}$$

$$F_{\beta=1} = \frac{2 \cdot P \cdot R}{P + R} \tag{2.44}$$

2.6 Summary and Method Justification

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

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*, *EWKM* and *LEKM* have all been introduced. They vary in how they manage high-dimensional data. 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 are 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 are 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. Results were to be tested against a ground truth label through the measurement of purity — due to the simplicity of the measurement — and a panel of judges — to evaluate the general quality and novelty — respectively.

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 is the main priority. Therefore, only numerical features of the dataset are used — See *Dataset* in section 3.1 for more details.

Table 2.1: Properties of algorithms discussed in this chapter [4, 10, 15, 20, 24, 28]

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	$O(n^2)^*$	Numerical	
	CLIQUE (HSC)	$O(n + k^2)^*$	Numerical	Yes[a]
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[b]
	OCIL	$O(tnkm)$	Mixed	Yes[b]
	W-K-Means	$O(tnk + tkm + tm)$	Numerical[c]	Yes[d]
	FSC	$O(tnk + tk + tkm)$	Numerical[c]	Yes[e]
	EWKM	$O(tnk + 2tkd)$	Numerical[c]	Yes[f]
	LEKM	$\approx O(tnkm)$**	Numerical[c]	Yes[f]
	WOCIL	$\approx O(tnkm)$	Mixed	Yes

[a] Exact Subspaces.
[b] Not all features have independent weights.
[c] Has been modified to allow mixed data.
[d] Only Feature weighting.
[e] Fuzzy-Weighting, Within-distance
[f] 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.

Chapter 3

Method

In order to generate results and compare the three different SSC algorithms with k-means, a method design was planned and executed. The method was divided into the step of *Preprocessing*, *Implementation*, *Parameter Selection* and *External Evaluation*.

3.1 Dataset

The given dataset is a real-world, high-dimensional mixed-dataset. It is a set of *song objects* extracted from an internal music database. There are in total $5{\cdot}10^7$ songs in the dataset. 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 independent of 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, and are derived from a supervised neural network. They are a sample of the neuron weights in a network trained to determine the genre of a song. Each song consists of an audio embedding — devised from an audio spectrogram — and a *genre* label — used to train the network.

Our dataset includes the genre feature. There are 33 possible genres for the songs in the dataset.

3.2 Preprocessing

The amount of dataset objects and features were reduced in the preprocessing stage for various reasons. For the ease of testing — reducing computation time — the original dataset of songs was sampled and reduced to a size of $5 \cdot 10^4$.

The thesis focuses on the problem of clustering high-dimensional data. Mixed-data SSC algorithms are not widely available online, and so the choice was to use the numerical clustering methods of EWKM, LEKM, FSC. This forced the exclusion of features that were not numerical. Given that the audio vector was used to categorize songs in the mentioned 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, 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). X_j is the vector of all datapoints for feature j, μ_j is the mean values for datapoints in feature j, σ_j is the standard deviation of feature j and Z_j is the normalized vector.

$$Z_j = \frac{X_j - \mu_j}{\sigma_j} \tag{3.1}$$

3.3 Implementation

3.3.1 EWKM

An implementation of the EWKM algorithm is available in *R* and *CSRAN* through the *wskm* package [36]. The code is written in C and wrapped for R. All of the source code is obtainable from an online repository[1]. The code has been extended by the author to work with *Python* and Numpy[37] — by re-wrapping the code.

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

There are modifications made in the implementation that differ from the original research paper [4]. Additional normalization steps have been added when calculating w_{lj}. Equation (3.4) shows how w_{lj}, the feature weight, 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, convergence is defined to occur if the value difference between cost functions of two iterations is within a user-defined percentage. If the difference is greater than the assigned percentage another iteration will occur.

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

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

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

3.3.2 FSC

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. (2.22), and eq. (2.23). γ is replaced by β, and the small value of ϵ was set to 0.001 as proposed in [11].

3.3.3 LEKM

LEKM like FSC was created by the author and based on the source code of EWKM [36]. The same steps are necessary for this algorithm as EWKM, since both are based on the negative entropy. The difference is the addition of logarithmic distances when updating the centers, partitions, and feature weights. How they were updated correspond to the equations of LEKM shown in eqs. (2.30) to (2.34).

A slight difference is that the right-hand side, i.e. the negative entropy was dropped 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.

3.3.4 k-means

EWKM is equivalent to k-means when $\lim_{\gamma \to 0}$ although simply setting $\gamma = 0$ forces division by zero. For this reason k-means needs its own implementation.

k-means is available for Python thorough various packages such as PyClustering [38]. However, to allow a fair comparison between algorithms, k-means was re-implemented by the author in *C* based on the EWKM source code.

3.4 Parameter Selection

Algorithms were tested with multiple candidate parameters on the same complete sample of $5 \cdot 10^4$ songs, in order to find the most suitable parameters. The impact of the different parameters was compared through an external validation index, a *purity* score that measured the genre purity of clusters. The best parameters in accordance to the index were deemed the most suitable, and chosen as the parameters for the algorithm for further tests.

Table 3.1 summarizes the hyperparameters of the different algorithms. Two parameters ϕ — the cost ratio between two iterations needed for convergence — and ϵ — the small constant to avoid division by zero for FSC — were not tuned. Convergence was set to occur if the cost ratio was less than 0.5% i.e. $\phi = 0.5$. Algorithm specific parameter selection is described in the sections below.

Hyperparameter	Description	Algorithm
k	Number of clusters	All
ϕ^*	Ratio for convergence (constant)	All
ϵ^*	Small constant to avoid division by zero	FSC
β	Weighing factor for fuzzy-weighting algorithms	FSC
γ	Entropy regularization factor	EWKM, LEKM

Table 3.1: Hyperparameters of the given algorithms (ϕ, ϵ were set as constants a priori)

3.4.1 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 as shown in eq. (2.40), the most dominant genre of each cluster — found through eq. (3.5), where $G = \{g_1, g_2, ..., g_p\}$ is the set of possible genres in the dataset, was aggregated and divided by the total amount of songs in the dataset.

$$\Psi_l = \underset{1 \leq o \leq p}{\text{argmax}} \sum_{X_i \in C_l} \left[X_i(genre) = g_o \right] \tag{3.5}$$

3.4.2 Amount of Clusters

The problem of deciding the amount of clusters (k) is central for all the chosen clustering algorithms. This section describes how k was chosen for all of the chosen algorithms.

The choice of k was chosen to be based on the purity results of the clustered dataset. A sampling approach was declined as the dataset was already a manageable size. Competitive learning was also declined as the strategy would add another layer of complexity on the algorithms and a possible point of error.

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 amount of clusters (50) was chosen to be larger than the 33 genres in the dataset. 500 was decided as the maximum amount for k to keep the average cluster size to be above 100.

3.4.3 EWKM

EWKM introduces the γ parameter. The recommended range of $\gamma \approx (0.5, 3)$ which was mentioned in Jing, Ng, and Huang [4] and Williams et al. [36], 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 undesirable, and not what was expected from the algorithms.

3.4.4 FSC

The unique parameters of *FSC*, β — the fuzziness variable, and ϵ — a small constant, were set to 2.1 and 0.00001, respectively, in [11]. In our tests β was

varied between 1.5 and 30 while ϵ was kept to the value mentioned in the report Gan, Wu, and Yang [11].

3.4.5 LEKM

In Gan and Chen [10] values of β were 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.

