Exploratory Cluster Analysis with Jointly Optimized Feature Transformations

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

eingereicht von

Dominik Mautz

München, den 3. März 2021

Erstgutachter: Prof. Dr. Christian Böhm
Zweitgutachter: Prof. Dr. Ingo Scholtes
Drittgutachter: Prof. Elena Baralis, PhD

Tag der mündlichen Prüfung: 10. Februar 2022

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12. Juli 2011, §8 Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Maisach, den 8. Mai 2022

.....

Dominik Mautz

Abstract

Cluster analysis is one of the fundamental tasks in exploratory data mining. Data scientists use cluster analysis to discover previously unknown structures within a data set. Over the years, a vast variety of specialized clustering methods have been developed to deal with the high diversity of data domains, which can be found in industry and science.

Clustering methods are the central topic of this cumulative dissertation. We discuss our contribution of five peer-reviewed and published clustering techniques aiming at different applications. The common ground of all proposed methods is the joint optimization of clustering objectives with linear and nonlinear feature transformations. The general idea behind the joint optimization is that a feature transformation can provide a clustering algorithm with a refined representation of the data. In return, the clustering target function provides information on how to improve and refine the data transformation. This joint optimization approach contrasts with the 'classical approach', in which the data scientist first transforms the data and then applies the cluster analysis in a separate step.

With extensive experiments using synthetic and real-world data sets, we empirically show that the simultaneous optimization of both objectives has an advantage over separate optimization steps. With the data transformation, we pursue a twofold goal. First, it mitigates the influence of irrelevant, structure-less features on the clustering objective. Second, the transformation often results in a substantial dimensional reduction. This reduction, in turn, promises an easier (visual) inspection of the representation itself, as well as, the structures found by the clustering method.

The first contribution is the SubKmeans algorithm. It extends the classic and well-known Lloyd's *k*-means method in a novel and elegant way to incorporate a linear feature transformation. SubKmeans aims to find a *k*-means-style clustering partition and transform the clusters linearly into a common, arbitrarily oriented subspace which is optimal for the cluster structures. Our method is able to pursue

these two goals simultaneously. The dimensionality of this subspace is found automatically, and therefore the algorithm does not have any additional parameters compared to *k*-means. At the same time, this subspace helps to mitigate the curse of dimensionality.

The second presented clustering method tackles the phenomenon that data in high-dimensional spaces can often be meaningfully clustered in multiple ways. For instance, objects could be grouped either by their shape, their material, or their color. Each of these groupings provides a unique view of the structures contained in the data, which is not covered by the other clusterings. With the Nr-Kmeans clustering algorithm, we follow the approach that different, non-redundant, *k*-means-like clusterings may exist in different, arbitrarily oriented subspaces of the high-dimensional input space. We assume that these subspaces (and optionally an additional subspace without any cluster structure) are orthogonal to each other. We empirically show that the orthogonality constraint induces the desired non-redundancy.

Although Nr-Kmeans can find multiple non-redundant clusterings within a data set, a user must specify the number of clusters within each subspace as an input parameter. With Nr-DipMeans, we propose a method that automatically finds the number of clusters. The only remaining parameter is the number of expected subspaces. Nr-DipMeans harnesses Hartigan's dip test to identify the number of clusters for each subspace under the assumption that clusters follow a unimodal distribution.

Deep clustering—the idea of combining deep learning techniques with clustering algorithms—has attracted much interest in recent years. With DeepECT, we contribute a novel method to this research area. DeepECT is the first method to combine a divisive, hierarchical clustering objective with the nonlinear transformation of an auto-encoding neural network (autoencoder). Previous deep clustering methods have only provided the user with a flat-clustering result where the user must specify the number of clusters. However, finding a suitable value for the number of clusters for these methods can be more complicated than in classical clustering settings. The nonlinear embedding adapts almost perfectly to the chosen number of clusters and destroys structural information not captured by the clusters. DeepECT utilizes a specialized optimization procedure that circumvents this problem. The level of detail to be analyzed (i.e., the selected number of clusters) can be chosen afterward and separately for each sub-tree.

With the final contribution, we demonstrate that the non-redundant clustering

objective of Nr-Kmeans can also be incorporated with deep clustering methods. ENRC is the first non-redundant clustering algorithm that utilizes the nonlinear transformation of an autoencoder. Like Nr-Kmeans, it can find multiple highly-non-redundant clusterings in subspaces of different dimensionality within a data set. In contrast to Nr-Kmeans, which hard-assigns each dimension of the rotated space to one clustering, we use in ENRC a differentiable soft assignment within the autoencoder's embedded space. The autoencoder's nonlinear transformation allows ENRC to cluster complex data sets like image data without the need for any explicit feature engineering.

Zusammenfassung

Die Clusteranalyse ist eine der grundlegenden Aufgaben des explorativen Data Minings. Data Scientists verwenden die Clusteranalyse, um bisher unbekannte Strukturen innerhalb eines Datensatzes zu entdecken. Über die Jahre wurden eine Vielzahl spezialisierter Clustering-Methoden entwickelt, welche für die vielfältigen Einsatzbereiche in Industrie und Wissenschaft notwendig sind.

Clusterverfahren sind auch das zentrale Thema dieser kumulativen Dissertation. Wir diskutieren fünf von uns veröffentlichte Verfahren, mit welchen wir zu diesem Forschungsfeld beigetragen. Die gemeinsame Grundlage aller vorgeschlagenen Methoden ist die eng verbundene Optimierung einer Clustering-Zielfunktion mit einer linearen bzw. nichtlinearen Merkmalstransformation. Die grundlegende Idee hinter dieser gemeinsamen Optimierung ist, dass die Merkmalstransformation der Clustering-Zielfunktion eine verfeinerte Repräsentation der Strukturen in den Daten liefert. Im Gegenzug liefert die Clustering-Zielfunktion Informationen darüber, wie die Datentransformation verbessert und verfeinert werden kann. Dieser Ansatz der gemeinsamen Optimierung steht im Kontrast zur "klassischen Herangehensweise", bei dem der Data Scientist zunächst die Daten transformiert und dann in einem separaten Schritt die Clusteranalyse anwendet.

Durch umfangreiche Experimente zeigen wir mit Hilfe von synthetischen und realen Datensätzen empirisch, dass die gleichzeitige Optimierung beider Ziele einen Vorteil gegenüber der getrennten Optimierung hat. Mit der Datentransformation werden dabei zwei Absichten gleichzeitig verfolgt. Erstens verringert diese den Einfluss irrelevanter, strukturloser Merkmale auf die Clustering-Zielfunktion. Zweitens führt die Transformation häufig zu einer substanziellen Dimensionsreduktion. Diese wiederum verspricht eine leichtere (visuelle) Überprüfung der Transformation selbst, als auch der von der Clustring-Methode gefundenen Strukturen.

Der erste wissenschaftliche Beitrag ist der SubKmeans-Algorithmus. Das Verfahren erweitert die klassische und sehr bekannte Lloyd's *k*-means-Methode auf eine neuartige und elegante Art und Weise, um eine lineare Merkmalstransformation

zu integrieren. SubKmeans zielt darauf ab, eine Clustering-Partition im Stil von *k*-means zu finden sowie die Cluster linear in einen gemeinsamen, beliebig orientierten Unterraum zu transformieren, welcher für die Clusterstrukturen optimal ist. Unsere Methode ist in der Lage, diese beiden Ziele gleichzeitig zu verfolgen. Die Dimensionalität des Unterraums wird automatisch gefunden, weshalb der Algorithmus über keine zusätzlichen Parameter gegenüber *k*-means verfügt. Gleichzeitig trägt dieser Unterraum dazu bei, den Fluch der Dimensionalität abzuschwächen.

Die zweite vorgestellte Clustering-Methode befasst sich mit dem Phänomen, dass Daten in hochdimensionalen Räumen oft auf mehr als eine Weise sinnvoll geclustert werden können. Beispielsweise könnten Objekte entweder nach ihrer Form, ihrem Material oder ihrer Farbe gruppiert werden. Jede dieser Gruppierungen stellt eine einzigartige Sicht auf die in den Daten enthaltenen Strukturen dar, die von den anderen Gruppierungen nicht abgedeckt wird. Mit dem Nr-Kmeans-Clustering-Algorithmus verfolgen wir den Ansatz, dass in verschiedenen, beliebig orientierten Unterräumen des hochdimensionalen Ausgangsraums unterschiedliche, nicht redundante, *k*-means-ähnliche Clusterings existieren können. Wir gehen davon aus, dass diese Teilräume (und optional ein weiterer Teilraum ohne jegliche Clusterstruktur) orthogonal zueinanderstehen. Wir zeigen empirisch, dass die Orthogonalitätsbeschränkung die gewünschte Nicht-Redundanz induziert.

Obgleich Nr-Kmeans mehrere nicht redundante Cluster innerhalb eines Datensatzes finden kann, hat ein Benutzer die Bürde, dass er die Anzahl der Cluster innerhalb jedes Unterraumes als Eingabeparameter angeben muss. Mit Nr-DipMeans schlagen wir eine Methode vor, welche in der Lage ist, die Anzahl der Cluster automatisch zu finden. Der einzige verbleibende Parameter ist die Anzahl der erwarteten Unterräume. Nr-DipMeans macht sich den Dip-Test von Hartigan zunutze, um die Anzahl der Cluster für jeden Unterraum unter der Annahme zu identifizieren, dass Cluster einer unimodalen Verteilung unterliegen.

Deep Clustering - die Idee, Techniken des Deep Learings mit Clustering-Algorithmen zu kombinieren - hat in den letzten Jahren großes Interesse gefunden. Mit DeepECT tragen wir eine neuartige Methode zu dieser Forschungsrichtung bei. DeepECT ist die erste Methode, die ein hierarchisches Top-Down-Clustering-Ziel mit der nichtlinearen Transformation eines autocodierenden neuronalen Netzes (Autoencoder) kombiniert. Bisherige Methoden des Deep-Clustering haben dem Benutzer nur ein flaches Clustering-Ergebnis mit einer vorgegebenen Anzahl von Clustern geliefert. Das Finden eines passenden Wertes für die Anzahl von Clustern

für diese Art von Methoden kann jedoch komplizierter sein als bei klassischen Clustering-Verfahren. Die nichtlineare Transformation passt sich an die gewählte Clusterzahl nahezu perfekt an und zerstört dabei Strukturinformationen, welche nicht durch die Cluster erfasst werden. DeepECT verwendet ein spezialisiertes Optimierungsverfahren, das dieses Problem umgeht. Der Detailgrad der Analyse (d.h. die gewählte Cluster-Anzahl) kann nachträglich und für jeden Teilbaum separat gewählt werden.

Mit dem letzten Clustering Algorithmus zeigen wir, dass das nichtredundante Clusteringziel von Nr-Kmeans auch mit Deep-Clustering Algorithmen umgesetzt werden kann. ENRC ist der erste nicht-redundante Clustering-Algorithmus, welcher die nichtlineare Transformation eines Autoencoders nutzt. Wie Nr-Kmeans kann er in einem Datensatz mehrere hochwertige, nicht-redundante Clustering-Ergebnisse in Unterräumen unterschiedlicher Dimensionalität finden. Im Gegensatz zum Nr-Kmeans Verfahren, welches jede Dimension des eingebetteten Raums einem Clustering fest zuordnet, verwenden wir in ENRC eine differenzierbare weiche Zuordnung. Die nichtlineare Transformation des Autoencoders ermöglicht es ENRC, komplexe Datensätze wie z. B. Bilddaten zu gruppieren, ohne dass ein explizites Feature-Engineering erforderlich ist.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, ohne deren Hilfe die vorliegende Doktorarbeit nicht möglich gewesen wäre! Insbesondere möchte ich mich herzlich bedanken...

... bei meinem Doktorvater Prof. Dr. Christian Böhm

... bei meinen Gutachtern, Prof. Dr. Ingo Scholtes und Prof. Elena Baralis, PhD

... bei Prof. Dr. Claudia Plant

... bei Lukas Miklautz, Wei Ye, Collin Leiber, Can Altinigneli, Linfei Zhou, Walid Durani, Lena Bauer, und den anderen Kollegen in München und Wien

... bei meinen Eltern und meiner Familie

... und natürlich bei meiner Frau Eva

So Long, and Thanks for All the Fish
– Douglas Adams

Contents

1 Introduction

The technological advances of the last decades have made it increasingly easier to accumulate data with unprecedented speed and level of detail. Consequently, the amount of stored data has doubled every two years since the beginning of the 1980s [GR12]. The collection and storing of data is nowadays an everyday experience. Some even go so far as to call data the oil of the 21st century [Too14]. This era of *big data* is usually characterized by the three *v*'s standing for *volume*, *variety*, and *velocity* [GH15]. But this characterization has also been extended by other properties [KM16], such as: *exhaustivity*, *resolution, indexicality, relationality, extensionality,* and *scalability*. Of course, the collection and storage of data is not an end in itself. Scientists, Companies, and Governments collect the data with the intention to verify assumptions and extract previously unknown patterns that provide information about underlying processes. The knowledge gathered this way grants new insights that guide future decision-making processes or even allows automating these processes entirely.

