This is the English title

EMIL JUZOVITSKI

Master in Machine Learning
Date: April 9, 2019
Supervisor: Johan Gustavsson
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Host company: Soundtrack Your Brand AB
Swedish title: Detta är den svenska översättningen av titeln

Abstract

English abstract goes here.

Sammanfattning

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

Contents

Chapter 1

Introduction

We use the *biblatex* package to handle our references. We therefore use the command `parencite` to get a reference in parenthesis, like this [**DENG201684**]. It is also possible to include the author as part of the sentence using `textcite`, like talking about the work of **DENG201684**.

1.1 Research Question

Chapter 2

Background

Chapter 2. intends to introduce the theory that is necessary to understand the research topic.

2.1 Objective of Clustering

Given the inputs of a Dataset $\mathcal{D} = \{X_1, X_2, ..., X_n\}$ — where $X_i = [x_{i1}, x_{i2}, ..., x_{im}]$ is a single data point with m attributes — and k is a positive integer, the objective of partition clustering is to divide objects into k disjoint clusters $C = \{C_1, C_2, ..., C_k\}$ [1].

The above objective is as stated just for partition-based clustering (See 2.3.2) . For density-based clustering the objective is to separate local dense areas from noise for more details (See 2.3.3) [2].

2.2 Similarity/Distance Measures

The relative closeness between two points X_i, X_j is quantified numerically through a similarity measure $s(X_i, X_j)$ [3]. A high value indicates that the points are close, while a low measure indicates that the points are dissimilar. Values of $s(X_i, X_j)$ go between 0 and 1.

The dissimilarity between the same two points can be described through a distance measure $d(X_i, X_j)$ — $d(X_i, X_j) \geq 0$. Distance measures are often used

with quantitative data i.e. numerical data. Similarity measures are frequently used when data is qualitative e.g. categorical and mixed data [4]. Both measures are symmetric i.e. $s(X_i, X_j) = s(X_j, X_i)$.

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

A point X_i can be compared with another point X_j or a cluster representation C_l — frequently used in partitioning algorithms. A representation can be the mean of all points in the cluster, or a centroid — the most central point in the cluster [5, 6]. For categorical data, the *mode* can represent the cluster — The cluster is represented by the most frequent value of each attribute in the cluster [7]. C_l in these representations can be handled as a point.

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

2.2.1 Numerical Similarity Measures

A frequent choice for a distance measure is the Euclidean measure [8]. The Euclidean distance is shown in 2.1. Often, the Squared Euclidean is used (2.2).

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

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

The Euclidean distance is a special case of the Minkowski distance where $q = 2$. The Euclidean and other Minkowski distances require normalized data to obtain good performance [8].

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

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

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

$$\tag{2.5}$$

The mentioned distances can be converted to a similarity distance — with values between 0 and 1 — by using the Gaussian kernel as shown in 2.6 [9].

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

2.2.2 Categorical Similarity Measures

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

Nominal data — e.g. chemical elements in a periodic table — are either identical or different. Ordinal data — e.g. shirt sizes — can be more or less similar. Additionally, nominal data does not have to be information balanced either — some values are more important than others — e.g. if two songs share the same artist it is more information than if they do not [6].

Most clustering algorithms that cluster categorical data do not take account the differences between the data types mentioned above. There are distance measures that do (see *daisy function* in [6]) but implementing them in a clustering algorithm would increase the complexity of the algorithms. The rest of the similarity measures below considers ordinal data nominal. From this point onward, both nominal and ordinal will be referred as categorical data, and treated as nominal data.

In its purest form a categorical similarity is simply yes or no, as shown in 2.7, i.e. Are two attribute values the same? [5, 6]

To define the similarity between a point and a cluster representation, the number of mismatches can be used as shown in 2.8 [1, 7]. Where a cluster representation is defined by the most frequent attribute values of the points in the cluster. This method is used in *K-modes* and the cluster representation is referred as a *mode*

Another way is to compare the values of X_i and the values of all data points $\forall X_k \in C_l$ for each attribute as shown in 2.10, 2.11 [5, 9]. In this method a cluster representation is not necessary. To allow fast computations of distances between points and clusters the frequency of each value of each cluster is stored in a frequency matrix. w_j is an attribute weight defining how important each attribute is. The weight is created by calculating the entropy of the attribute. How to calculate the entropy is mentioned in section 2.4.