3.4.6 k-means

k-means does not have any algorithm specific 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 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 as shown in chapter 2. For k-means that was the regular Euclidean distance. For SSC algorithms the distance included the feature weights of the cluster.

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 i.e. the average distance to the cluster center for songs partitioned in the cluster. The sorting was used as a ranking, based on the idea that ”better” clusters would have smaller intra-cluster distances than ”worse” clusters. Genre accuracy was not used for the ranking in order to allow clusters that have novel themes — that are not genre-specific — a chance to be ranked high. From an application perspective it is more valuable to find non-genre specific themes, as genre specific themes are easily found through genre filtering. Another option would be inter-cluster distances, but deciding on a inter-cluster distance is non-trivial and algorithm dependent.

The ranking allowed for a cutoff in songs and clusters, i.e. the top five clusters could be chosen. To hinder the possibility of the same genre occurring in multiple clusters, 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 clusters were picked from small to large distance with the condition stated above.

3.6 Expert Evaluation

To answer how performance in terms of novelty and cohesion (similarity) varied between traditional and SSC algorithms, a blind test was created where professional composers were used as evaluation judges.

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 more familiar to 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* criteria was added to solve the second problem of scores not translating to a general quality. Evaluators were also allowed to add any additional description they thought summarized the cluster. The final criteria are shown in table 3.2.

Criteria	Description
Audio Similarity	How well do songs in the cluster share audio patterns?
Cultural Similarity	How well do songs in the cluster share an audience?
Playlist Uniqueness	How easy would it be for a composer come up with a similar playlist?
General Quality	How useful is the cluster for playlist creation?

Table 3.2: Description of criteria for external evaluation, each criteria was rated by the composers with an integer scale of 1 to 10

The platform of the evaluation was a webpage created by the author. In here, 15 different clusters could be browsed — 5 of each source (k-means, SSC-algorithm and existing playlists). 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 to the cluster center had a higher chance to be picked than those with a high distance. A picture of the webpage is shown in fig. 3.1. Django [2] and Bootstrap 4[3] were used to build the page.

The group of evaluators consisted of six composers. The chosen composers work professionally to create playlists that fit the needs of different retailers. They were deemed suitable as judges due to their experience in playlist creation. The composers would also be the beneficiaries of a clustering solution.

The test was conducted as a blind test to remove any bias composers might have. The composers were not told that clusters had different sources. Instead they were told that all clusters were from the same machine learning algorithm. To avoid possible suspicion, the presentation order of clusters were randomized. Information shown of clusters were also consistent between sources.

At the time of the evaluation, the created website was displayed onto a TV in a conference room. All 6 composers were situated in the room together with the conductor of the blind test. The test started by composers filling out their name, after which a random cluster was displayed on the screen. When a cluster was displayed, composers started their evaluation by listening to all the 10 songs shown, with varying durations — some songs did not require 30 seconds to understand how the song affected the cluster in regards to the chosen criteria. The next step involved a pre-criteria grading discussion with all composers, this discussion was used to conclude the groups general thoughts of the cluster. Finally, the cluster was discussed on each specific criteria and then graded from 1 to 10 through a slider. After the submission of a graded cluster, a new cluster was displayed. Throughout the test, discussions composers had were recorded by the conductor. Additionally, questions composers had on webpage navigation and general criteria considerations were answered by the conductor. In total the time for the evaluation took 2 hours and involved 5 clusters of each cluster source.

To evaluate the scores given by the composers on the three different cluster sources, a null hypothesis and an alternate hypothesis were created as shown in eq. (3.6) and eq. (3.7). A single-factor ANOVA test ($\alpha = 0.05$) was used to test the null hypothesis. Each criteria were subjected to the test in order to understand if there was any significant difference between the means of the given sources.

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

$$H_0 = \left[\mu_{pl} = \mu_{SSC} = \mu_{k\text{-}means} \right] \tag{3.6}$$

$$H_a = \left[\mu_{pl} \neq \mu_{SSC} \neq \mu_{k\text{-}means} \right] \tag{3.7}$$

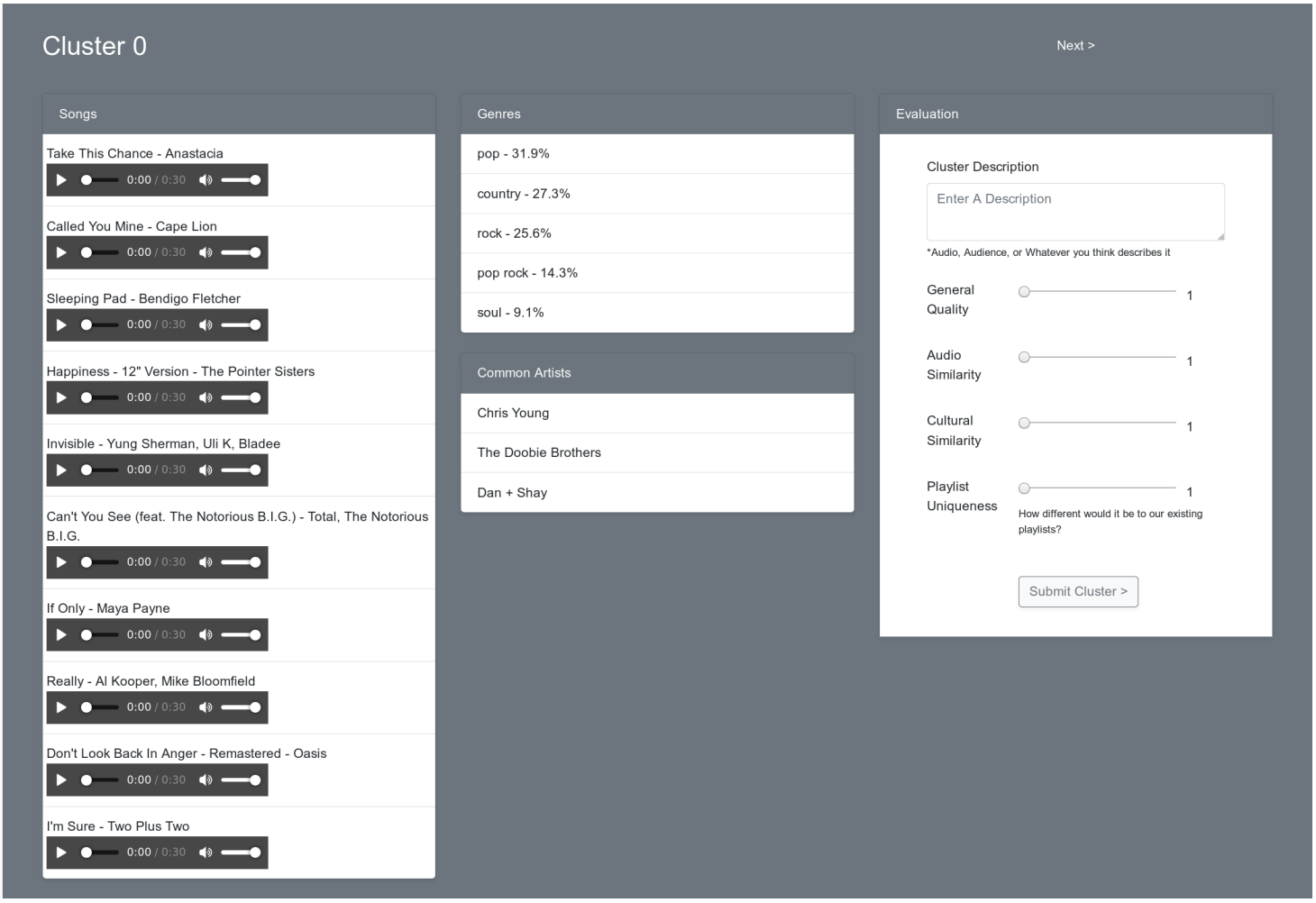


Figure 3.1: Screenshot of the evaluation website.
From the left-hand-side song samples are shown and played. In the middle, statistics about genres and common artists are shown. On the right-hand-side, the evaluation form is situated with a general description text area, and sliders for the rest of the evaluation categories.