Many different methods and algorithms have been proposed to extract information from and make predictions based on data. These techniques are researched and applied in data mining and the closely related fields of data science, artificial intelligence, statistics, and machine learning. The influence of these fields in today's society is increasingly prominent, and they make regular appearances in the headlines of mainstream media, e.g., in [Tau20; Wax13; Wei13; Bro20; Ros20]. The Harvard Business Review even goes as far as declaring a data scientist's job as 'The Sexiest Job of the 21st Century' [DP12]. The boundaries between the fields mentioned above are vague, and different conflicting definitions exist, e.g., in [Val18b; Val18a; Lip15; Cas16; Rap17; May16; Hei17].

An influential view on data mining is described by Usama Fayyad in [FPS96] that defines data mining as one step in the process of *knowledge discovery in databases* (KDD). The five-step pipeline starts with a raw and unprocessed database. In the first step, we select objects and variables from the whole database, which promise to

correspond to our research goal. For this selection, we should also consider criteria such as availability and quality. Next, we need to preprocess the data. This includes removing noise, handling missing values, or normalizing and standardizing the data representation. In the third step, we apply dimensional reduction and feature transformation methods to reduce the number of effective variables or gain a task-specific representation. The fourth step is the central Data Mining task. We use various algorithms to extract patterns and information from the transformed data. The concrete algorithms to be used depend heavily on our goal, and selecting suitable methods is a non-trivial problem. Finally, we interpret and evaluate the found patterns, which broadens our understanding and increases our knowledge of the mechanisms and processes underlying the data. Last but not least, we can also use the patterns to re-iterate the pipeline partially or entirely to refine and distill the found information.

Depending on the concrete research goal, we have to choose different types of Data Mining algorithms. For instance, we might want to summarize the data, detect unusual or uncommon instances (outliers), classify objects based on their properties, or find discrete groups of clusters. Each of these research targets requires different types of methods. In this dissertation, we focus on unsupervised clustering algorithms. The general goal of cluster analysis is to partition the instances of a data set in such a way that objects within the same group are more similar to each other than to objects of a different group. These algorithms are unsupervised, meaning that they require no further information about structures and patterns within the given data set. Instead, it can be used to find these structures.

1.1 Research Goal

This cumulative dissertation contributes five publications proposing algorithms for cluster analysis. The common denominator and goal of all proposed algorithms is the objective to fuse the third and fourth step of the KDD process into a single technique that jointly optimizes an unsupervised feature transformation and a clustering objective. Thereby, the unsupervised feature transformation can provide the clustering objectives with a refined representation of the data. In turn, the clustering objective provides information on how this representation can be improved. The empirical evaluation of these algorithms on various data sets shows that jointly optimizing both objectives is advantageous over the combination of algorithms that

perform those two steps separately.

The feature transformations integrated into the algorithms are either linear or nonlinear. A linear transformation has the advantage that, in general, it is regarded as easier to interpret. The interpretability increases even further when the linear transformation is further restricted to be an orthogonal matrix, as used in three of the contributed algorithms. An orthogonal matrix allows its weights to be interpreted directly as feature importance. Further, all proposed linear methods also perform a projection onto a subspace of the original space. From the perspective of the clustering objective, this is also a type of reduction of dimensionality. This provides for easier subsequent visualization and supports the interpretation of the found patterns.

However, a linear transformation may be too restricted in some cases. In these scenarios, more powerful nonlinear transformations can help. The general concept of these transformations is to extract more abstract, higher-level structures within the provided low-level features. Since the number of higher-level features is usually (but not necessarily) far lower, it also is often used as a dimensionality reduction method. Further, it even has the chance to find structures in the low-level features that would have never been extracted by hand-crafted features. A downside of nonlinear transformation is its interpretability, which is much harder than with a linear model. In the literature, such transformations are called black-box methods—in contrast to easy to interpret white-box methods, such as linear transformations. Nevertheless, in recent years different methods have been proposed to analyze neural-network-based nonlinear transformations, e.g., in [MSM18; Ola+20; Car+20].

1.2 Thesis Structure

The remainder of this thesis is ordered as follows. Chapter 2 describes fundamental data mining methods directly related to the contributions of this dissertation. Chapter 3 provides general information about the contributed publications and other publications written at the time of the doctorate. Chapter 4 discusses the common and distinguishing properties among the contributions and the preliminary methods described in Chapter 2. Chapter 5 offers potential future research directions based on the contributions and concluding remarks. The appendix contains the five publications, as well as the attribution of work for all co-authors.

2 Preliminaries

This thesis's contributions rest on two central areas within data science: dimensional transformation and cluster analysis. This chapter discusses several fundamental techniques directly related to the methods and algorithms we contribute through this thesis. We introduce and discuss each technique only to the depth necessary to understand the connection between them and the thesis contributions. Interested readers can find more comprehensive discussions in more subject-specific literature referenced in the respective sections.

2.1 Centroid-based Cluster Analysis

In general, cluster analysis is an unsupervised learning task that aims to find structures within a set of data objects.

Clustering is an ill-posed and only vaguely-defined problem. In its most general form, one could say that clustering methods aim to group instances of a data set, such that objects within a group are more similar to each other than to data points of other groups. More formally defined, the optimization goal can be described as 'maximizing the intraclass similarity and minimizing the interclass similarity' [HPK11, p.20]. However, it can be shown that even a simple set of three axiomatic and desirable properties for a partitioning technique is impossible to fulfill [Kle03]. Nevertheless, this result should not be seen as a coffin nail for the clustering endeavor. Instead, different algorithms yield different relaxations and trade-offs of such axiomatic properties. Further, clustering algorithms define their objective through explicit and implicit assumptions about patterns and structures that they aim to unveil. Selecting a suitable clustering algorithm for a given data set is part of a data scientist's job.

Cluster algorithms with a hard and exclusive object-to-cluster assignment are also called partition-based clustering methods. Determining the optimal partition for a given loss function and set of objects is usually NP-hard. Therefore, clustering al-

gorithms only aim to find a sufficient local optimum. They utilize different starting conditions to reach different local optima of their respective clustering objective. All contributed algorithms in this thesis are partition-based clustering algorithms. Further, they have in common that they represent clusters by representatives, usually termed centroids. Often a centroid can be seen as a model object representing the typical and specific properties of its cluster. However, this is only valid if the cluster contains objects that are similar enough to each other such that the representative is able to express their common properties. Because each cluster has only a single centroid, an implicit assumption of these methods is that clusters of objects tend to have a spherical or at least a convex shape. However, this is a soft requirement. All methods can cope with a violation of this property up to a certain degree.

Numerous different clustering methods have been developed and applied to different areas in science and industry. Therefore, the following books and surveys are only a small selection for more comprehensive discussions: [AR16; HPK11; Ber06; GMW07; XT15; XW05; KKZ09].

2.1.1 *k*-means Clustering Algorithm

Algorithm 1: Lloyd's *k*-means algorithm

Input: data set $\mathcal{D}$; number of clusters k
Output: clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$;
`// Random initialization:`
$\forall i \in [1, k] : \boldsymbol{\mu}_i \leftarrow$ random data point of $\mathcal{D}$
repeat
  `// Assignment step`
  $\mathcal{C}_j \leftarrow \emptyset$
  $\forall \mathbf{x} \in \mathcal{D}$:
    $j \leftarrow \arg\min_{i \in [1,k]} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$
    $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{\mathbf{x}\}$
  `// Update step`
  $\forall i \in [1, k]$:
    $\boldsymbol{\mu}_i \leftarrow \frac{1}{|\mathcal{C}_i|} \sum_{\mathbf{x} \in \mathcal{C}_i} \mathbf{x}$
until *convergence*;

The k-means clustering algorithm is one of the most influential data mining algorithms [Wu+08]. A key element of its success is the variability and simplicity of this clustering approach, which allows it to be easily implemented and applied in various scenarios. The central idea of k-means is that we want to find a set of k clusters $\mathcal{C}_i$, with $1 <= i <= k$, which partitions the data such that the sum of squared errors is minimized,

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2, \tag{2.1}$$

where $\mathbf{x}$ represents a data point, $\boldsymbol{\mu}_i$ represents the cluster representative of cluster i, and $\|\cdot\|_2$ represents the Euclidean norm.

This loss can be optimized in different ways. The most common way to optimize this loss function is Lloyd's algorithm [Llo82] shown in Algorithm 1. It alternates between assigning each data point to the nearest centroid and updating the centroid locations based on the assigned data points. Both steps are repeated until convergence, i.e., the assignment does not change anymore. Figure 2.1 shows an example of the k-means algorithm for a simple data set. One can also apply other stopping criteria; for instance, the number of iterations reaches some specified value or difference in the loss of two consecutive iterations falls below some threshold. The solution found at convergence is only a local optimum because, as with most clustering objectives, finding the global optimum for the loss function shown above is NP-hard, even for $k = 2$ [GJW82]. The optimization method itself is a particular instance of Newton's method [BB95], and the convergence is at least quadratic [BBV04, p. 488]. We can find different local optima by running the algorithm with different initialization for the centroids. The center initialization strategy using random data points shown in Algorithm 1 is only one possibility. Nowadays, a widespread way is to use the k-means++ method[AV07], which is $O(\log k)$-competitive to the global optimum.

When we use k-means, we have to select a value for the number of clusters k. Different methods have been proposed to guide this selection process. The silhouette coefficient [Rou87] and the elbow method [KS96] are both techniques that aim to help the researcher to determine the *right* number of clusters by running the algorithm multiple times and evaluating the clustering outcome with different metrics. The silhouette coefficient determines the cluster quality by comparing an object's similarity to the objects of its cluster compared to objects from the

other clusters. For the elbow method[1], we create a line-diagram with the number of clusters on the x-axis and the percentage of explained variance on the y-axis. When doing this, we can often find a characteristic change in the slope of the line—the elbow. X-means [PM+00], Dip-means [KL12], and G-means [HE04] are three methods that aim to automatically determine the number of clusters by applying statistical measures and hypothesis testing.

k-means with the Euclidean norm is the standard version of this algorithm. However, one could also use other distance measures or norms, which has implications on how the cluster centroids $\boldsymbol{\mu}_i$ are calculated. For instance, in *k*-medians the $L1$-norm is used. As a consequence, the centroid is calculated using the median instead of the mean. The algorithm *k*-medoids exclusively selects data points as centroids instead of calculating them based on the assigned data objects. This method can be used in situations where we want to use a similarity measure that is not based on a metric, and a proper way to calculate the centroids is undefined.

The extensions mentioned above are relevant for the contributions in this thesis, but the list is nowhere complete. The simplicity of k-means inspired many researchers to propose different extensions and modifications. A starting point for a more comprehensive view of these algorithms can be found in [Wu12]. Despite its age, *k*-means is actively researched with extensions proposed at top-tier conferences and in high-impact journals.

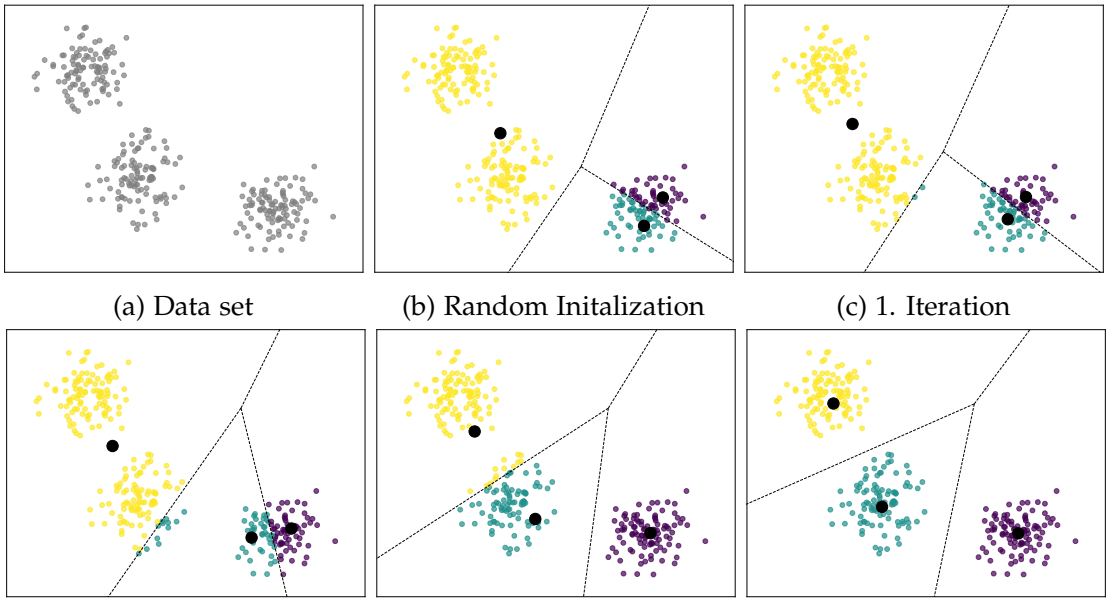
2.1.2 Fuzzy-c-Means

The Fuzzy-c-Means clustering algorithm [Dun73; Bez81] is a flat clustering algorithm closely related to *k*-means. The name-giving difference of Fuzzy-c-Means to *k*-means is its property to assign data points softly and with a certain degree to all clusters. This concept is represented by a positive weight $u_{j,i}$ indicating how much the j'th data object belongs to the i'th cluster:

$$\sum_{j=1}^{N} \sum_{i=1}^{k} u_{j,i}^{m} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|_2^2 . \tag{2.2}$$

The sum over all weights for a given data point sums up to one. The parameter m with $m \geq 1$ is called the fuzzifier and determines how crisp the cluster assignments

[1] Sometimes this method is also termed as 'finding the knee of a curve.'