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

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

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

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

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

2.2.3 Mixed Similarity Measures

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

Huang proposes the similarity measure shown in 2.12 [1]. w_l determines how important categorical data is compared to its numerical counterpart for cluster

l. In *K-Protoypes* a simplification is made local w_l for each cluster l with a single global weight w. The weight w is a user-defined input.

Cheung and Jia [9] propose representing numerical data as an vector X_i^r where numerical data is denoted by r, while letting each categorical attribute have a single weight . The categorical weights are automatically weighed through information gain, as such the algorithm is a *feature weighting* algorithm. See section 2.4 for more. By defining the numerical attributes as a single numerical vector the amount of attributes is defined as the number of categorical attributes, plus 1. X_i.

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

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

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

2.3 Clustering Algorithms

Clustering algorithms are often divided into categories. Hierarchical-, Partition- and Density-based clustering are the largest categories [10]. The algorithms below are exclusively numerical if not stated otherwise.

2.3.1 Hierarchical Clustering

A Hierarchical algorithm generates a set of nested clusters, also known as a dendogram [8, 10]. There are two approaches two hierarchical clustering. Agglomerative- and divisive-hierarchical clustering.

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

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

CURE, BIRCH and, ROCK (categorical) are popular algorithms of hierarchical clustering. The creation of a dendogram in these clustering algorithms results in a time complexity that make the algorithms less suitable for larger datasets however, with sampling techniques as used in CURE and ROCK bigger datasets can be managed [8, 10]. In addition to creating a dendogram a positive is the possibility to cluster with any arbitrary shape.

2.3.2 Partition Clustering

In partition clustering a partition of clusters U is found by iteratively optimizing a cost function [8, 10, 11]. The algorithms include two iterative steps of assigning each point to the most similar cluster center representation, and, updating a cluster center representation — which is a mean in *K-means* and the most central point in *K-medoids*.

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

2.3.2.1 K-means Family

K-means is arguably the most common clustering algorithm. The cost function of which K-means is optimized on is shown in 2.14, 2.15. Either 2.14 is minimized or 2.15 is maximized [11]. There are many algorithms based on K-means, they are referred as the K-means family of clusters [12]. The family includes *K-Modes* [7], *K-Prototypes* [1], *W-K-means* [11], *E-W-K-means* [13].

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

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

U is a partition matrix of size $N \times K$. The Partition matrix shows in which clusters point X_i is in. In hard clustering one point can only exist in a single cluster C_l. Thus, $u_{il} = [0,1]$ and $(\sum_{l=1}^{k} u_{il}) = 1$.

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

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

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

P2. Update the mean for each cluster.

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

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

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

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

- For numeric values solved by obtaining the average:

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

- For categorical values, one option is defining the center as mode [7], which is solved by:

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

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

- For mixed values, one option is representing the cluster as a Prototype[1, 12]. The solution is simply to use 2.17 for numerical attributes and 2.18 for categorical.

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

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

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

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

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

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

Different similarity functions determine what K-means family algorithm the optimization represents [11]. Euclidean distance would result in *K-means*. The categorical similarity shown in 2.8 results in *K-modes* [7]. If the similarity is the mixed type shown in 2.12 the optimization represents *K-Prototypes* [1].

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 [14]. Compared to 2.14 the difference is that the distance function compares a point X_i with the medoids C_l^m of that cluster. Using medoids instead of a mean results in a more robust clustering performance. The tradeoff is *K-medioids* runtime. *K-medioids* is usually implemented through the *PAM* algorithm which has a time complexity of $O(k(n-k)^2)$ per iteration, whereas the basic *K-means* has an iteration complexity of $O(nk)$.