Chapter 4

Results

The results of this chapter have been divided into two sections: *General Results* and *Evaluation Results*. General Results shows how different algorithms performed 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 section 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 section 4.1.5).

4.1.1 EWKM

Figure 4.1a presents the purity given different values of γ 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.

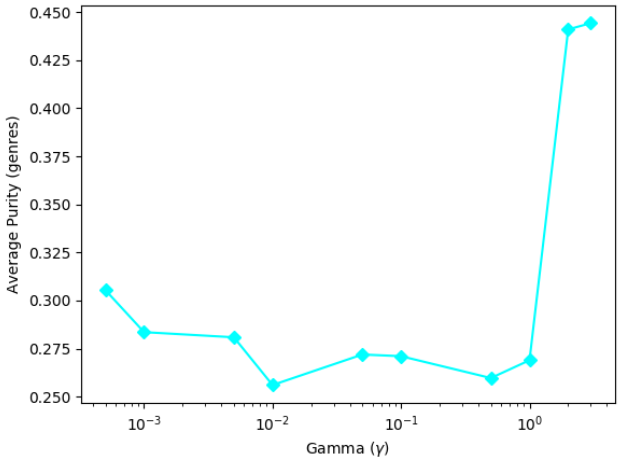
Figure 4.1b shows the corresponding amount of iterations until convergence. The amount of iterations needed to converge is decreasing when the value of

γ is increasing. Immediate convergence occurs for values of 0.05 and higher.

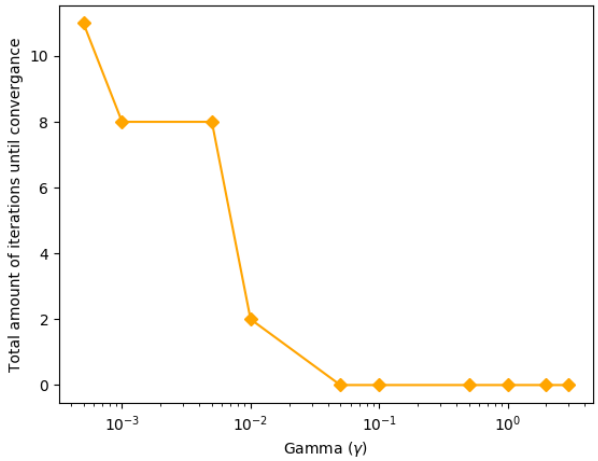
Figure 4.1c shows the amount of restarts needed to generate non-empty cluster partitions. Most value selections needed zero or one restarts. The outlier was the choice of 0.01, which required 10 restarts. Selections larger than 2.0 resulted in zero restarts.

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.

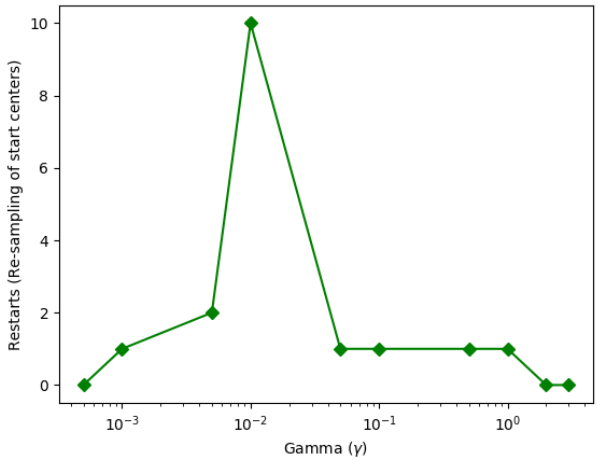
Figure 4.2 represents the feature weights of each cluster as colored grids (often referred to as a meshgrid) for various choices of γ. The grids in fig. 4.2 show the weight of a specific feature for a cluster as a colored rectangle. Yellow denotes a high feature weight, while purple denotes a low value. 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.



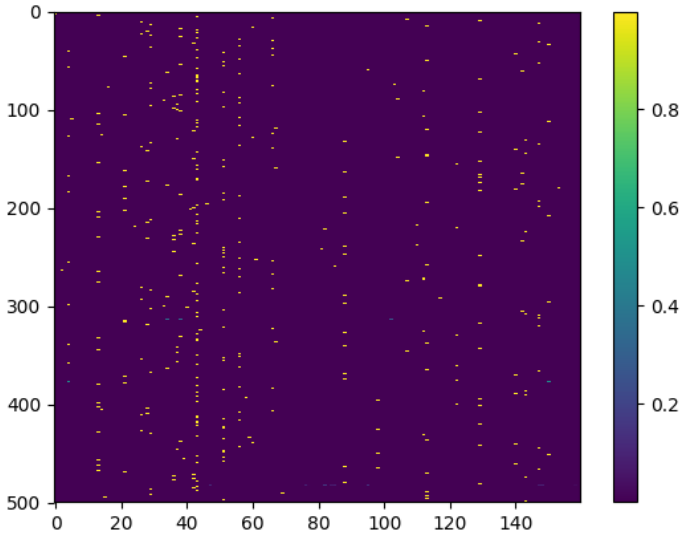
(a) Purity accuracy of EWKM given various values of γ ($k = 500$).



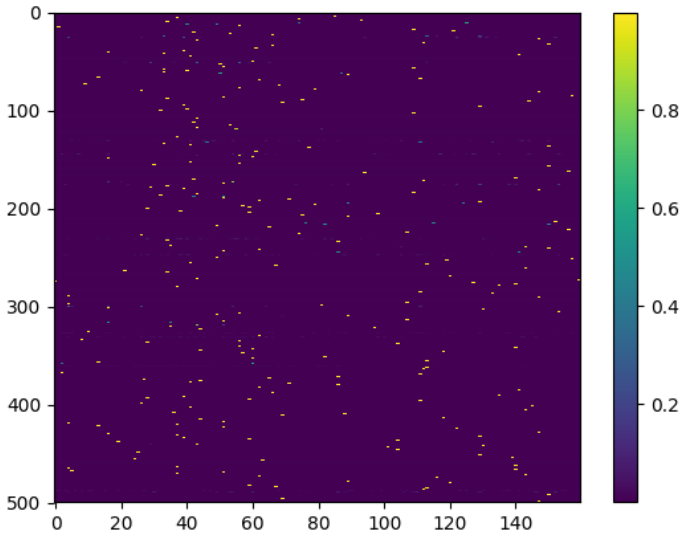
(b) Iterations until convergence of EWKM given various values of γ ($k = 500$).