(a) Data set (b) Random Initalization (c) 1. Iteration

(d) 4. Iteration (e) 7. Iteration (f) 8. Iteration

Figure 2.1: The plots show an example of the k-means method for a toy data set. Each plot shows one iteration of the loop in Algorithm 1. The lines represent decision boundaries for the assignment of data points to clusters. Figure (f) shows the converged algorithm.

are. In the limit $m = 1$, the objective is equal to k-means. In the limit m going to infinity, the algorithm assigns objects equally to all clusters.

We can optimize the cost function with an adapted version of Algorithm 1. The update equations full-filling the constraints on m can be found with a Lagrange multiplier. In the update step, we update the centroids with a weighted average of all data points, where the weights are the 'fuzzified' weights $u_{j,i}$:

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{N} u_{j,i}^m \mathbf{x}_j}{\sum_{j=1}^{N} u_{j,i}^m}. \tag{2.3}$$

In the assignment step, we update the soft-assignments $u_{j,i}$ for each data point j to each centroid i based on the distance:

$$u_{j,l} = \frac{1}{\sum_{i=1}^{k} \left(\frac{\left\| \mathbf{x}_j - \boldsymbol{\mu}_l \right\|_2}{\left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|_2} \right)^{\frac{2}{m-1}}}. \tag{2.4}$$

An illustration of the Fuzzy-c-Means algorithm is shown in Figure 2.2.

2.1.3 Gaussian Mixture Models

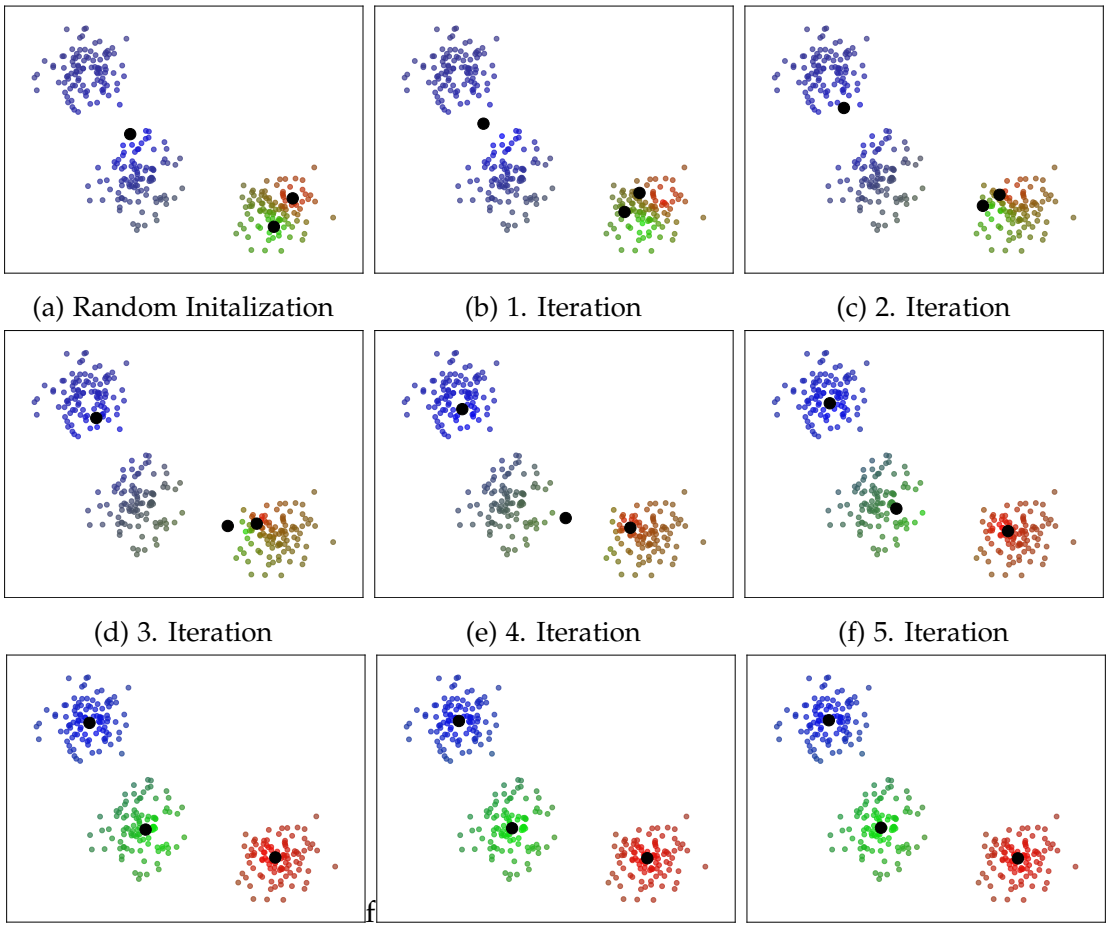
A Gaussian mixture model is a probabilistic model that combines several multivariate normal distributions—each representing one cluster—to a single probability:

$$p(\mathbf{x}) = \sum_{i}^{k} \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \Sigma_i), \tag{2.5}$$

where $\boldsymbol{\mu}_i$ is the mean vector and Σ_i is the covariance matrix of the i'th normal distribution. π_i is called the mixing component. It can be seen as the probability that a data point is created by the i'th Gaussian distribution.

Usually, we use the expectation-maximization algorithm, an iterative procedure related to Lloyd's k-means algorithm, to optimize the likelihood function of this probability distribution for a given data set (e.g., see [Bis06, p. 430ff]).

In the expectation-step (the assignment-step in k-means), we determine the soft-



(a) Random Initalization (b) 1. Iteration (c) 2. Iteration

(d) 3. Iteration (e) 4. Iteration (f) 5. Iteration

(g) 6. Iteration (h) 7. Iteration (i) 8. Iteration

Figure 2.2: The figure shows an illustration of the Fuzzy-c-Means algorithm for a toy data set. The fuzzifier was set to $m = 3$. We used the same initial centroids for k-means in Figure 2.1. We use the colors red, green, and blue to represents each of the clusters. The colors of the data points represent the soft-assignments to each cluster in the RGB color model. The last plot shows the converged algorithm.

assignment of a data point j to a cluster i:

$$\gamma_{j,i} = \frac{\pi_i \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_i, \Sigma_i)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \Sigma_l)}. \tag{2.6}$$

Within the Gaussian mixture model, this is equivalent to the posterior probability that the data point j was generated by the Gaussian distribution representing this cluster i.

In the maximization-step (the update-step in k-means), we optimize the distribution parameters of each cluster. The mean vector is the weighted empirical average over all data points:

$$\boldsymbol{\mu}_i = \frac{1}{\sum_{j=1}^{N} \gamma_{j,i}} \sum_{j=1}^{N} \gamma_{j,i} \mathbf{x}_j. \tag{2.7}$$

The covariance matrix is updated with the weighted empirical covariance matrix:

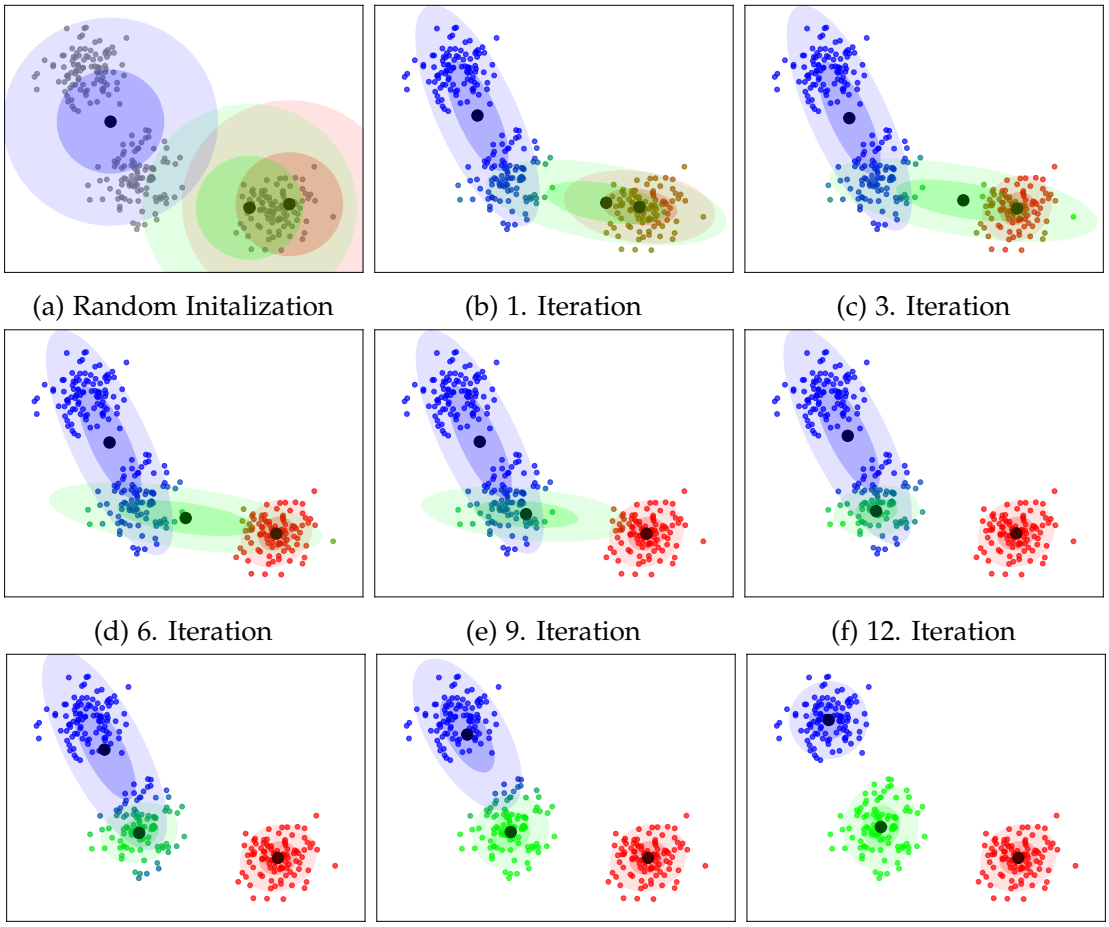
$$\Sigma_i = \frac{1}{\sum_{j=1}^{N} \gamma_{j,i}} \sum_{j=1}^{N} \gamma_{j,i} \left(\mathbf{x}_j - \boldsymbol{\mu}_i\right) \left(\mathbf{x}_j - \boldsymbol{\mu}_i\right)^{\mathsf{T}}. \tag{2.8}$$

We perform both steps until convergence. Figure 2.3 shows an example of the expectation-maximization optimization algorithm applied to a Gaussian mixture model for a simple data set.

The k-means cost function is the limit to a Gaussian Mixture Model, where the covariance matrix is isotropic $\sigma^2\mathbf{I}$ and where we let σ go to zero [KJ12].

2.1.4 Bisecting-k-means

The bisecting-k-means [SKK+00] is a simple hierarchical top-down clustering method that harnesses k-means. Initially, we consider the whole data set as a single cluster. Then, we recursively apply k-means with $k = 2$ to a selected cluster. In that way, we create a binary-tree-like hierarchy of clusters and sub-clusters. We can apply different strategies for the selection of the next cluster we want to split. The same is true for the stopping criteria. A typical selection strategy is always to split the largest cluster [SKK+00]. We stop the recursive process when a specific desired condition is met. Examples of such conditions could be the number of leaf nodes, the maximum number of data objects assigned to a leaf node, or when the sum of

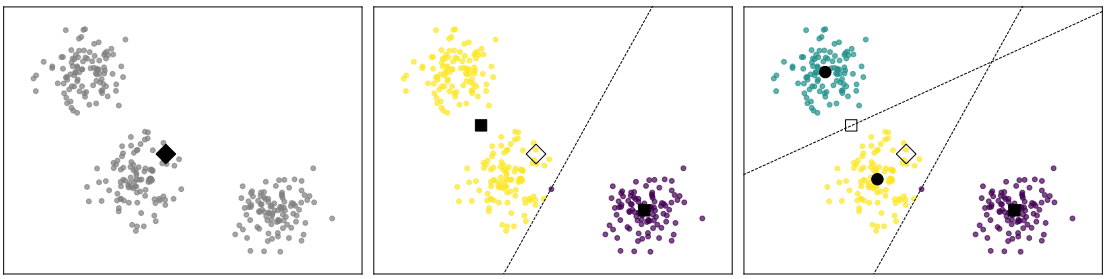


(a) Random Initalization (b) 1. Iteration (c) 3. Iteration

(d) 6. Iteration (e) 9. Iteration (f) 12. Iteration

(g) 15. Iteration (h) 18. Iteration (i) 21. Iteration

Figure 2.3: The plots show the iterations of a Gaussian mixture model on the toy data set. Figure (a) shows the random initialization with spherical covariance matrices. We use the colors red, green, and blue to represent the clusters. The colors of the data points represent the soft-assignments to each cluster in the RGB color model. The ellipses over each centroid show the standard deviation and twice the standard deviation of the underlying multivariate Gaussian distribution. Figure (i) shows the converged algorithm.