Extensions of *PAM*, such as, *CLARA*, and, *CLARANS* are approaches to improve the scalability of the clustering type by clustering on a sample representation which improves the run time [14]. *CLARANS* improves the simple sampling procedure in *CLARA*. It defines a sample of cluster medoids as a node in a graph which is the whole dataset. Neighbouring nodes differ by

one medoid. *PAM* traverses all neighbours for each node it traverses to find the node with minimum distance between points. CLARA can be seen as only finding the minimum of a sub-graph containing only the sample points. *CLARANS* samples dynamically neighbours while traversing the graph not restricting the traversal to a subgraph.

2.3.2.3 Determining K and K-initialization

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

A way to determine the K is by running the algorithm with different values of K and then through an internal criterion measure (see 2.5.1) decide the best K for the dataset [1, 15]. Running an algorithm multiple times makes the process of clustering multiple times slower. When clustering a large dataset this is something to avoid.

Sampling is used in ClARA, CLARANS [14] to find medoids for the whole dataset. The idea is that a small($40 + 2k$ points) randomly uniform sample of the dataset should represent the whole dataset. The same sampling process can be used to find *K* and so, finding a k for a sample would be finding an approximation of *K* for the whole dataset. The algorithm still has to be ran on the sample multiple times and evaluated against the inner criterion, but the time to compute the clusters for a sample rather than the whole dataset is significantly less.

Cheung and Jia propose another approach [9, 16]. Here, competitive learning is used — giving every cluster a weight that together with the distance function determines the chance of a datapoint becoming a member of the cluster. When a datapoint is asssigned to the cluster, that cluster's weight and its neighbour's weight increase, and the rest of the clusters' weights decrease. Eventually, some clusters disappear. The positive aspect of this approach is determining the amount of clusters occurs during the clustering process removing the need to cluster the dataset multiple times. Note: A user input of maximum number of clusters $\mathbf{k}^*$ is required.

In addition to choosing k, picking good initial points is also a problem that should be considered. Good initial clusters allow for a faster convergence while bad can result in convergence on a suboptimal result[16, 17], forcing multiple clusterings to obtain a result of confidence. While random uniform

sampling is a way to initialize K centers there are more robust solutions, that can exclude picking e.g. outliers allowing the result to be less random.

K-means++[17] proposes replacing uniformly at random sampling with the following steps:

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

In [16] a specific approach for mixed data is mentioned.

2.3.3 Density-Based Clustering

Density-based clustering algorithms define clusters to be higher-density areas — a local area with a relative high number of data points i.e. higher density than noise [2, 8, 10, 11]. DBSCAN [2] is the most well known clustering algorithm where each data point in a cluster must have a user defined *MinPts* in its $\mathcal{E}$-neighborhood, where $d(X_i, X_j) < \mathcal{E}$ and $\mathcal{E}$ is user-defined. The shape of the created clusters is defined by the chosen distance measure.

The algorithms of this type perform well on larger datasets especially on spatial data [2].

2.4 Weighed clustering

K-means assumes that all attributes are of the same importance [6]. If one were to scale the range of one attribute by two, it would become twice as important for the clustering result. To allow attributes of naturally smaller value ranges the same importance as attributes with larger value ranges, attributes are often normalized.

After normalizing the attributes, attributes can be given a weight based on the perceived importance of the attribute. Weighting attributes is necessary for good performance on most datasets. Using irrelevant attributes damages the clustering performance [6]. It is worse than not using the attribute at all.

Moreover, increasing the importance of a relevant attribute allows the clustering algorithm to perform better.

Deciding on how to weight attributes is hard. A simple solution is using technical expertise to assign attribute weights. In e.g. data-mining, a dataset is often of high-dimensionality as it may be generated from a database with hundreds of tables and columns [13]. The high degree of dimensions makes manually determining the weights of the attributes almost impossible.

A fairly popular way to handle high-dimensional data is through the use dimensionality-reduction techniques e.g. PCA [18, 19]. 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.

Automated weighting by the clustering algorithm can also solve the problem. In simple terms, an additional *attribute weight* variable is added to the cost function given in 2.14. How the cost function changes from 2.14 depends on what concepts we use to automate the weighting.