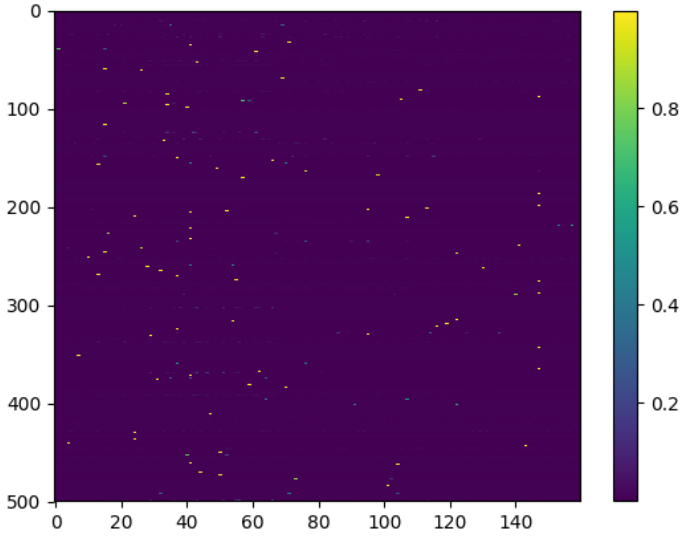
(c) Restarts of EWKM given various values of γ ($k = 500$).



(a) $\gamma = 0.0005$



(b) $\gamma = 0.05$



(c) $\gamma = 0.5$

Figure 4.2: Feature-weight meshgrid of EWKM ($k = 500$). A row represents a cluster subspace (feature-weight vector), and a column represents a feature-weight along clusters.

4.1.2 LEKM

The purity scores of LEKM are shown in fig. 4.3. With higher values of gamma 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. 4.4. Iterations kept increasing with γ until a value of 2.6. The largest amount of iterations needed was found at $\gamma = 2.2$ with 20 iterations. Values of 2.6 and larger resulted in an immediate convergence. Compared to EWKM, the γ's of LEKM are larger when immediate convergence occurs.

The feature weight grids of LEKM is shown in fig. 4.5, note that colors represent different values for each grid. We see that there are multiple features representing the cluster through the figures. Certain features are dominant in very few clusters (visible as a purple-dominant vertical line in the grid) while others are dominant in most clusters (visible as a yellow-dominant vertical line in the grid). The highest weights are also 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. 4.6. The histograms of the figure are the distribution of values of all feature weights in all clusters. The feature distribution of LEKM resembles a slightly leaning hill with an J-shaped peak, similar to what is mentioned in [4]. The deviation of weight values decrease with an increase of the γ value, leading to a more and more uniform distribution.

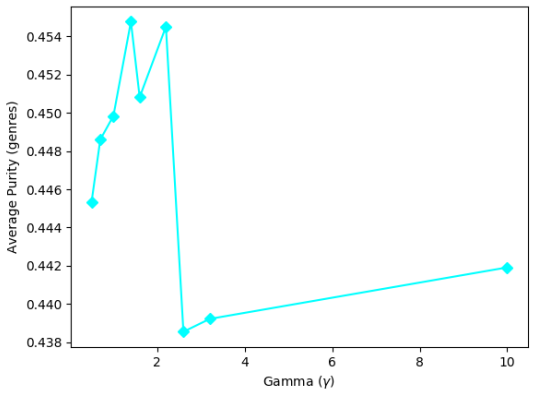


Figure 4.3: Purity accuracy of LEKM given various values of γ ($k = 500$).

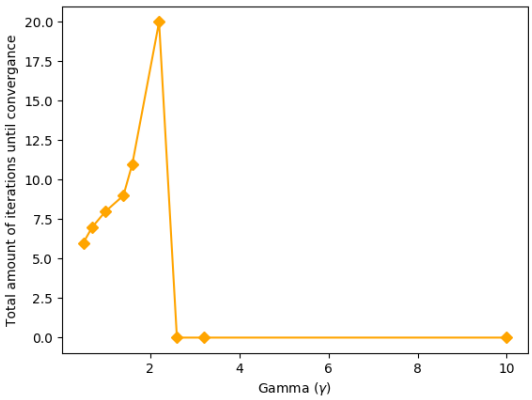
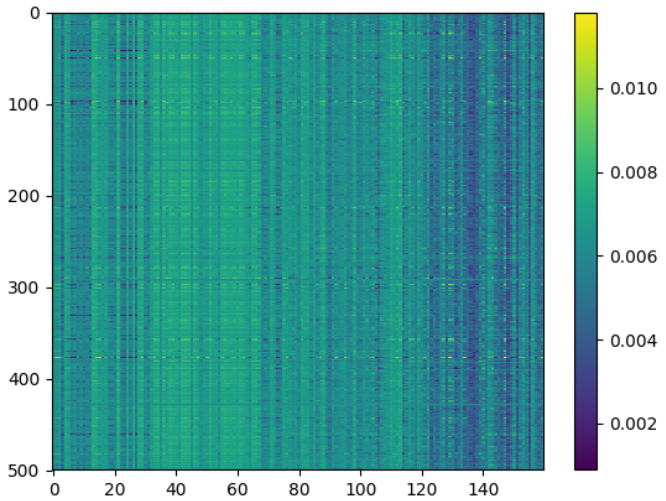
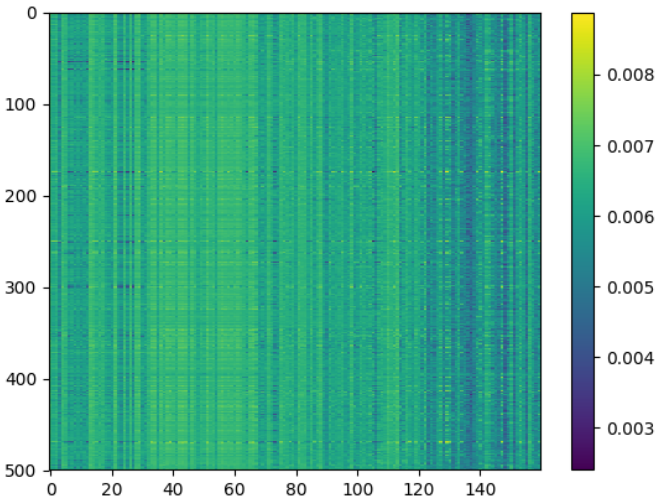


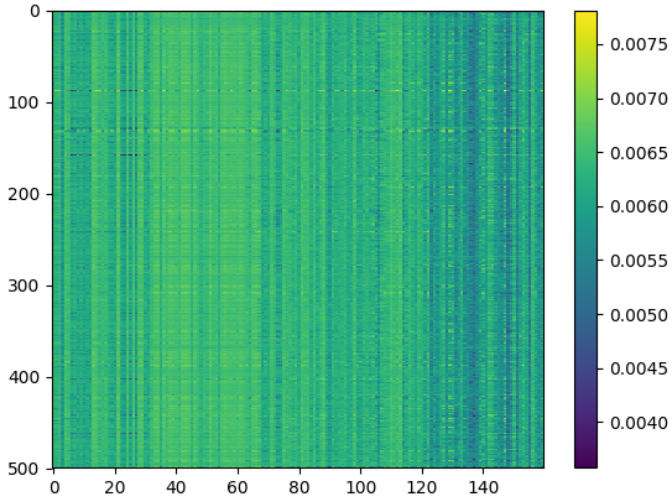
Figure 4.4: Iterations until convergence of LEKM given various values of γ ($k = 500$).