(a) Initialization (b) 1. Iteration (c) 2. Iteration

Figure 2.4: The figures show the application of Bisecting-k-means on the toy data set. Figure (a) shows the single root node with all data points assigned to it. Figure (b) and (c) show the iterations until three clusters, represented by the leaf nodes, are found. In each iteration, we apply k-means with $k = 2$ to the leaf node with the highest sum of squared distances between the points and the centroid. The centroids of inner nodes are hollow. Leaf node centroids are filled. The centroid of each level has its respective symbol. If we compare the decision boundaries with the ones of k-means in Figure 2.1, we can see slight differences, which are due to the divisive, top-down assignment approach of Bisecting-k-means.

squared distances of the leaf nodes is all below a specified threshold. In the extreme case, we stop when every data point resides in its own leaf node cluster. Figure 2.4 shows the application of Bisecting-k-means for a simple data set.

2.2 Feature Transformation

The related research areas of feature transformation, dimensionality reduction, feature extraction, and representation learning can roughly be described as aiming to transform the given presentation of a data set such that they are more suited for subsequent data science tasks. A tightly related field is feature selection, in which methods aim to select appropriate features and remove the unimportant ones from the data set representation.

An important application of this technique is to provide a researcher with useful low-dimensional visualizations of the structures and patterns hidden in the data's high-dimensionality. A low-dimensional presentation can help to understand the

processes underlying the data set. However, feature transformation and dimensionality reduction are not only useful for visualization but are also often able to improve the performance of unsupervised, semi-supervised, and supervised learning tasks.

Data transformation also touches on some of the aspects of the infamous *curse of dimensionality*. With increasing dimensionality, the notion of naive similarities between two data points gets increasingly vague. The underlying obstacle is that in each feature-dimension, small dissimilarities add up to influence the measure considerably. In addition, it can be assumed that the data set also contains features that do not add relevant information to a research task such as clustering; yet, these features also contribute to the similarity measure [KKZ09].

A common assumption for high-dimensional spaces is that the data is contained in a lower-dimensional manifold. This manifold can be locally approximated as a Euclidean space, while this does not hold for the global space. An informal yet illustrative example of a manifold is a metal spring. If we ignore the wire's width, we can treat the spring as a one-dimensional object that is embedded in our three-dimensional world. Consequently, a sheet of paper—ignoring the thickness—is a two-dimensional manifold.

Dimensional transformation methods aim at approximating such a manifold in a lower-dimensional space. The most common methods are unsupervised and do not use any target information. Nevertheless, there are also supervised and semi-supervised methods, exploiting target labels that are available for all, or at least some data objects. In the following, we introduce the unsupervised methods Principal Component Analysis as a linear dimensionality reduction method and autoencoder as a potentially nonlinear dimensionality reduction method.

t-distributed stochastic neighbor embedding (t-SNE) [MH08] and *Uniform manifold approximation and projection* (UMAP) [MH18] are two nonlinear dimensionality reduction methods, which have become popular in recent years for the visualization of high-dimensional data. Other essential methods are *Multidimensional Scaling*, *Independent Component Analysis*, and *Non-Negative Matrix Factorization*. A more general overview of different methods can be found in the following books and surveys: [Aar13], [CHU07], [LV07].

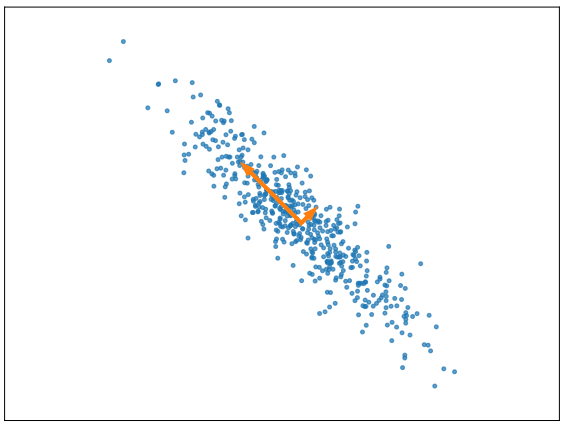


Figure 2.5: The diagram shows a two-dimensional example data set. The arrows are indicating the two principal, orthogonal directions of maximal variance found by PCA. The arrows' length is equal to the variance in the respective direction.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a commonly used linear dimensional reduction method. In PCA, we want to find a matrix $U \in \mathbb{R}^{d \times n}$, which represents a linear transformation of the original data space with dimensionality d onto a lower-dimensional subspace with dimensionality $m, m < d$, such that we can transform the data points onto the lower-dimensional space by $U^\mathsf{T}(\mathbf{x} - \boldsymbol{\mu})$, where $\boldsymbol{\mu}$ represents the data mean. Thereby, each column in U represents an orthogonal direction within the data set, in which the spread of the data (i.e., the variance) is maximal. Figure 2.5 illustrates this for an example data set. PCA does not compress or stretch the data. Therefore, we constrain the Principal components—the columns of U—to be orthonormal. This transformation relates to a rigid transformation (rotation and reflection, where only the former is desired and the latter can be ignored), followed by a projection where we remove the dimensions with low-spread.

These properties can be formalized into a cost function, where we want to maximize the spread of the data in the transformed space under the constraint that U is orthogonal:

$$\max \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \left\| U^\mathsf{T}(\mathbf{x} - \boldsymbol{\mu}) \right\|_2^2 \quad s.t. \quad U^\mathsf{T} U = \mathbf{I}_{n \times n}. \tag{2.9}$$

We can utilize the cyclic properties of the trace operations to rewrite this loss function:

$$\max \operatorname{Tr}\left(U^{\mathsf{T}}\left[\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\left(\mathbf{x}-\boldsymbol{\mu}\right)\left(\mathbf{x}-\boldsymbol{\mu}\right)^{\mathsf{T}}\right]U\right) s.t. \quad U^{\mathsf{T}}U = \mathbf{I}_{n\times n}. \tag{2.10}$$

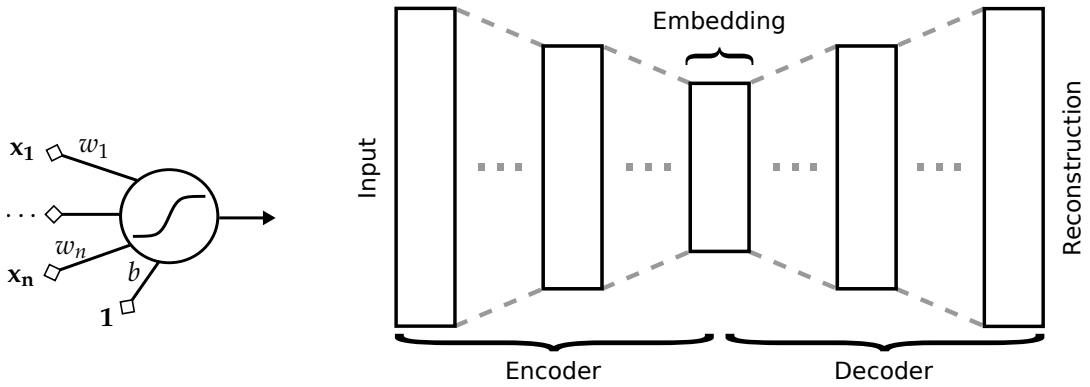
The term in the square brackets is the data covariance matrix. Since it is symmetric and positive definite, we can optimize this loss function using the eigendecomposition [NBS10]. Thereby, we put the eigenvectors corresponding to the largest eigenvalues as columns in U, until we have the desired number of dimensions in the projection. Thereby, the trace itself is the sum of these largest eigenvalues, as well as the sum of the variances in the directions of the eigenvectors [Bar12, p. 329ff].

The loss function above does not have a unique solution. All sets of orthonormal vectors spanning the same subspace are a solution to the optimization problem. However, only the eigenvectors (the Principal components) have the property that the m eigenvectors corresponding to the largest eigenvalues are a solution for dimensionality m. In other words, the eigenvector corresponding to the largest eigenvalue minimizes the loss for $m = 1$. Adding the eigenvector of the second largest eigenvalue, we get the solution for $m = 2$, and so forth. None of the other minimizers have this property.

Further, we should mention that above we implicitly assumed that the data covariance matrix is of full rank. If this is not the case, there is a strong direct linear correlation among the data set's dimensions. In [Bis06, p. 562], the interested reader can find another detailed explanation for the emergence of the eigenvector, which does not rely on the trace optimization trick. However, the above-shown way allows us to highlight the relationship between PCA and the contributions in this thesis.

An alternative view on PCA aims to minimize the mean squared error between a data point and its reconstruction. We will utilize this definition without deriving it explicitly in Section 2.2.3 to explain the connections to autoencoders. In [Bis06, p. 563f], the interested reader can find a detailed derivation from this point of view on PCA.

In the literature, we can also find extensions and modifications of PCA. A close relative to PCA is a preprocessing procedure called whitening, which transforms the data such that its data covariance matrix is the identity matrix. Other extensions



(a) A single neuron. (b) The general architecture of autoencoder.

Figure 2.6: A stylized version of a single neuron and the architecture of a typical bottlenecked autoencoder.

make use of Kernels in Kernel-PCA, the use of the $L1$-norm instead of the Euclidean norm in Robust PCA, or probabilistic interpretations as in Probabilistic PCA and Bayesian PCA. We can find a more comprehensive look at various extensions, for example, in [Bar12, p. 338ff] or [Bis06, p. 570ff]).

2.2.2 Autoencoder

As deep learning is a vast and very fast developing research area, the following short paragraphs to neural networks and autoencoders only provide a very brief introduction needed to link it with our contributions. We will not touch on more specific topics like convolution networks, recurrent networks, or specific types of network architectures. A recent comprehensive introduction to deep learning can be found in [GBC16].

The smallest building block of a neural network is a neuron. A neuron has a—potentially multi-dimensional—input and builds the weighted sum of this input using its weight vector w. Next, a constant bias b term is added to this weighted sum. Then, this linear combination is passed through an activation function $\mathrm{a}(\cdot)$, which often is a nonlinear function. If this output is large, neurons are usually considered to be active and inactive if the output is small. Figure 2.6a shows a stylized version of a neuron. Mathematically it can be expressed by the following

formula:

$$y = \mathrm{a}(w^{\mathsf{T}}\mathbf{x} + b), \tag{2.11}$$

where w is the weight vector and b the bias term of the neuron. Common activation functions $\mathrm{a}(\cdot)$ are:

- Identity function (for a linear activation),
- Sigmoid: $\frac{1}{\exp(-x)+1}$,
- Hyperbolic tangent: $1 - \frac{2}{\exp(2x)+1}$,
- Rectified linear unit: $\max(0, x)$.

A single neuron with a linear activation function is equivalent to the statistical linear regression model, and with the sigmoid activation, it is similar to the logistic regression model.

Usually, multiple neurons are arranged in layers so that all neurons of a layer use the previous layer's output as input and provide the next layer in turn with their output. This simple type is called a feed-forward or fully-connected neural network. The weights of such a layer can be represented as a matrix and the biases as a vector.

The network parameters and the input values are only one part. Further, we need some kind of target value the network should produce for a given input. For example, this target could be the label information in a classification task. Specific to the task, we also need some differentiable loss function. Common loss functions are the squared error or the cross-entropy loss. Finally, we have to initialize the network parameters for which researchers have introduced many different strategies for general and architecture-specific situations.

We can optimize the network parameters with respect to the loss function by a procedure called backpropagation. The underlying idea is to apply the chain rule to derive the gradient for the parameters of each layer. Modern frameworks for neural networks such as Pytorch [Pas+19] or Tensorflow [Mar+15] facilitate this for their users through automatic differentiation. The gradient is then used to update the network parameters. Besides pure stochastic gradient descent, other algorithms have been proposed for this task, such as *Averaged Stochastic Gradient Descent* [PJ92], *Adadelta* [Zei12], *RmsProp* [TH12], or *Adam* [KB15]. These algorithms

utilize different tricks like running averages and momentum over several update steps to accelerate the convergence speed.

An autoencoder (AE) is a particular type of neural network that aims to replicate the input at the output in an unsupervised manner. Let us assume that we have a neural network where the number of neurons in each layer is equal to the input dimensionality. This neural network could simply learn to copy the input faithfully to the output, i.e., it learns the identity function. Therefore it would not learn anything sensible. However, the goal of autoencoders is to learn a transformation of the data that extracts meaningful patterns and structural information. Therefore, we have to remove the network's ability—its capacity—just to copy the data and instead extract sensible structural information that abstracts from the input but also allows the network to reconstruct it. A common way for this is to define the autoencoder's architecture to have a bottleneck in the middle layer, where the number of neurons is much fewer than the number of input dimensions. An optimized bottlenecked autoencoder, able to replicate the input, cannot simply copy the input to the output. Instead, it has to extract sensible information and patterns within the input data, pass it through the bottleneck, and replicate the input based on this high-level information. The network part from the input up until the middle layer is called the encoder. The part from the middle to the output is called the decoder. The bottlenecked middle part is also called the embedding layer. Figure 2.6b shows a stylized version of these parts.

Autoencoders pose a vast area of research. For instance, a sub-field follows the direction of probabilistic generative models, where the embedded space is considered as an unobserved, latent variable. Other strategies aim to provide sparse representations or introduce regularization terms penalizing higher derivatives. More details regarding different kinds of autoencoders can be found, for example, in [Bal12], [GBC16, p. 493ff], or [Aar13].

2.2.3 Relationship between PCA and Autoencoder

There exists a well-known relationship between PCA and autoencoders.