The simplest form of attribute weighting is *feature weighting* — Each attribute has a single weight of importance determined by the cost function during clustering. One algorithm of this class is *w-k-means* which extends *k-means* by adding a weight to each attribute [11].

w-k-means uses the cost function in 2.19 which introduces a new varaible W — the weights of each attribute — to be optimized. β is a hyperparamater for the importance of weights. In *w-k-means* W is optimized on the variance of the inter-cluster distances. **P3.** is the additional problem of optimizing W and is shown in 2.21. An additional step in the k-means algorithm is added for the optimization:

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

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

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

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

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

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

The function can be modified to allow clustering on mixed data [16].

Instead of optimizing the weights on only inter-cluster distances — which is tricky for categorical data — The weights can be optimized by information gain, more specifically the entropy. Entropy can be referred as the amount of "disorder" in a system. Entropy is used in [9, 13]. Using the entropy works for both numerical and categorical data. In [9] the importance of an categorical attribute is defined as the average entropy of each value of the attribute. This is shown in 2.22. To allow weights in the range of $\{0, 1\}$ the importance is divided by the total importance of all attributes in the dataset as shown in 2.23.

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

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

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

The above methods are arguably over-simplified. Attributes can correlate less or more, and be of different importance in each cluster . Take medical deceases as clusters for example. The chance of some deceases vary if a patient has recently traveled to a tropical climate. For other deceases that info (attribute) does not matter.

To allow clusters to have their own attribute weighting, subspace clustering is used [13, 16, 20, 21]. Subspace clustering lets clusters have their individual subspaces, hence the name. Subspace clustering can be divided into hard-subspace clustering and soft-subspace clustering.

Hard-subspace clustering tries to find the exact subspaces [13, 16, 20, 21]. Soft-subspace clustering clusters the data points globally but with weight vectors having been assigned to each cluster. Each vector represents the attribute

weights for that cluster. Hard-subspace clustering is a much more time consuming task than its soft-subspace clustering counterpart.

Automated-Weighting Algorithm (*AWA*) [22] is a soft-subspace approach similar to *w-k-means*, the difference is that for the cost function of AWA w_j^β is replaced with w_{lj}^β — a weight for an attribute in a cluster C_l. The algorithm does not work when the variance of an attribute in a cluster is zero as the learning rules denominator becomes zero. *FWKM* [23] solves the problem by adding a small constant to the distance function, forcing the variance to not be zero.

Entropy can also be used in soft-subspace clustering. In Entropy Weighed K-means (*EWKM*) [13] — A soft-subspace clustering method — the inter-cluster distance is combined with the negative entropy to create the cost function shown in 2.26. As in *W-k-means* **P3.** and Step **4.** becomes optimizing W for the function while fixing U and C. γ is a user-inputted variable used to control the size of the weights, For $\gamma > 0$ The smaller D_{lj} the more important attribute A_j is to cluster C_l. A modified version of *EWKM* is *IEWKM* [24]. It specifies a cost function for both numerical and categorical data.

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

$$\textit{where:} \quad \gamma \geq 0 \tag{2.25}$$

$$w_{lj} = \frac{\exp(\frac{-D_{lj}}{\gamma})}{\sum_{t=1}^{m} \exp(\frac{-D_{lt}}{\gamma})} \tag{2.26}$$

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

WOCIL [16] is a Mixed-data soft-subspace *k-means* type clustering algorithm. To determine W — which in this case is a matrix, due to the method being a subspace method — the inner criterion is used. To determine the inter cluster similarity, the distribution of attribute $A_r \in C_l$ is compared to the distribution of A_r outside of C_l. In this case, Hellinger distance is used the quantify the dissimilarity between the two distributions. For categorical data the distribution is assumed as 2... and for numerical attributes a Gaussian distribution is assumed. The intra-cluster similarity is then found through 2.29.