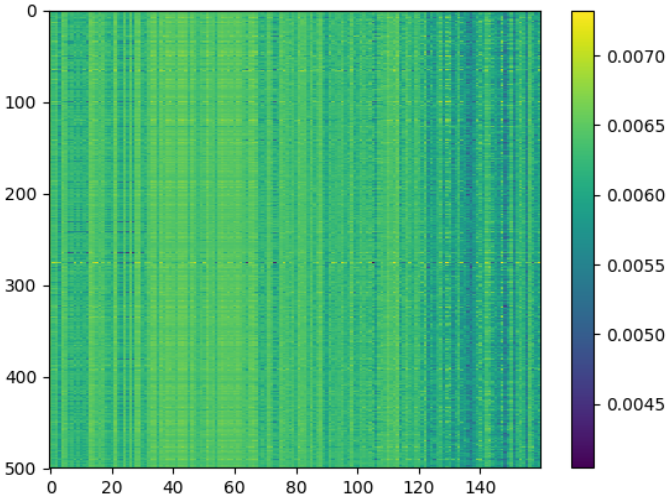
(a) $\gamma = 0.5$



(b) $\gamma = 1.0$

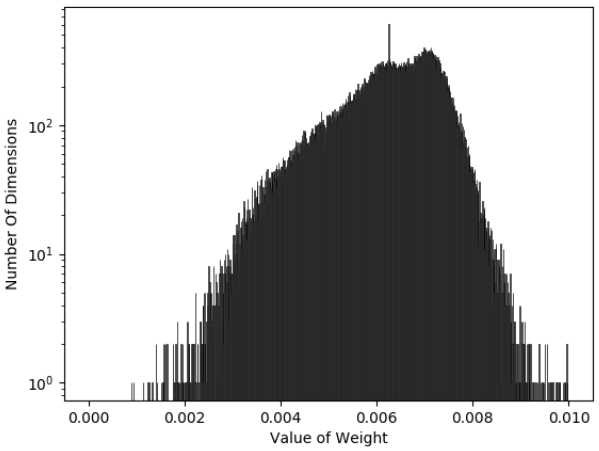


(c) $\gamma = 1.6$

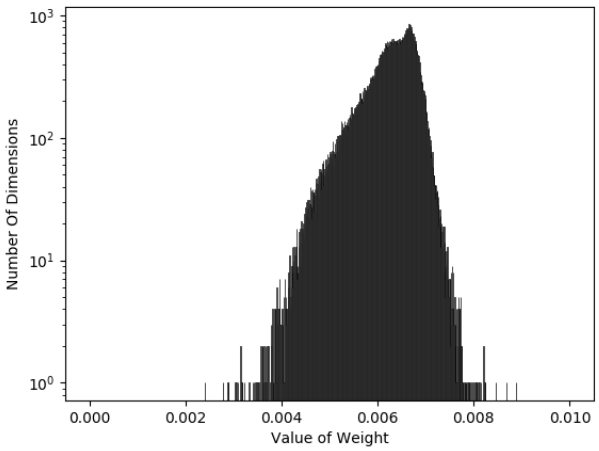


(d) $\gamma = 2.2$

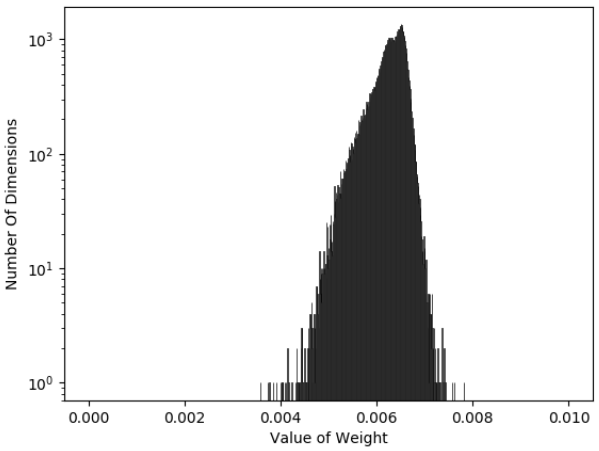
Figure 4.5: Feature-weight meshgrid of LEKM (k=500). A row represents a cluster subspace (feature-weight vector), and a column represents a feature-weight along clusters.



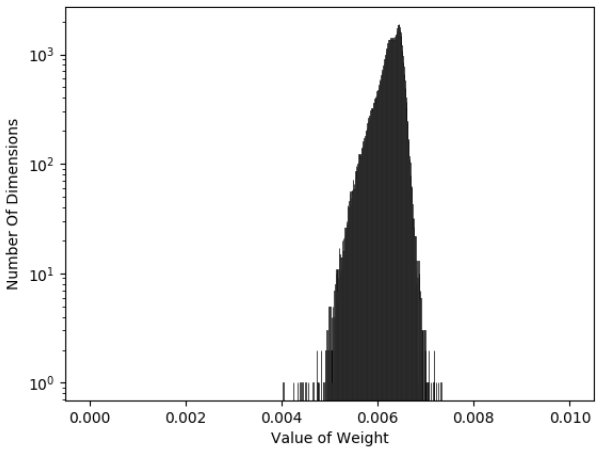
(a) $\gamma = 0.5$



(b) $\gamma = 1.0$



(c) $\gamma = 1.6$



(d) $\gamma = 2.2$

Figure 4.6: Distribution of feature weights values upon all clusters ($k = 500$)

4.1.3 FSC

Figure 4.7 shows how the algorithm performs in terms of purity on the dataset. From the figure we can see that the purity scores keep increasing and that the best score is found where $\lim_{\beta \to \infty}$, for our test a limit was set to $\beta = 30$ and was therefore the best performing value. The most significant increase occurs between $\beta = 1.8$ and $\beta = 2.02$ with an increase from 0.29 to 0.40.

The amount of iterations, as shown in fig. 4.8, are high for smaller values of β and equal two for larger β.

Figure 4.9 shows how feature weights are distributed along clusters. A selection of a lower β ($\beta = 1.5$) results in a single dominant feature weight for most clusters, similar to EWKM. As γ increased, more and more features were given importance in clusters. Again, note the scale of the meshgrids, some small amount of clusters had larger differences between feature weight values than other clusters, resulting in a more ”blue” than ”yellow” picture. A choice of $\beta = 1.5$ is shown to have feature weights that led to a single dominant weight per feature. Higher values resulted in uniform more dominant features. Similar to LEKM, certain features appeared dominant feature in many clusters, while other features were deemed unimportant for multiple clusters.

In terms of distribution as shown in fig. 4.10, the resulting distribution are positively skewed, but like LEKM the distribution ended up being more sharp for larger values of β.

4.1.4 k-means

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

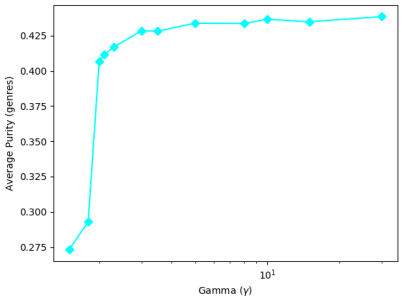


Figure 4.7: Purity accuracy of FSC given various values of γ ($k = 500$).