Let us assume that we define an autoencoder with a single layer as an encoder and decoder, respectively. The decoder uses the same weight matrix as the encoder only transposed. This is a commonly used technique in autoencoders and is called tied weights. Further, we use a linear activation and fix the bias term to be zero. Also, we use the squared Euclidean distance as an error loss. Finally, for simplicity,

we assume that the data has been centered in a preprocessing step. The loss term we get in this way is:

$$\min \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \left\| \mathbf{x} - WW^{\mathsf{T}}\mathbf{x} \right\|_2^2, \tag{2.12}$$

where $W \in \mathbb{R}^{d \times m}$ is the weight matrix.

This loss function itself is equivalent to the minimum reconstruction error definition of PCA. The difference is that PCA requires the explicit constraint that the weight matrix is orthogonal $WW^{\mathsf{T}} = \mathbf{I}$. The autoencoder formulation does not require this constraint explicitly. Instead, during stochastic optimization of the loss function, the constraint $WW^{\mathsf{T}} \approx \mathbf{I}$ is approximated, but it is not guaranteed that it holds. Furthermore, the weights only approximate a basis that spans the same subspace as the Principal components of PCA. In general, there is no incentive for the autoencoder to learn the Principal components directly. However, this can be accomplished using specific optimization schedules [Oja89]. These learning schedules can be of use in the face of a large data set, where one wants to approximate *Principal Component Analysis*.

More details about the relationship between PCA and autoencoders can, for instance, be found in [Pla18].

3 General Overview of the Contributions

This cumulative dissertation aggregates five previously published contributions. The publications were accepted at prestigious venues. In the following, we provide general information about the contributed work and their respective publication outlet. The appendix contains all five publications with additional information, including detailed listings of co-author contributions.

3.1 Impact of Publication Outlets

All contributions are published at prestigious, high-impact, peer-reviewed venues, which are specialized in research in knowledge discovery, data mining, machine learning, and artificial intelligence. While the reputation of a venue in itself is no indicator of the importance and impact of a publication, it is the case that more esteemed conferences and journals are highly competitive. Therefore, there is a strong tendency for original and influential content. For readers from other fields, we should mention that in computer science, commonly conferences are regarded as more prestigious than journals [Ern06]. Most publications were published at the following conferences *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD), *IEEE International Conference on Data Mining* (ICDM), and *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence* (AAAI). These conferences take place once per year in different cities around the world. One contribution was published in the journal *ACM Transactions on Knowledge Discovery from Data* (TKDD).

Determining the importance and influence of venues is an intricate problem that we will not discuss here. Instead, we provide the results—from the time of writing—of two rankings by CORE and Google Scholar. The Computing Research

and Education Association of Australasia[1] (CORE)—an association of university departments of computer science in Australia and New Zealand—provides a ranking system for computer science conferences. Every publication of this dissertation was published at an outlet having the best ranking of A^*. Google Scholar provides a ranking system for several research fields, where both journals and conferences are ranked together. In the category *data mining & analysis*, KDD is ranked 1st, ICDM is ranked 5th, and TKDD is ranked 15th [Goo20b]. AAAI is ranked 4th in the category of *artificial intelligence* [Goo20a].

3.2 Contributed Publications

The list below shows the five contributed publications ordered by year. Publications $1-4$ were published at conferences. Publication 5 was invited to the TKDD journal. It is an extension of the second publication with at least 30% new content, discussing more aspects of the original publication and proposing a new algorithm.

1. Dominik Mautz, Wei Ye, Claudia Plant and Christian Böhm
   Towards an Optimal Subspace for K-Means
   Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017; [Mau+17]

2. Dominik Mautz, Wei Ye, Claudia Plant and Christian Böhm
   Discovering Non-Redundant K-means Clusterings in Optimal Subspaces
   Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018; [Mau+18]

3. Dominik Mautz, Claudia Plant and Christian Böhm
   Deep Embedded Cluster Tree
   Proceedings of the IEEE International Conference on Data Mining, ICDM, 2019; [MPB19]

[1] `http://www.core.edu.au`

4. Lukas Miklautz*, Dominik Mautz*, Muzaffer Can Altinigneli, Christian Böhm and Claudia Plant
   Deep Embedded Non-Redundant Clustering
   Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020; [Mik+20];
   Note: * marks authors with equal contribution

5. Dominik Mautz, Wei Ye, Claudia Plant and Christian Böhm
   Non-Redundant Subspace Clusterings with Nr-Kmeans and Nr-DipMeans
   ACM Transactions on Knowledge Discovery from Data, Vol 14, Nr. 5, 2020; [Mau+20]

3.3 Other Publications

The author also contributed to the following publications during his doctorate. However, these are not part of this dissertation. Publication iii. is an extended abstract of the second publication in the list above. Publication v. is an extended journal version of the fourth publication above with an additional 30% new content.

i. Dominik Mautz, Christian Böhm and Claudia Plant
   Subspace Clustering Ensembles through Tensor Decomposition
   Proceedings of the IEEE 16th International Conference on Data Mining Workshops, ICDM Workshops, 2016

ii. Wei Ye, Linfei Zhou, Dominik Mautz, Claudia Plant and Christian Böhm
   Learning from Labeled and Unlabeled Vertices in Networks
   Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017

iii. Dominik Mautz, Wei Ye, Claudia Plant and Christian Böhm
   Discovering Non-Redundant K-means Clusterings in Optimal Subspaces (Extended Abstract)
   Proceedings of the INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik

iv. Fiete Lüer, Dominik Mautz and Christian Böhm
   Anomaly Detection in Time Series using Generative Adversarial Networks (Abstract)
   Proceedings of the IEEE International Conference on Data Mining Workshops, ICDM Workshops, 2019

v. Dominik Mautz, Claudia Plant and Christian Böhm
DeepECT: The Deep Embedded Cluster Tree (Journal extension)
Springer Journal, Data Science and Engineering, 2020

vi. Wei Ye, Dominik Mautz, Christian Böhm, Ambuj Singh and Claudia Plant
Incorporating User's Preference into Attributed Graph Clustering
In IEEE Transactions on Knowledge and Data Engineering, 2021

vii. Lukas Miklautz, Lena Bauer, Dominik Mautz, Sebastian Tschiatschek, Christian Böhm and Claudia Plant
Details (Don't) Matter: Isolating Cluster Information in Deep Embedded Spaces
Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021

viii. Collin Leiber, Dominik Mautz, Claudia Plant and Christian Böhm
Automatic Parameter Selection for Non-Redundant Clustering
Proceedings of the 2022 SIAM International Conference on Data Mining, SDM 2022

Further, the following publication is, at the time of writing, under review:

ix. Walid Durani, Dominik Mautz, Claudia Plant, Christian Böhm
DBHD: Density-based clustering for highly varying density
Submitted to the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2022

4 Contributions in Detail

In this chapter, we provide more details for each contribution. We start by discussing central common properties and features shared among the methods. Further, we characterize and summarize the central ideas of each method. Finally, we provide links between them.

The appendix contains all five publications with additional information, including detailed listings of co-author contributions.

4.1 Common Properties and Features

All contributions are focused on unsupervised cluster analysis combined with unsupervised linear or nonlinear transformations. In this section, we describe the central properties and features of the contributed methods. Table 4.1 gives an overview of which clustering method has which feature or property.

4.1.1 Centroid-Based Cluster Representation

Many different clustering algorithms utilize centroids. Each cluster found is thereby characterized by such a representative. The coordinates of this centroid are often calculated and are, in general, not equal to an object of the cluster. Its feature values can be seen as prototypical for this cluster and are what one can expect on average to see within this cluster. Algorithms that use centroids to represent clusters have the implicit assumption that clusters have a spherical or at least convex shape.

4.1.2 Clustering Specific Subspaces

Classic clustering algorithms aim to provide the user with a result that considers all features of a data set. In contrast, most of our proposed algorithms provide a clustering together with an arbitrarily-oriented subspace of the original data space,

Table 4.1: The table shows the prominent features of each contributed clustering algorithm.

Features	SubKmeans	Nr-Kmeans	Nr-DipMeans	ENRC	DeepECT
centroid representation	✓	✓	✓	✓	✓
clustering specific subspaces	✓	✓	✓	✓	
non-redundant		✓	✓	✓	
flat clusterings	✓	✓	✓	✓	
hierarchical clustering					✓
learn number of clusters k			✓		(✓)
linear transformation	✓	✓	✓		
nonlinear transformation				✓	✓
noise space	✓	✓	✓		
visualization-friendly	✓	✓	✓	(✓)	

in which the clustering structure can be found. Subspace clustering algorithms are a related field, which provide for each cluster an axis-parallel subspace [KKZ09].

4.1.3 Non-redundant Clustering Results

Much like the goal of clustering itself, non-redundant clustering is only vaguely defined. Algorithms that are attributed to target non-redundancy share the idea that multiple, equally-meaningful partitions can be found within the data set such that these clusterings do not share structural information. Since non-redundant clustering is an ill-posed problem, and each algorithm uses a slightly different definition. However, from its general notion, it has parallels to the concept of statistical independence in probability theory. There are two different types of non-redundant clustering algorithms. The first type extracts the different clusterings sequentially, whereas the second type extracts all clusterings in parallel. All contributed non-redundant clustering algorithms belong to the latter type.

Further, non-redundant clustering is related to alternative or multiple clusterings. The distinguishing feature is that non-redundant clustering algorithms aim to find non-overlapping subspaces of the data space for each clustering, whereas alternative clustering aims to find the different clusterings in the same data space. Subspace clustering algorithms assign each cluster to its own (overlapping) subspace. Finally, algorithms in the field of multi-view clustering aim to find clusters that are observed from different views or sources (e.g., audio and video). A more comprehending discussion of these different fields can be found in [Mul+12; KKZ09; Kri+10; YW18].

Non-redundant cluster structures can be found in data domains with natural high-dimensional data sets, for instance, in medicine [Cui09] or genomics [TBB13]. Figure 4.2 shows an example of such non-redundant, equally valid groupings for an image data set.

4.1.4 The *noise* space

A noise space builds the complement to clustering-specific subspaces. We define it as an arbitrarily-oriented subspace complementing one or more *clustered* spaces. It does not contain any structural information important to any found clustering. We expect that such a subspace follows a unimodal distribution such that its dimensions cannot be used to distinguish between the different clusters in the *clustered* spaces. In our methods, we are not interested in the actual distribution of these

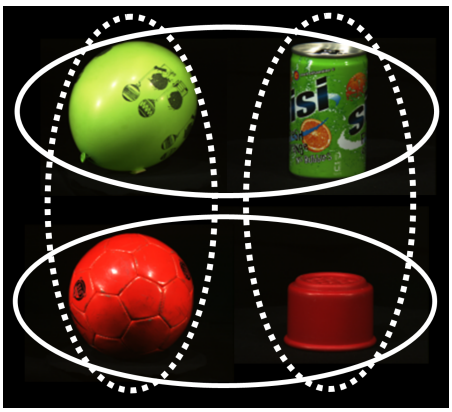


Figure 4.2: The image shows an illustrative example of non-redundant structures within a data set. The four objects can either be grouped by color or by shape. Both clusterings are sensible, non-redundant, and reveal different patterns.

noise dimensions. Therefore, we assume that this space can be described as a single cluster.

4.1.5 Linear and nonlinear Transformation

Mathematically a linear transformation or linear mapping is defined as a mapping f between two vector spaces over a field such that addition and scalar multiplications are preserved [Hef14, p. 176]:

$$f(a \cdot \vec{x} + b \cdot \vec{y}) = a \cdot f(\vec{x}) + b \cdot f(\vec{y}), \tag{4.1}$$

where a and b are some scalar values of a field and $\vec{x}$ and $\vec{y}$ are some vectors of a vector space over the same field. In the case of data mining and machine learning, this field is almost always the real numbers $\mathbb{R}$. A major advantage of linear transformations is the ease of interpretability, especially if we constrain the linear mapping further to build an orthonormal basis (i.e., a rotation or reflection). All contributed methods utilize this constraint, which allows us to directly interpret the absolute values of the mapping matrix as feature importance indicators for the directions in the rotated space. The example for a PCA in Figure 2.5 of Chapter 2 shows a linear transformation of a data set.

Nonlinear transformations are not restricted to the rules of linear transformations and therefore allow more complex manipulations of the data features, allowing it to find more abstract representations. A downside of these transformations is that they are, in general, harder to interpret. Further, the mapping is in danger of becoming arbitrary. Our contributions constrain the nonlinear transformations to be approximatively invertible, such that the original data point can be approximated from the representation.

4.1.6 Visualization-Friendly

We call a clustering result visualization-friendly if the clustering algorithm also provides the user with an accompanying subspace of the transformed space, in which the data manifests the clustering structure. Further, the dimensions may be ordered according to how well the clustering pattern is exhibited by them. Both the selection of a subspace and the ordering of its dimensions help a user to examine the found patterns visually.

4.1.7 Flat and Hierarchical Clustering Results

Flat clustering algorithms split a data set into groups, but the groups themselves do share any structural sub-cluster information. The majority of published clustering algorithms produce a flat clustering structure.

In contrast, hierarchical clustering algorithms yield a tree of clusters, where an inner node represents the join of its child nodes. In a complete tree, the root node cluster contains the whole data set, and each leaf node represents a cluster containing one single data point of the data set. The tree usually is build up either top-down or bottom-up and—depending on the use-case—may not build to its full height.

4.1.8 Learn Number of Clusters k

Almost all clustering algorithms require the user to provide and tune algorithm-specific parameters. A common parameter is the expected number of clusters a user wants to find. However, some algorithms also provide mechanisms to learn these parameters. Common approaches utilize statistical hypothesis testing. Others use model complexity measures like the *Akaike Information Criterion* [Aka98], the

Bayesian Information Criterion [Sch78], or the *Minimum Description Length Principle* [Ris78; GGR07].