$$M_{lj} = \frac{1}{\sum_{X_i \in C_l} 1} \sum_{X_i \in C_l} s(x_{ij}, C_l) \tag{2.28}$$

$$H_{lj} = F_{lj} M_{lj} \tag{2.29}$$

$$w_{lj} = \frac{H_{lj}}{\sum_{t=1}^{m} H_{lt}} \tag{2.30}$$

2.5 Evaluation

There are two main ways of which clustering are evaluated: Internal- and external-criterion.[25] In some research the categories are extended with the relative-criterion.[26]

2.5.1 Inner Criteria

Internal criterion is an unsupervised validation approach and can be described as evaluating the results without respect to external information [26]. An internal criterion tries to verify the objective of the clustering algorithm on the dataset. For example: Making sure that points assigned to the same cluster are in general more similar than points outside of the cluster.

2.5.1.1 Average Silhouette Coefficient

The average silhouette coefficient upon all datapoints shown in 2.35, is one way to evaluate the internal criteria [27]. A single silhouette coefficient shown in 2.34, looks at a data points intra-cluster similarity — similarity to points within the same cluster, and, inter-cluster similarity — similarity with point outside of the cluster. $\overline{s}_{co}$ goes between -1 and 1 where a high value (close to 1) indicates a natural clustered dataset.

When $s_{co}(X_i)$ is a high value (close to 1) a point is well clustered i.e. the intra-cluster similarity is high relative to the inter-cluster similarity [27] . The same reasoning is extended to $\overline{s}_{co}(\mathcal{D})$ where a high value is a well clustered dataset.

$$d_{avg}(X_i, C_l) = \frac{\sum_{X_k \in C_l} d(X_i, X_k)}{Count(X_k \in C_l)} \tag{2.31}$$

$$a(X_i) = d_{avg}(X_i, C_a) \text{ , } \textit{where } (X_i \in C_a) \tag{2.32}$$

$$b(X_i) = min_{C \neq C_a}(d_{avg}(X_i, C)) \tag{2.33}$$

$$s_{co}(X_i) = \frac{b(X_i) - a(X_i)}{max(a(X_i), b(X_i)} \tag{2.34}$$

$$\overline{s}_{co}(\mathcal{D}) = \frac{\sum_{i=1}^{N} s_{co}(X_i)}{N} \tag{2.35}$$

2.5.1.2 Calinski-Harabasz Index

Calinski-Harabasz is for measuring the validity

2.5.2 External Criteria

External criterion is an supervised validation approach and can be described by the following sentence: Validation of the results by imposing a pre-defined structure on the dataset i.e. data not used for generating the clustering results [26]. As clustering analysis is an unsupervised learning method, it is often hard to assess the criteria — There is often no test data available for a new dataset. Still, to quantify how well the clustering approach works on a new dataset defining a ground truth is essential. One way is to use judges, experts in the field [25]. Given a ground truth an external measurement can be deployed.

2.5.2.1 Measurements

Purity is a measurement for the external criterion. It measures the ratio of the most dominant class in a cluster. The data-points are labeled with a class beforehand on the ground truth [25]. Equation 2.36 defines the measurement. The equation does not penalize small clusters as such small clusters produces a high score.

$$P(W, C) = \frac{1}{N} \sum_{k} max_j \left| w_k \cap c_j \right| \tag{2.36}$$

Where:
$W = \{w_1, w_2, ..., w_k\}$ is the set of clusters, and, $C = \{c_1, c_2, ..., c_k\}$ is the set of clusters.

Another measurement is the *F-measure* shown in 2.39[25]. Where P is the precision defined in 2.37 and R (recall) is defined in 2.38. The measurement is a way to take account both Precision and Recall. The balanced *F-measure* is called the *F_1-measure* and is defined in 2.40. It weighs the impact of precision and recall the same.

$$P = \frac{TP}{TP + FP} \tag{2.37}$$

$$R = \frac{TP}{TP + FN} \tag{2.38}$$

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R)}{\beta^2 \cdot P + R} \tag{2.39}$$

$$F_{\beta=1} = \frac{2 \cdot P \cdot R}{P + R} \tag{2.40}$$

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

2.6 Chapter Summary

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

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

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

The clustering method for this thesis will be to use EWKM algorithm on the given dataset. The first reason for choosing the algorithm is that it is a soft-subspace clustering algorithm. Secondly, it takes into account th information gain in the cost function. To evaluate the performance, the algorithm will be compared by the traditional *K-means* algorithm. The evaluation will compare the average silhouette coefficient for a chosen *K-parameter* on the two algorithms results. The results will also be evaluated by an external criteria of F_1 Score basing the ground truth on the genre feature which is a given label for all songs in the dataset.