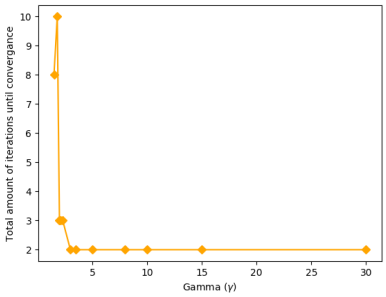
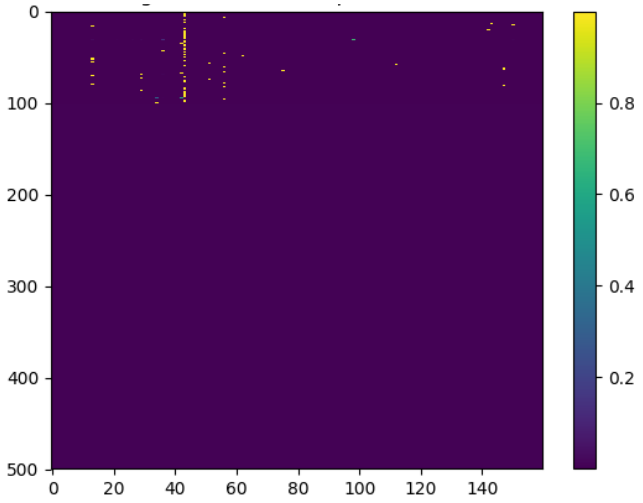
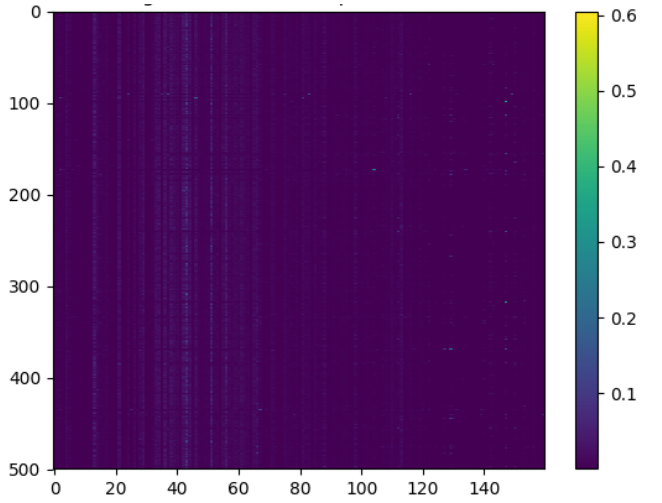


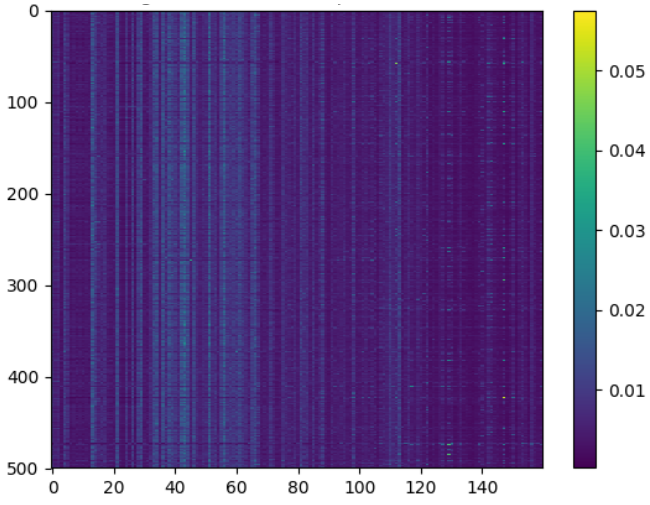
Figure 4.8: Iterations until convergence of FSC given various values of γ ($k = 500$).