4.2 Core Idea of each Method

In this section, we provide a basic overview of each proposed method and give a short explanation of the core ideas. The contributions are discussed in order of publication.

4.2.1 SubKmeans

The first and simplest proposed algorithm is SubKmeans [Mau+17]. This algorithm can be seen as an extension of the well-known Lloyd's *k*-means clustering method. It aims to solve the problem that not all available features within a data set contain clustering structures. Due to correlations and causal relations among the dimensions, these structures might not follow the coordinate-axis but may exist in an arbitrarily oriented, linear subspace (called *clustered* space) of the data space. SubKmeans aims to find a *k*-means-style clustering partition within such a subspace. The complementing space without any clustering structure (called *noise* space) is thereby assumed to be unimodal. The dimensionality of this subspace is found automatically, and therefore the algorithm does not have any additional parameters compared to *k*-means. An illustration of SubKmeans for an example data set and a comparison to the results of standalone *k*-means, as well as a PCA transformation followed by *k*-means is shown in Figure 4.3.

4.2.1.1 General Concept and the Clustering Objective Function

Based on the above-described assumptions, we define that the clustering structure can be captured by the *k*-means algorithm and fix this representation in the loss function. Further, we assume that this structure lies in an arbitrarily oriented *clustered* space. Therefore, we need a rotation matrix V that rotates the data space in such a way that the *clustered* space aligns with the coordinate axis. We describe the dimensionality of the *clustered* space as m. Note that this is not a parameter set by the user but is found automatically during optimization. As a consequence, the dimensionality of the *noise* space is $d - m$, where d is the dimensionality of the full space.

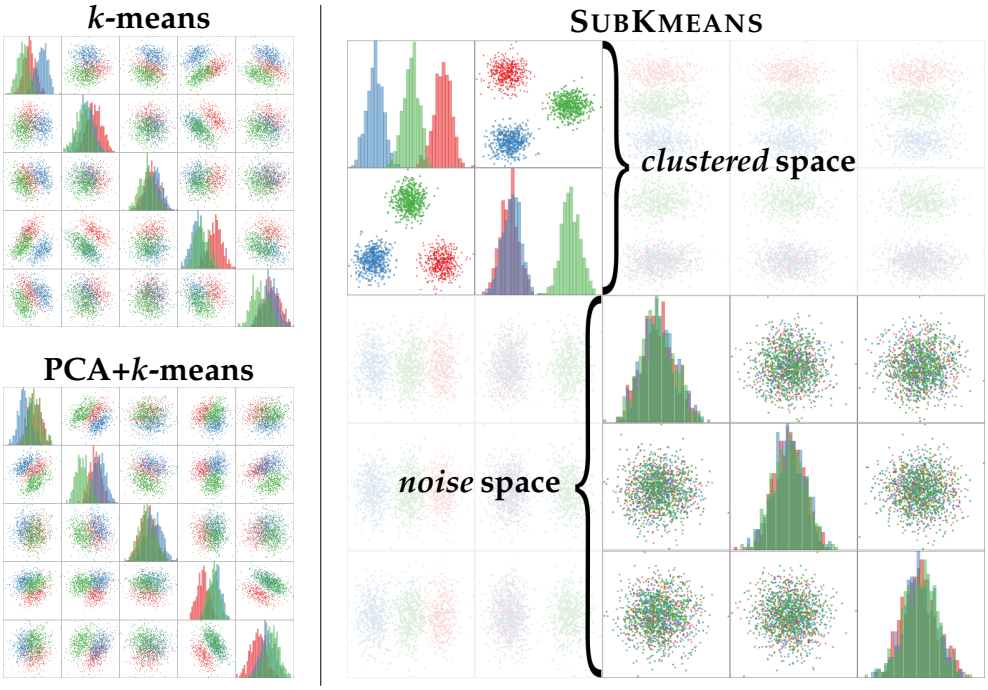


Figure 4.3: The three scatter plots show the result of k-means, PCA+k-means, and SUBKMEANS of an artificial data set with three ground truth clusters. In each plot, the found clusters are color-coded. The upper left plot shows the original data set together with the result of k-means, which recovers the ground truth for this simple data set, but the result is hard to interpret, and the visualization is not helpful to identify the actual cluster structure. The PCA transformation rotates the space, but the clustering structure is not clearly visible. The large plot on the right shows the result of SUBKMEANS. The method rotates the data set, finds the correct cluster partition, and automatically identifies the first two features in the transformed space as the *clustered* space. The remaining feature dimensions do not contain any clustering structures and are therefore identified as the *noise* space.

Once the data space is rotated, we can extract the *clustered* space using a simple projection matrix P_c, which we also call a masking matrix. We can do the same with and a *noise* space and a masking matrix P_n, respectively. Without loss of generality, we assume that V rotates the data in such a way that the first m dimensions represent the *clustered* space and the remaining $d - m$ dimensions the *noise* space. Given these assumptions, the matrices P_c and P_n can be defined as:

$$P_c = \begin{bmatrix} \mathbf{I}_m \\ \mathbf{0}_{d-m,m} \end{bmatrix}, \quad P_n = \begin{bmatrix} \mathbf{0}_{m,d-m} \\ \mathbf{I}_{d-m} \end{bmatrix}, \tag{4.2}$$

where $\mathbf{I}$ is the identity and $\mathbf{0}$ is the zero matrix. We can then project a data point $\mathbf{x}$ into the *clustered* space by $P_c^\mathsf{T} V^\mathsf{T} \mathbf{x}$ and onto the *noise* space in a similar manner using P_n.

The last remaining puzzle piece is the presentation of the data in the *noise* space. Our assumption above is that the data in this subspace follows a unimodal distribution without clustering structures. Further, we have the assumption of a k-means clustering structure in the *clustered* space. Therefore, a straightforward and natural representation of the data in the *noise* space is a single k-means-like cluster.

Once we put these concepts together, we get the following loss function:

$$\begin{aligned} \mathcal{J} = &\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} \left\| P_c^\mathsf{T} V^\mathsf{T} \mathbf{x} - P_c^\mathsf{T} V^\mathsf{T} \boldsymbol{\mu}_i \right\|^2 \\ &+ \sum_{\mathbf{x} \in \mathcal{D}} \left\| P_n^\mathsf{T} V^\mathsf{T} \mathbf{x} - P_n^\mathsf{T} V^\mathsf{T} \boldsymbol{\mu}_\mathcal{D} \right\|^2, \end{aligned} \tag{4.3}$$

where $\boldsymbol{\mu}_\mathcal{D}$ is the data set mean vector and all other definitions of symbols not defined in this section can be found in Section 2.1.1.

4.2.1.2 Optimization Procedure

We can optimize this loss function by adapting the alternating k-means algorithm shown in Algorithm 1. We alternate between assigning the data points and updating the parameters until we converge to a local minimum. In the assignment step, we assign each data point to the cluster with the closest centroid in the *clustered* space. As in the cost function, we use for this the euclidean norm. Additionally, we assign

all data objects to the single cluster in the *noise* space.

In the update step, we first update the cluster centroids. This update is independent of the other parameters because we defined the centroids in terms of the original space and only project them onto the *clustered* space. We simply have to update the centroid μ_i of cluster i with the arithmetic mean of the data points assigned to the cluster:

$$\boldsymbol{\mu}_i := \frac{1}{|\mathcal{C}_i|} \sum_{\mathbf{x} \in \mathcal{C}_i} \mathbf{x}.$$

Next, we update the rotation matrix V and m. For this, we can show that we have to minimize the following loss:

$$\operatorname{Tr}\left(P_c^\mathsf{T} V^\mathsf{T} \left[\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^\mathsf{T} - \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \boldsymbol{\mu}_\mathcal{D})(\mathbf{x} - \boldsymbol{\mu}_\mathcal{D})^\mathsf{T} \right] V P_c \right), \tag{4.4}$$

where $\operatorname{Tr}(\cdot)$ represents the trace of the matrix. Inside the square brackets is a symmetric matrix, consisting of the difference of a double sum on the left and a single sum on the right. The single sum on the right is the scatter matrix of the data set. A scatter matrix is the unnormalized empirical covariance matrix. The double sum on the left is the sum over all cluster-specific scatter matrices.

In order to minimize this function, we take an approach that is similar to the eigen-decomposition we use to solve PCA, as shown in Section 2.2.1. We can minimize the loss by setting m to the number of negative eigenvalues. Then, we put the calculated eigenvectors of the matrix inside the square brackets into V, such that P_c selects the eigenvectors with negative eigenvalues. Essentially, we let the trace sum up all the negative eigenvalues and thereby minimize the sum.

We can initialize the parameters of SubKmeans by first selecting a random orthogonal matrix for V and a value for m smaller than the dimensionality of the data set. We then initialize the centroids like in k-means and can therefore use strategies proposed for this step, such as the previously mentioned k-means++[AV07] procedure.

4.2.2 Nr-Kmeans

Nr-Kmeans [Mau+18] tackles the phenomenon that data in high-dimensional spaces can often be meaningfully clustered in more than one way. Figure 4.2 illustrates this situation. The underlying assumption of Nr-Kmeans is that each of these cluster partitionings is non-redundant. Each clustering lies in its own arbitrarily oriented, linear subspace of the high-dimensional input space (a *clustered* space). We assume that these subspaces (and optionally an additional *noise* space without any cluster structure) are orthogonal to each other. Further, we assume that *k*-means can comprise each clustering structure. These assumptions share some similarities to SubKmeans, and therefore the cost functions and the optimization procedure are quite similar to SubKmeans. In fact, both SubKmeans and *k*-means can even be seen as special cases of Nr-Kmeans. However, the central idea of multiple, non-redundant clusterings in Nr-Kmeans is very different from the single clustering problem the former two algorithms aim for.

One difference between SubKmeans and Nr-Kmeans is that in the former, the *noise* space is an important and necessary part of the algorithm, whereas it is just an optional extension in Nr-Kmeans.

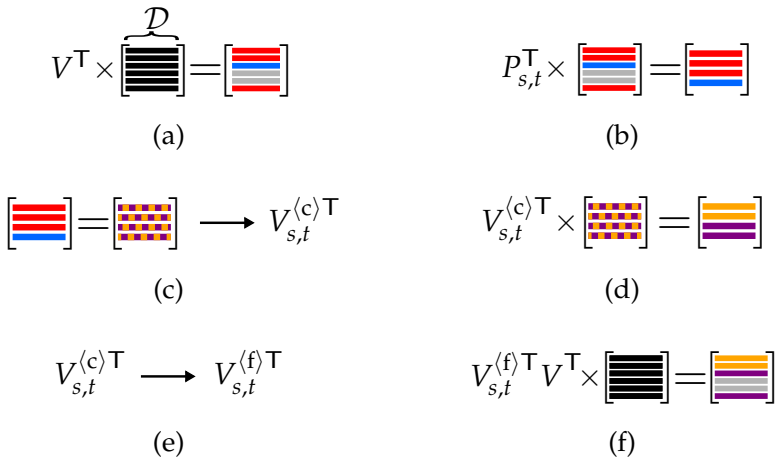


Figure 4.4: The diagrams illustrate how NR-KMEANS optimizes the rotation matrix V in the general case with more than two subspaces. The central idea is to consider each pair of clusterings and their corresponding subspaces separately and seeing them as an instance of the two-subspace case. (a) Here, we assume that we want to optimize V the rotation matrix w.r.t to the two clusterings s (■) and t (■), where rows in the illustrations of the data matrix $\mathcal{D}$ represent features. (b) First, we project both subspaces onto the combined s-t-subspace (with $P_{s,t}$). (c) Next, we assume that the rotation in not yet optimal w.r.t. to these two clusterings and find a rotation matrix $V_{s,t}^{\langle \mathrm{c} \rangle}$ and corresponding projections that optimize the loss in this subspace for s (■ → ■) and t (■ → ■)—as shown in (d). (e) Then, we can translate the rotation matrix $V_{s,t}^{\langle \mathrm{c} \rangle}$ into its full-space equivalent $V_{s,t}^{\langle \mathrm{f} \rangle}$. Further, we have to update the dimension-to-subspace assignments accordingly. (f) Finally, we update $V \leftarrow V V_{s,t}^{\langle \mathrm{f} \rangle}$. Updating these parameters optimizes the objective function w.r.t to both clusterings. Performing these actions for all clustering pairs optimizes the loss function w.r.t. V and the subspace projections P_i.

4.2.2.1 General Concept and the Clustering Objective Function

We adopt most of the notation as introduced for SubKmeans in Section 4.2.1. Like in SubKmeans, we assume that a clustering structure can be captured by the k-means algorithm and keep this representation in the loss function. However, with Nr-Kmeans, we have not one, but S-many of these clusterings and the same amount of arbitrarily oriented *clustered* spaces, each with its own dimensionality m_j. The desired non-redundancy property is induced by the constraint that these subspaces are non-overlapping and orthogonal to each other. Like in SubKmeans, we have to align these subspaces with the coordinate axis. Again we use the orthogonal transformation matrix V for this task.