A caveat with choosing the method is its use of a numerical distance measure. As such, only numerical features of the dataset can be used.

Table 2.1: Properties of algorithms discussed in this chapter [4, 20]

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

[a] Not all features have independent weights.
[b] Has been modified to allow mixed data.
[c] Only Feature weighting.
[d] Breaks on zero variance.
[*] Dimensionality disregarded in complexity notation

Chapter 3

Method

Chapter 4

Method

4.1 Dataset

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

The general metadata include attributes that identify the song without resulting to audio analysis. They include: Album, Artist, Title, Year of Origin. The type also includes BPM as well as .... The general metadata consists of both numerical and categorical data — features stored as strings can be converted to categorical data. Furthermore, certain features are stored dynamically i.e. a song can have multiple genres or no genre at all.

The second feature type is a 160-dimensional vector embedding — with numerical values ranging through -1 and 1 — that characterizes the song through audio analysis. The embedding is generated through a neural network of auto-encoders, trained on the Mel-frequency cepstrum coefficients(MFCC)[28]. The feature type is anonymous to humans, i.e. what exactly dimension 5 represents is unknown.

4.1.1 MFCC

To understand MFCC we need to first understand how humans perceive sound and how speech is created. Humans are more sensitive to frequency differences at lower frequencies compared to higher frequencies. The Mel-scale takes the differences in sensitivity into account by creating a logarithmic scale. Human speech is generated by the shape of the vocal tract. The shape determines what sound is created. A power spectrum describes how much power is in the frequency component of a signal. The envelope of the power spectrum can be used to represent the shape of the vocal tract of the given audio. An MFCC represents the envelope [28]. With the shape of the tract represented, generated phonemes can be determined of the vocals of the audio.

4.2 Pre-Processing of dataset

The choice of numerical clustering methods restricts the dataset for clustering to only include numerical data. As such, only the numerical features of the general metadata could be used in addition to the audio embedding. The decision was made to restrict the dataset in the beginning to only the audio-embedding excluding all of the general features as there was a hypothesis that the audio embedding would be result in good enough clusters.

The audio-embedding features were extracted from the original dataset and used as the input for the clustering algorithms. Before sending in the data, it was standardized using the z-score method.

4.3 Algorithm Implementation

4.4 EWKM Implementation

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

The implementation differs slightly in how λ is calculated from the algorithm shown in ... It includes additional normalization steps. Eq .. shows how λ is

updated in the given implementation.

Due to the thesis author's familiarity with Python and Pandas DataFrame, the code was re-wrapped for Python using numpy-C-API — allowing for the dataset to be sent in to the algorithm as a numpy array. The C-code was tweaked to not be dependent on $R's unif_r and() function$. The wrapping can be found in the attachments....

4.5 K-means Implementaion

K-means is a widely available algorithm for Python. The thesis used the PyCluster implementaion of $K - means$. The implementaion was created in $C++$ and wrapped for Python.

4.6 Initialization and Parameter Tuning

γ is an additional parameter of EWKM. A too high γ results in a total entropy larger than the total distance. A γ set too low resulted in slow convergence and a high cost function. The parameter was tuned by iteratively running the clustering method on multiple γ's and choosing the γ which results in the lowest cost function.

Chapter 5

Results

Chapter 6

Discussion

Chapter 7

Conclusions

Bibliography

[1] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

[15] 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.

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

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

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

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

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

[21] 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.

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

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

[24] 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.

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

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

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

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

Appendix A

Something Extra