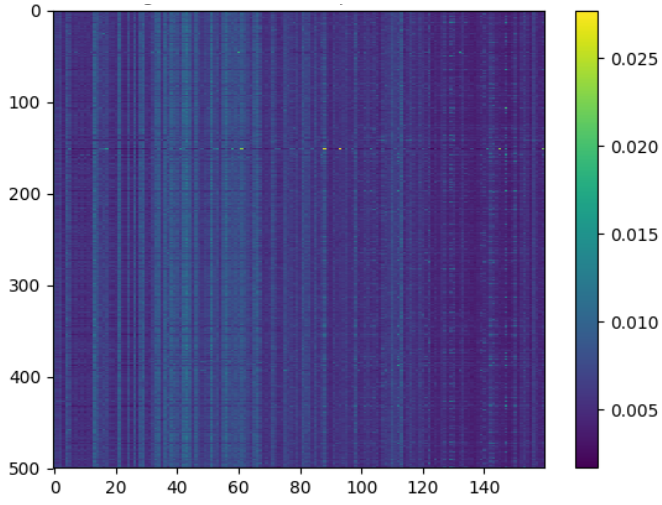
(a) $\gamma = 1.5$



(b) $\gamma = 2.1$

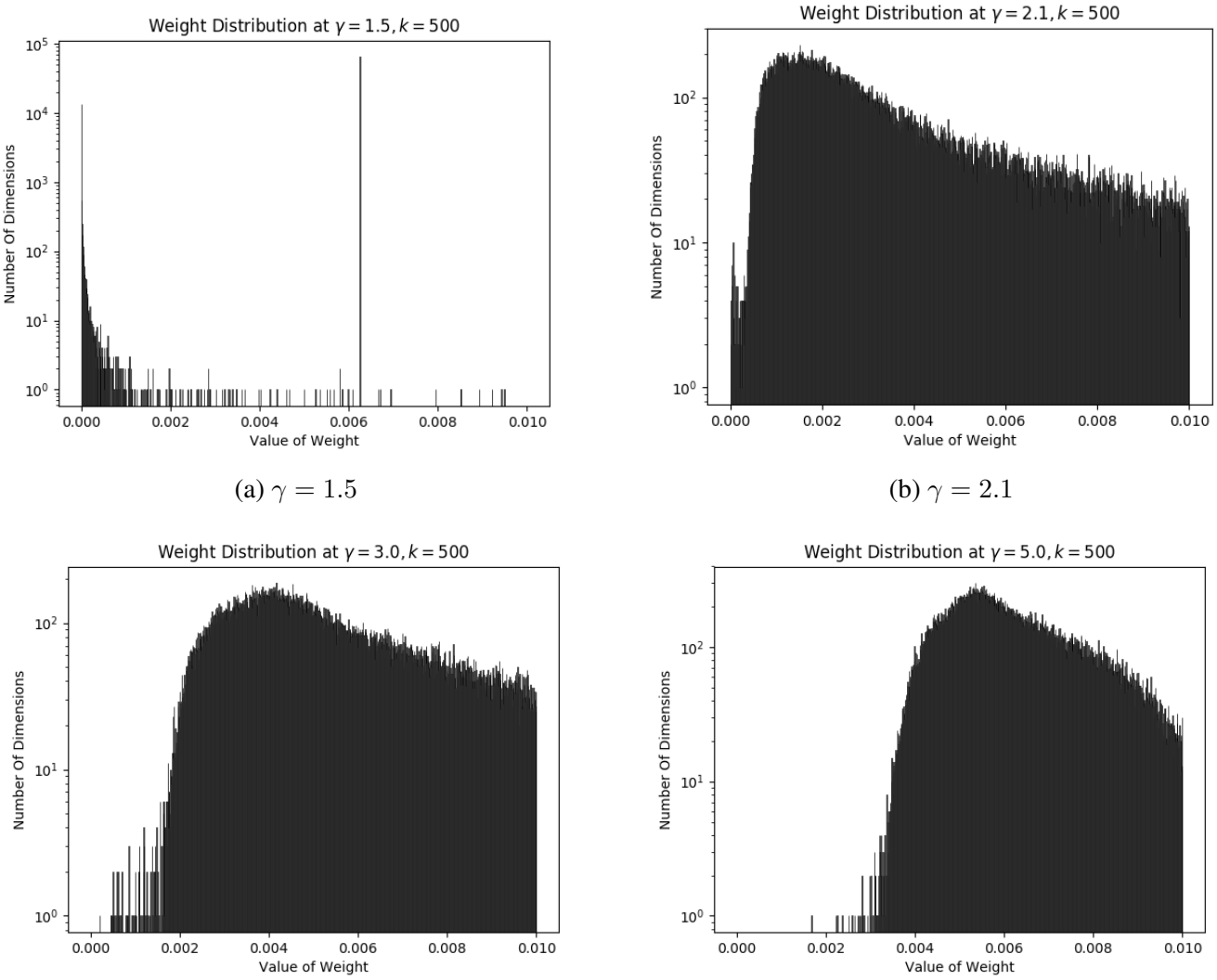


(c) $\gamma = 3.0$



(d) $\gamma = 5.0$

Figure 4.9: Feature-weight meshgrid of FSC ($k = 500$). A row represents a cluster subspace (feature-weight vector), and a column represents a feature-weight along clusters.



(a) $\gamma = 1.5$

(b) $\gamma = 2.1$

(c) $\gamma = 3.0$

(d) $\gamma = 5.0$

Figure 4.10: Distribution of feature weights values upon all clusters ($k = 500$)

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 $k = 500$ 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%). For 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, where scores represent the mean purity.

4.2 External Evaluation

The results of the blind test can be found in table 4.2. The mean scores and standard deviation of existing playlists, k-means and LEKM is shown for each criteria in the first-part of the table. A single-factor ANOVA test is then shown beneath the mean scores.

The clusters sourced from existing playlists were shown to have a higher mean than k-means and LEKM on all measured criteria except *playlist uniqueness*. k-means was shown to have marginally higher mean scores compared to LEKM for all measured criteria. A null hypothesis stating that all sources had the same mean, could however not be rejected for any criterion based on the single-factor ANOVA test shown in the same table. It can therefore not be proven to be any significant difference between the different sources.

There were also results obtained from the discussions of the blind test. Clusters from the algorithms were mentioned to have a core of songs that had an *obvious* theme. In addition to the core, there were two or three songs in the shown sample of the songs, that were *culturally* different according to the composers, i.e. the songs' artists had a different audience than the core songs. For well performing clusters — in terms of *General Quality* scores — that difference added depth to the playlist in a positive fashion, e.g. adding reggae to a hip-hop cluster . For bad performing clusters the mixins were off-putting for the theme of the cluster and made the cluster worse in terms of general quality, but also in terms of *Playlist Uniqueness*, e.g. adding country songs to an indie rock core.

	$\bar{x}(\sigma)$			
Source	General Quality	Audio Similarity	Cultural Similarity	Playlist Uniqueness
Existing Playlist	7.7 (1.2)	7.8 (1.5)	7.7 (0.6)	4.5 (3.5)
k-means	6.6 (2.5)	7.0 (2.3)	6.2 (2.9)	5.4 (3.2)
LEKM	6.4 (3.4)	6.4 (2.4)	6.0 (2.5)	3.8 (2.6)
	ANOVA			
Index	General Quality	Audio Similarity	Cultural Similarity	Playlist Uniqueness
$F_{0.05}$	0.21	0.45	0.47	0.37
F_{crit}	4.10	3.98	4.10	3.98
P_{value}	0.81	0.65	0.64	0.70

Table 4.2: Mean rating and standard deviation (in brackets) of five clusters given source type, along with single factor ANOVA scores that show that we cannot reject H_0 for any criterion.

Chapter 5

Discussion

The objective of this thesis was to answer whether clustering algorithms can be used to find high-quality novel musical themes. The problem statement asked if there was a performance difference between traditional methods (represented through *k-means*) and *SSC-algorithms* on the given high-dimensional dataset.

In this paper, k-means and SSC algorithms have been compared for a given musical high-dimensional dataset. Three SSC algorithms EWKM, FSC and LEKM were selected for the comparison with k-means. The given dataset was preprocessed to only include numerical audio features — based on neuron weights of a trained supervised neural network — and normalized using the *z-score*. The dataset was sampled to a size of $50k$. Two external evaluation indices were set up, a *purity* measurement based on a genre feature of the dataset, and a blind test involving a panel of expert judges. The purity measurement was used to find suitable input parameters — algorithm-specific ones, e.g. β and *k* (given a choice of 50, 100, 500). Additionally, the algorithm with the best purity score (LEKM) was used as the SSC-algorithm for comparison with k-means in the blind test. The blind-test was composed of clusters of different sources: Playlist, k-means, and LEKM. The expert judges — playlist composers — had to evaluate the clusters based on multiple criteria.

Resulting purity scores of k-means and the best performing clustering algorithm LEKM show a purity difference of less than 0.1% with optimal parameters on the sample dataset. EWKM performed significantly worse than other algorithms with regards to purity. As for the expert evaluation, it was shown that a null hypothesis — $H_0 = [\mu_{pl} = \mu_{SSC} = \mu_{k\text{-}means}]$ — cannot be rejected

based on the evaluation scores given by the playlist composers that were subjected to the test.

5.1 General Discussion

Based on previous work [4, 11], a high-dimensional dataset such as the given dataset, should see SSC-algorithms improve clustering performance by lessening the weight of irrelevant features through automatic feature weighting. The results of this paper did not show improvements in using a SSC-algorithm on the dataset: there was negligible difference between k-means and the best performing SSC algorithm. The results are unexpected as SSC algorithms are made for high-dimensional datasets.

Properties of the dataset could have made it less suitable for SSC algorithms, a reason for the difference between the results obtained and what was shown in [4]. One possibility is that the dataset had in a sense already had its features selected beforehand: the 160 weights selected as features, were already sampled from the set of weights in the neural network. SSC algorithms could theoretically have boosted the performance even with the feature sampling, by finding the unique subspaces of each cluster, but it did not. There was no significant improvement of purity performance with LEKM, and a downgrade in performance occurred with the usage of EWKM. It can be concluded that SSC-algorithms are not beneficiary for the given dataset. On the other-hand, it can be questioned whether the dataset as it was pre-sampled is a good representation of a high-dimensional dataset. The results could have been more representative if another publicly available dataset was used. As it stands now, there is a limitation on how the results can be extended to other datasets.

The popular soft-subspace clustering algorithm of EWKM performed poorly on the dataset. The possible choices of γ were restricted to lower values in order to avoid immediate convergence. For values that did not immediately converge, the algorithm suffered from single feature-weight dominance in clusters, making it hard to represent the clusters in an impactful manner. The algorithm had clearly failed the task of feature reduction, which in turn had resulted in worse purity scores. This shows that the algorithm should not be used for just any numerical high-dimensional dataset.

The problem of dominant weights was not mentioned in the paper introducing the EWKM algorithm [4]. The feature weights were supposed to be of a normal distribution, with some features more and less important per cluster.