We can project the by V rotated space onto the subspace of clustering j via the masking matrices $P_j \in \mathbb{R}^{m_j \times d}$ that is defined as follows:

$$P_j[a,b] := \begin{cases} 1, & \text{if dimension } a \text{ of the rotated data space maps} \\ & \quad \text{to subspace dimension } b \text{ of clustering } j \\ 0, & \text{otherwise,} \end{cases} \tag{4.5}$$

where each dimension a is only assigned once to a subspace. Putting all definitions together, we get the following loss function:

$$\mathcal{F} = \sum_{j=1}^{S} \sum_{i=1}^{k_j} \sum_{\mathbf{x} \in \mathcal{C}_{j,i}} \left\| P_j^\mathsf{T} V^\mathsf{T} \mathbf{x} - P_j^\mathsf{T} V^\mathsf{T} \boldsymbol{\mu}_{j,i} \right\|^2 \tag{4.6}$$

The *clustered* space is defined similarly to SubKmeans. However, in contrast to SubKmeans, we are more flexible in Nr-Kmeans and can, depending on our task and goals, decide whether to use or not to use it.

4.2.2.2 The Non-Redundancy Aspect

Our experiments show that Nr-Kmeans indeed finds—induced by the orthogonality constraint—non-redundancy clusterings of high quality. We can give a further theoretic justification for these empirical results by highlighting the relationship of Nr-Kmeans to a mixture of Gaussian mixture models (GMM) with statistically independent subspaces. It is a well-known fact that there exists a direct connection between GMM as defined through the probability distribution

$p(\mathbf{x}) = \Pi_i^k \pi_i \mathcal{N}(\mathbf{x}|\mu_i, \sigma\mathbf{I})$ and the k-means cost function. The cost function of k-means is the limit of the GMM's likelihood function, where we let σ go to zero [KJ12].

We can follow the same approach for NR-KMEANS. It can be viewed as the limit of the log-likelihood function (w.r.t. $\sigma \to 0$) of the following mixture model with S statistically independent subspaces:

$$p(\mathbf{x}) = \prod_j^S \sum_i^{k_j} \pi_{j,i} \mathcal{N}(P_j^\mathsf{T} V^\mathsf{T} \mathbf{x} | P_j^\mathsf{T} V^\mathsf{T} \boldsymbol{\mu}_{j,i}, \sigma\mathbf{I}). \tag{4.7}$$

This connection shows that the orthogonal subspaces of the NR-KMEANS model are the limit of the statistically independent components of the above-defined mixture model.

4.2.2.3 Optimization Procedure

The optimization procedure again alternates between the assignment step and the update step. In the assignment step, we assign for each clustering each data point to the closest centroid of the corresponding subspace. In the update step, we first update the centroids as we do in SUBKMEANS. Next, we have to update V and each masking matrix. Updating V and the masking matrices is not as straightforward as in SUBKMEANS. However, we can utilize the same building block. The essential idea is to consider each pair of clusterings as a special case with only two clusterings s and t. We can optimize the rotation in the combined subspace of s and t and then translate this rotation into an update for V and the masking matrices. Applying this procedure to all pairs optimizes V for the overall loss function. Figure 4.4 illustrates this procedure.

In addition, we can speed the process of optimizing V up. The key trick is that we can determine the sum of scatter matrices of a clustering s within a combined subspace with a second clustering t based on the full-dimensional sum. That way, we have to determine the sum of scatter matrices for each clustering only once after we update the centroids and can use them to optimize V. For each combined subspace, we thereby have to perform the eigen-decomposition like in the following term:

$$\text{eig}\left(P_{s,t}^\mathsf{T} V^\mathsf{T} \left[\mathcal{S}_s - \mathcal{S}_t\right] V P_{s,t}\right), \tag{4.8}$$

where $\mathcal{S}_s$ and $\mathcal{S}_t$ is the full-dimensional scatter matrix of the respective clustering, defined as:

$$\mathcal{S}_c = \sum_{i=1}^{k_c} \sum_{\mathbf{x} \in C_{c,i}} (\mathbf{x} - \boldsymbol{\mu}_{c,i})(\mathbf{x} - \boldsymbol{\mu}_{c,i})^\mathsf{T}, \tag{4.9}$$

for some clustering c.

4.2.3 Nr-DipMeans

One disadvantage of Nr-Kmeans is that the user has to specify the number of clusters for each of the non-redundant clusterings as an input parameter. With the extension Nr-DipMeans [Mau+20], we propose a method that automatically determines the number of clusters in each subspace. This, in essence, leaves only the number of expected subspaces to be specified by the user, which is by far easier to estimate. As an extension of Nr-Kmeans, Nr-DipMeans has the additional assumption that clusters are unimodal (along its direction of highest variance). It utilizes Hartigan's dip test of unimodality [HH85] to identify if a cluster is multimodal and must therefore be split.

4.2.3.1 General Concept

The essential idea of Nr-DipMeans is to lift the user from the burden to specify the number of clusters within each subspace. The general concept of k-means assumes that a single representative centroid can be found, which resides in the center of a cluster. Its connection to Gaussian mixture models also suggests that the assumption of a unimodal cluster—containing only one peak, when regarded as a probability distribution—is sensible. We, therefore, decided to utilize Hartigan's dip test [HH85] to test if a cluster is unimodal.

The problem of using Hartigan's dip test is that it is only defined for one-dimensional data. Yet, Nr-DipMeans is not the first algorithm that combines k-means clustering with one-dimensional statistical hypothesis testing. Thus, we can draw ideas from Dip-means [KL12]—utilizing the dip test—and from G-means [HE04]—utilizing the Anderson-Darling hypothesis test. The Anderson-Darling test determines if data is normally distributed, which we consider too much of a constraint, and instead use the dip test. However, in our experiments, the strategy of G-means—to project the data of a cluster into a single dimension along the direc-

tion of highest variance—worked sufficiently well. Hence, we use this projection strategy and combine it with Hartigan's dip test.

Hartigan's dip test [HH85] is a classical hypothesis test. With the null hypothesis, we assume that a given one-dimensional data set is unimodally distributed. The alternative hypothesis suggests that the data is multimodal. Roughly spoken, if the null hypothesis is true, the empirical cumulative distribution function (CDF) is first convex—up to the mode—and then concave. With the dip test, we can measure the deviation from this expected shape of the CDF, and we can reject the null hypothesis for some significance level.

4.2.3.2 Clustering Objective and the Optimization Algorithm

The clustering objective is essentially the same as in NR-KMEANS. The big difference is that in addition to the loss function in Eq. 4.6, we have the further constraint that the one-dimensional projection of each cluster along the direction of the highest variance must be unimodal. We can find this direction within a cluster approximately by applying two-means (k-means with $k = 2$) to the data points of the cluster. k-means usually splits a cluster along the direction of the highest variance [FS19]. Therefore, we can use a projection onto the connection line between the two centroids as an approximation for this direction. If we can reject the null hypothesis, we found a potential cluster that actually consists of at least two clusters. We will use this in the greedy optimization procedure, explained in the following.

During optimization, our algorithm has to overcome two issues. In the first situation, the clusters in all *clustered* spaces are unimodal, but the optional *noise* space might still contain relevant cluster structures. In the second situation, the current number of clusters is too low for a clustering a. These structures might 'show through' in one or more other subspaces corresponding to other clusterings. As a consequence, one or more clusters in these other subspaces might look multimodal. Splitting these multimodal clusters then might strengthen this phenomenon instead of removing it. An example of these issues can be found in the paper [Mau+20].

The NR-DIPMEANS algorithm has to address both of these issues. The method starts with two clusters in each *clustered* space.

While Hartigan's dip test indicates that the *noise* space seems to be multimodal, we create J new NR-KMEANS candidate configurations, where we increase in the j'th candidate configuration the number of clusters in the j'th clustering by one. We do this by splitting the cluster into two clusters that seems the least unimodal (ac-

cording to the dip test). We then run Nr-Kmeans for each of these candidates and keep the candidate that minimizes the Nr-Kmeans loss function. This procedure only applies if the user wants to utilize the optional *noise* space. It solves the first issue described above.

Once the *noise* space seems to be unimodal, we test each cluster in the *clustered* spaces if they are unimodal. If one of them does not pass the test, we perform the same procedure as described in the last paragraph. We stop the algorithm once every cluster passes the dip test. We circumvent the second issue described above by creating one candidate configuration for each subspace and keeping only the one minimizing the loss function.

4.2.4 DeepECT

In recent years, deep clustering methods have been a very active research direction. Most of these methods share the same conceptional idea, as illustrated in Figure 4.5: an autoencoder is used to perform a nonlinear transformation on the data space. The clustering method operates on the embedded space and aims to provide feedback to the autoencoder on how to improve the representation of the found clustering structure. However, in contrast to the rigid linear transformation, which we employ for SubKmeans, Nr-Kmeans, and Nr-DipMeans, the nonlinear transformation of the autoencoder is powerful enough to transform the embedded space in arbitrary ways. This leads to the problem that the embedded space of a flat deep clustering method will resemble the chosen number of clusters, regardless of the actual number of clusters. Figure 4.6 illustrates this problem. As a result, classical internal cluster evaluation methods, like the in Section 2.1.1 described silhouette coefficient or the elbow method, are not applicable.

DeepECT [MPB19] circumvents this problem by utilizing a hierarchical clustering method. We do not have to specify the number of clusters before the optimization procedure. Instead, we can optimize the tree to a sufficient size. Then, after the training, we can choose the level of detail we want to analyze and for each sub-tree separately.

In summary, DeepECT is quite different from the previous contributions. The first difference is that we employ a nonlinear transformation utilizing autoencoders instead of a rigid linear transformation. The second difference is that DeepECT aims to find a hierarchical clustering structure instead of a flat one.

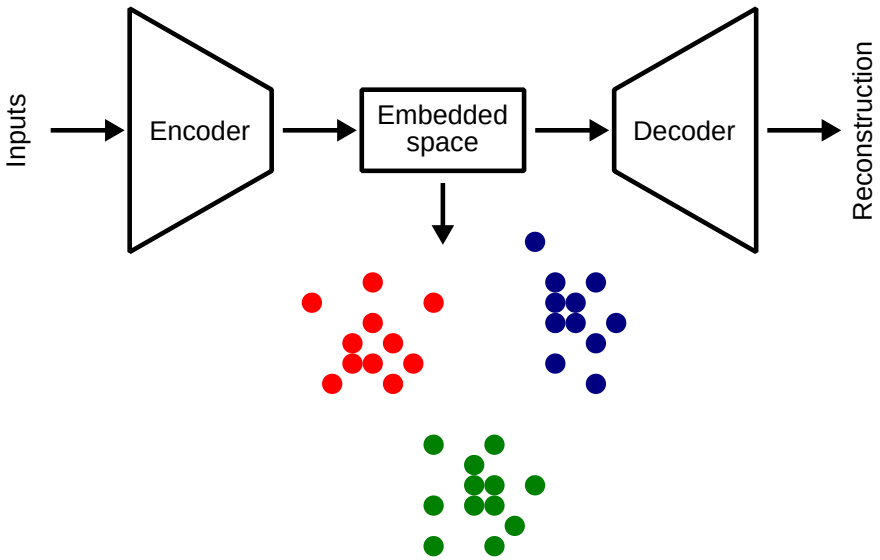
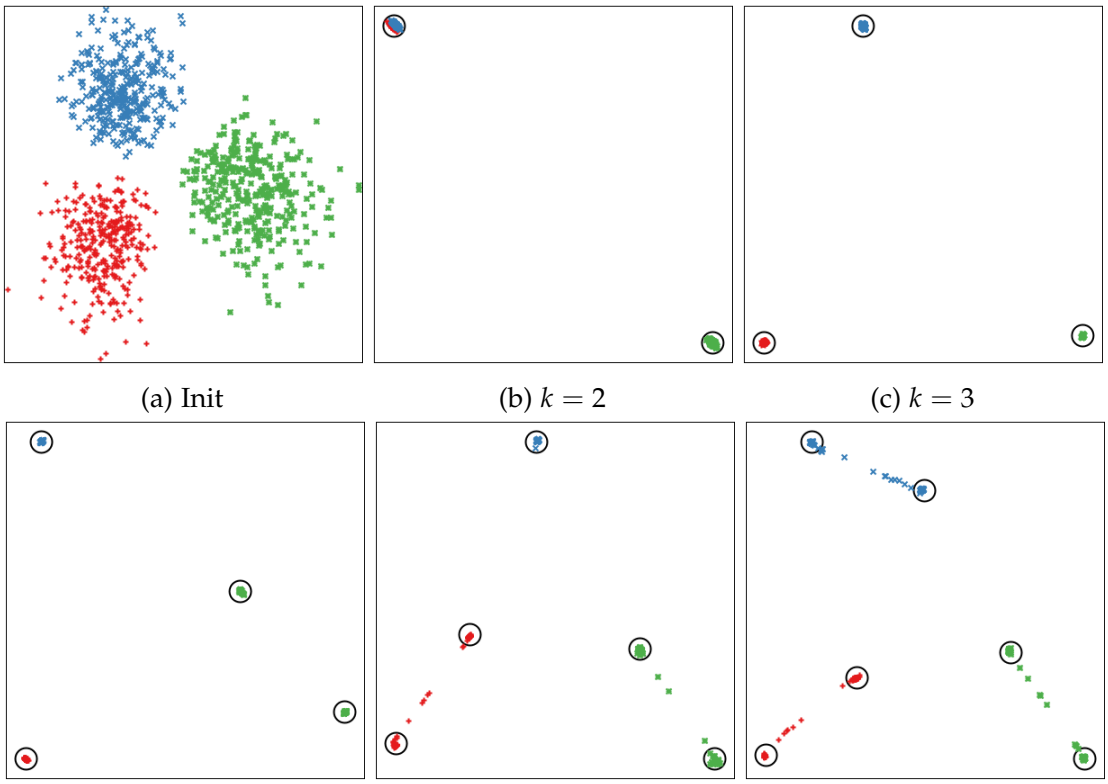


Figure 4.5: The figure illustrates the general idea behind deep clustering. The autoencoder transforms the data such that the structural information is more prominent in the embedding. The goal is to better capture the structural information by a joint optimization of the clustering objective and a nonlinear transformation compared to two separate steps.