FSC had the accuracy scores increase in comparison to EWKM. The best scores were found with an increasingly large β. A larger β resulted in feature weights becoming more uniform in each cluster. What essentially FSC offered in feature weighting, was disregarded when selecting larger values, i.e. $\beta = 15, 30$. From this standpoint the best (and largest) β was not true to subspace clustering, there was no significant feature weight dispersion, but for this dataset it was not necessary according to the purity scores. A reason could be that the algorithm failed to determine the important features from the unimportant features, and clusters were given subspaces of actually unimportant features. For this case, it would be better to allow all features the same weight. This would also mean that the algorithm failed in terms of feature reduction.

LEKM, which created feature weights that were normally distributed on each cluster, performed similarly to k-means. It had one advantage to k-means. A smaller fraction of features had an impact when calculating the distance between data-object and cluster center, when using the best γ. LEKM allowed more features to be of a significant weight in comparison to EWKM, which shows the benefit of using log-transformed distances. Clusters could be dimensionally reduced to a smaller subspace without impacting the accuracy negatively unlike FSC. The algorithm is based on this, suitable for this dataset and other datasets of high-dimensionality.

The chosen external validation index — Purity with *Genres* as a ground truth — promotes the choice of values based on genre homogeneity. It is probable that more homogeneous genre clusters are in general of better quality than those with more genre dispersion. What it failed to do is to promote genre novelty — new types of genres based on unique properties. A better choice of parameters could have been found if the index had taken the novelty into account, however, genre was the only label provided for the dataset.

That the highest value of k (500) would have the highest purity value was not unexpected — smaller clusters tend to allow for higher purity accuracy. The F_1 score could have avoided such bias, but as mentioned in the background, would involve more complex computations. The choice of candidate k's, which were decided by the author to be 50, 100, and 500, could have been extended with more options for the higher probability of finding the optimal k. This suggested extension would have exponentially increased the parameter selection process, as each *k* requires an array of algorithm-specific parameter values to be tested.

An intentional limitation was to run algorithms once for each parameter combination on the sample dataset. There were two problems of this limitation: We could not with statistical significance determine whether there was a purity difference between algorithms, as performance differences could be caused by initial cluster sampling. As for the second problem, we assumed that the parameters chosen are representative for the whole dataset when no such assumption can be made.

5.2 Evaluation Discussion

The evaluation scores of the blind-test indicate 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 some songs did not fit the general theme of the cluster due to cultural differences, such occurrences could be tuned out by allowing composers to filter out songs by artist or genre. It can therefore be stated that the audio features selected in the preprocessing were indeed enough to cluster the 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. In terms of mean score a difference is shown between k-means and LEKM, k-means has an overall higher mean scores than LEKM.

The output of the single-factor ANOVA tests yield a result that cannot reject a null hypothesis for any of the given evaluation criteria. It cannot be statistically determined if there is a significant score difference between the clusters of the different sources for the chosen criteria. The inability to reject the null hypothesis can be seen as a positive result: LEKM and k-means are performing similarly to existing playlists. The results are however, contradicting expected behaviour: The null hypothesis is not rejected for *Cultural Similarity*, yet the algorithms are only trained on audio-based features. In other words a difference is expected for existing playlists and the algorithms in regards to *Cultural Similarity*.

5.3 Ethics, Sustainability and Societal Aspects

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 from another country of origin which 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 country of origin should have merit or not. From a cluster quality standpoint excluding some songs, for a better precision is a worthy trade-off. In this thesis only audio data is processed for clustering, which forces the algorithm to only compare songs based on audio.

High-dimensional datasets are frequent in the real-world. The comparison between the lesser known SSC-algorithms and k-means on the given dataset are beneficial for researchers and data-scientists looking for the possibility to cluster high-dimensional data. This thesis provides results on a new real-world dataset, without any bias towards a self-published algorithm. Additionally, problems of using the given algorithms on the dataset are discussed, which helps highlight possible problems on similar datasets.

5.4 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 [24] 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 neg-

atively. 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 be to scale the algorithms 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 [10] 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 take both intra-cluster distance and inter-cluster distance into account — 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 into account.

Finally, to allow the results to be statistically tested, algorithms should be run multiple times for each possible candidate combination on one or more representative samples of the dataset, and judges should be given more clusters to rate.

Chapter 6

Conclusions

Multiple SSC algorithms were tested on the given high-dimensional musical dataset — sampled to a size of $50k$, and processed to only hold numerical features. LEKM turned out to be the best performing *SSC*-algorithm in terms of genre purity. Comparisons between k-means (a traditional algorithm) and LEKM show that there is no significant performance difference between the algorithms, i.e. feature-weighting did not improve the performance, possibly due to the nature of the dataset. 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 general quality similar to existing playlists. Based on statistical analysis, similar performance were found for k-means and SSC. The results indicate that the generated clusters could be used as a tool when composing new playlists on the musical library that is our dataset. Future work should see a comparison between k-means and LEKM on a larger sample of the dataset, to see if there is a benefit for feature weighting on a larger scale. A larger sample would need the code of LEKM to be rewritten for a distributed network. From a broader perspective on how SSC-algorithms performs on high-dimensional datasets, another dataset comparison is needed.

Bibliography

[1] Clara Gustafsson. “Sonic branding: A consumer-oriented literature review”. In: *Journal of Brand Management* 22.1 (2015), pp. 20–37.

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

[3] Christine Miaskowski et al. “Subgroups of patients with cancer with different symptom experiences and quality-of-life outcomes: a cluster analysis.” In: *Oncology nursing forum* 33 5 (2006), E79–89.

[4] 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.

[5] 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.

[6] 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.

[7] Zhaohong Deng, Kup-Sze Choi, Fu-Lai Chung, and Shitong Wang. “Enhanced soft subspace clustering integrating within-cluster and between-cluster information”. In: *Pattern Recognition* 43.3 (2010), pp. 767–781.

[8] Carlotta Domeniconi, Dimitrios Gunopulos, Sheng Ma, Bojun Yan, Muna Al-Razgan, and Dimitris Papadopoulos. “Locally adaptive metrics for clustering high dimensional data”. In: *Data Mining and Knowledge Discovery* 14.1 (Feb. 2007), pp. 63–97.

[9] Joshua Zhexue Huang, Michael K Ng, Hongqiang Rong, and Zichen Li. “Automated variable weighting in k-means type clustering”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2005), pp. 657–668.

[10] 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.

[11] 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.

[12] 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.

[13] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Tech. rep. 1996.

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

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

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] Raymond T Ng and Jiawei Han. “CLARANS: A Method for Clustering Objects for Spatial Data Mining”. In: *IEEE Transaction on Knowledge and Data Engineering* (2002).

[23] 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.

[24] 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.

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

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

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

[28] Zhaohong Deng, Kup-Sze Choi, Yizhang Jiang, Jun Wang, and Shitong Wang. “A survey on soft subspace clustering”. In: *Information Sciences* 348 (2016), pp. 84–106.

[29] 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.

[30] Elaine Y Chan, Wai Ki Ching, Michael K Ng, and Joshua Z Huang. “An optimization algorithm for clustering using weighted dissimilarity measures”. In: *Pattern Recognition* 37.5 (2004), pp. 943–952.

[31] Liping Jing, Michael K Ng, Jun Xu, and Joshua Zhexue Huang. “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.

[32] 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.

[33] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Vol. 16. 1. Cambridge university press, 2010, pp. 100–103.

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

[35] 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.

[36] Graham Williams, Joshua Zhexue Huang, Xiaojun Chen, Qiang Wang, and Longfei Xiao. *wskm: Weighted k-Means Clustering*. 2015.

[37] *NumPy C-API — NumPy Manual*.

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