(a) Init (b) $k = 2$ (c) $k = 3$

(d) $k = 4$ (e) $k = 5$ (f) $k = 6$

Figure 4.6: The plots show a serious problem of deep clustering methods: the embedded space adapts its appearance to the chosen number of clusters. (a) shows the pre-trained embedding of a data set with three ground-truth clusters (colors represent class labels). The plots (b)–(f) show the results of the deep clustering algorithm IDEC [Guo+17] for different values for the number of clusters k. We can see that after training, the embedding reflects the chosen number of clusters. This is one of the problems we want to overcome with DeepECT.

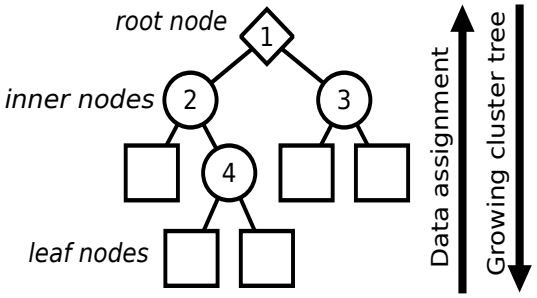


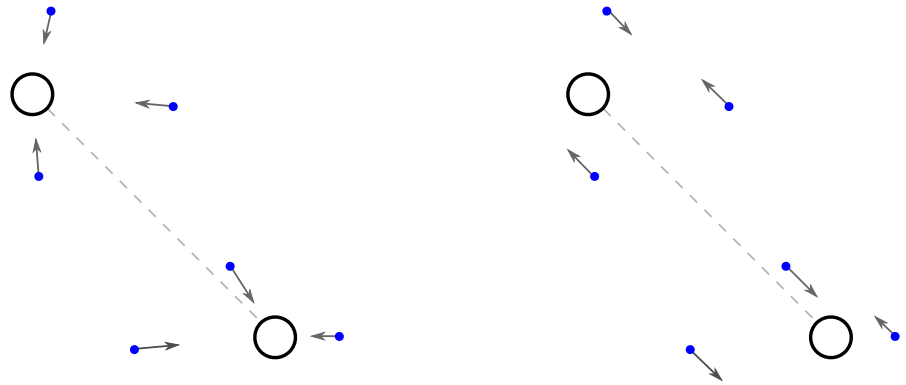
Figure 4.7: The diagram shows an illustration of the tree created by DeepECT. The tree is grown top-down, starting with a single root node. The data assignment is bottom-up. Each node is represented by a centroid. The centroids of inner nodes are weighted averages of their child nodes.

4.2.4.1 General Concepts and the Clustering Objective

The general concept of DeepECT is shown in Figure 4.7. The tree grows top-down, starting with a single root node. We grow the tree at regular intervals within the optimization procedure. Thereby, we select a leaf-node and split it into two sub-clusters with the help of Bisecting-k-means, as presented in Section 2.1.4. In contrast to the top-down Bisecting-k-means algorithm, the data point assignment in DeepECT is bottom-up. We assign each data point to the leaf node with the closest centroid. The inner nodes inherit their assigned data points from their child nodes. Each node is represented by a cluster centroid, much like in k-means. The leaf node centroids are parameters that need to be optimized. However, the centroids of the inner nodes are not parameters but are weighted averages of their child nodes. This reduces the number of parameters to be optimized. Further, it ensures that an inner node's centroid is coherent to the child nodes centroids. For the nonlinear transformation, we make use of an autoencoder with a data-set-specific, sensible reconstruction loss. This ensures that the data can be approximatively reconstructed. Thereby, we ensure that the embedding does not become arbitrary.

Combining these assumptions and definitions yield the loss function of DeepECT. The loss function is defined as the sum of the following three specialized loss functions:

- *Autoencoder reconstruction loss* – It ensures that the embedding is sensible and does not become arbitrary.



(a) Cluster hardening of flat algorithms (b) Cluster hardening in DeepECT

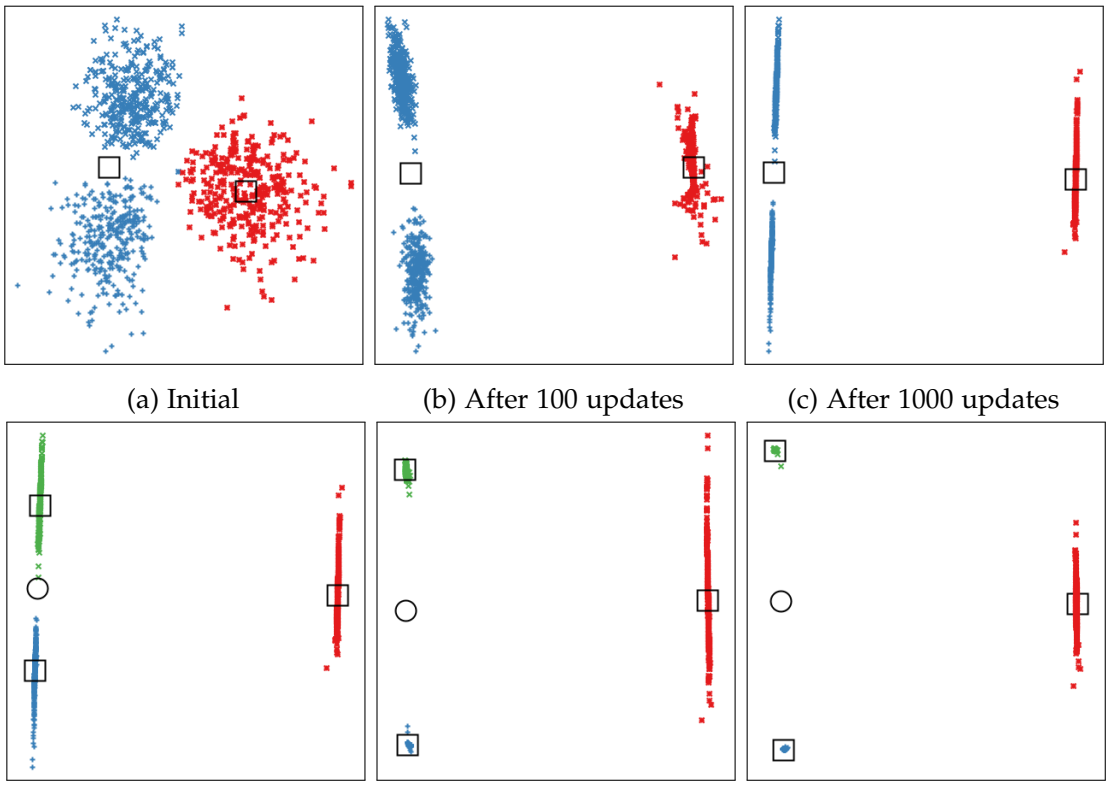
Figure 4.8: The loss functions of deep centroid-based, flat clustering algorithms have a cost function that optimizes the embedding to be closer to the respective centroid, as shown in (a). The optimization procedure of DeepECT is shown in (b). The loss function only optimizes the embedding along the connection line of two sibling cluster nodes. This behavior ensures that orthogonal structures are preserved.

- *Leaf node centroid loss* – This loss ensures that the centroid parameter of each leaf node is moved towards the average value of the data points assigned to it.
- *Cluster hardening loss* – It forces the encoder part of the autoencoder to embed data points closer to the centroids they are assigned to. From a bird's eye perspective, every deep clustering algorithm employs such a loss. However, a crucial distinction for DeepECT is that this happens only projected onto the connection line between a cluster node and its sibling node. This ensures that orthogonal structures are preserved. An illustration comparing the loss of flat deep clustering methods and DeepECT is shown in Figure 4.8.

4.2.4.2 Optimization Procedure

The optimization itself is straightforward. We can optimize the loss function with a gradient-based optimization procedure, such as stochastic gradient descent or Adam [KB15]. Since the data sets in deep learning are usually very large, we utilize

mini-batches instead of calculating the gradient for the whole data set. An example of the optimization procedure of a toy data set is shown in Figure 4.9.



(a) Initial (b) After 100 updates (c) After 1000 updates

(d) After split of left node (e) After 100 updates (f) After 1000 updates

Figure 4.9: The plots show an example of how DeepECT moves data points closer to the assigned cluster centroids but keeps orthogonal structures unchanged. (a)-(c) show the optimization of two sibling nodes (squares). In (d)-(e), the node on the left side is split into two child clusters. We can see how the margin between the two clusters is increased. At the same time, the vertical structure on the right is preserved. This behavior is stable even for a longer training period, as shown in (f).

4.2.5 ENRC

The final contribution is the algorithm ENRC. This method is the result of our research endeavor of combining the concept of deep clustering with non-redundant clustering.

4.2.5.1 General Concept

Much like DeepECT, ENRC employs an autoencoder for the nonlinear transformation, which ensures that the embedding is not arbitrary and contains sensible structures. The clustering objective itself is related to the objective of Nr-Kmeans.

We assume a scenario that several non-redundant clusterings can be found in the embedding of a pre-trained autoencoder. The actual number of clusterings and clusters within is set by the user. As with Nr-Kmeans, we presume that each clustering resides in its own arbitrarily oriented subspace of the embedding. Further, we assume that after the nonlinear transformation, we can rotate the embedded space with an additional linear transformation such that the subspaces axis-align. For this, we employ an approximative orthogonal matrix V. Since we employ a gradient-based optimization method, the subspace assignment has to be differentiable. Therefore, each dimension of the embedded and rotated space is (softly) assigned to a certain (positive) degree to each subspace in such a way that the weights are differentiable and sum to one. Last, we represent the clusters within each subspace by centroids like in Nr-Kmeans and many other deep clustering algorithms. Figure 4.10 gives an illustration of the architecture of ENRC.

4.2.5.2 Clustering Objective

Since we have a soft assignment of the dimensions of the rotated space to a clustering, we cannot simply employ the Euclidean norm to determine the distance within the embedded space. Instead, we use the weighted Euclidean distance to measure the distance between the centroid a and a data point b:

$$||a - b||^2_{\beta_j} := \sum_{i=1}^{n} \beta_j[i](a[i] - b[i])^2, \tag{4.10}$$

where $a[i]$ selects the i'th entry of the vector a and β_j is the dimension assignment vector for the clustering j.

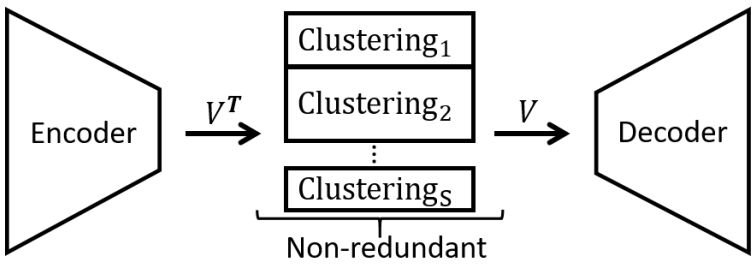


Figure 4.10: The diagram shows a conceptional illustration of ENRC's architecture. The layer V^T and V share the same parameters (they are tied). Analogous to the rotation of SubKmeans and Nr-Kmeans, the purpose of these layers is to align the subspaces of the different clusterings along the axes. The difference is that both layers are only approximatively orthogonal. Both linear layers, together with the encoder network, and the decoder network build a nonlinear transformation of the data space into an embedded space and back into the original space.

For the soft-assignments of the rotated dimensions to the clusterings, we employ the softmax function over an unbounded parameter vector. In that way, the weights are positive and the weights of each dimension sum to one and can be simply optimized by a gradient-based method.

The loss function contains two parts. The first part is the autoencoders reconstruction loss, which we have to adapt in one crucial aspect. The second part is the clustering loss. First, we adopt the reconstruction loss to ensure that V does not degenerate and is approximately orthogonal. We do this by incorporating the orthogonality constraint $VV^\mathsf{T} = \mathbf{I}$ into the reconstruction loss of the autoencoder:

$$\mathcal{L}_{\text{rec}} = \text{dist}[\mathbf{x}, \text{dec}(VV^\mathsf{T}\,\text{enc}(\mathbf{x}))], \tag{4.11}$$

where dist is the regular reconstruction loss function of the autoencoder, for instance, the Euclidean norm. The loss ensures that V will be approximately orthogonal and does not degenerate, but at the same time, it allows V do be optimized w.r.t. the needs of the clustering loss. The second part of the loss term is the clustering loss. For this, we simply penalize the weighted Euclidean distance (seen in Eq. 4.10) between a data point and the cluster centroid it is assigned to for each clustering. Then, the final loss term is the weighted sum between these two loss terms, where

the user has to choose an appropriate weight.

4.2.5.3 Optimization Procedure

During the optimization of ENRC, we employ an adapted version of mini-batch k-means [Scu10] that adjusts the learning rate of each centroid based on the number of data points assigned to each cluster. The parameters V, the subspace weights, and the autoencoder weights can be optimized using a standard gradient descent algorithm. An example application of ENRC can be found in Figure 4.11.

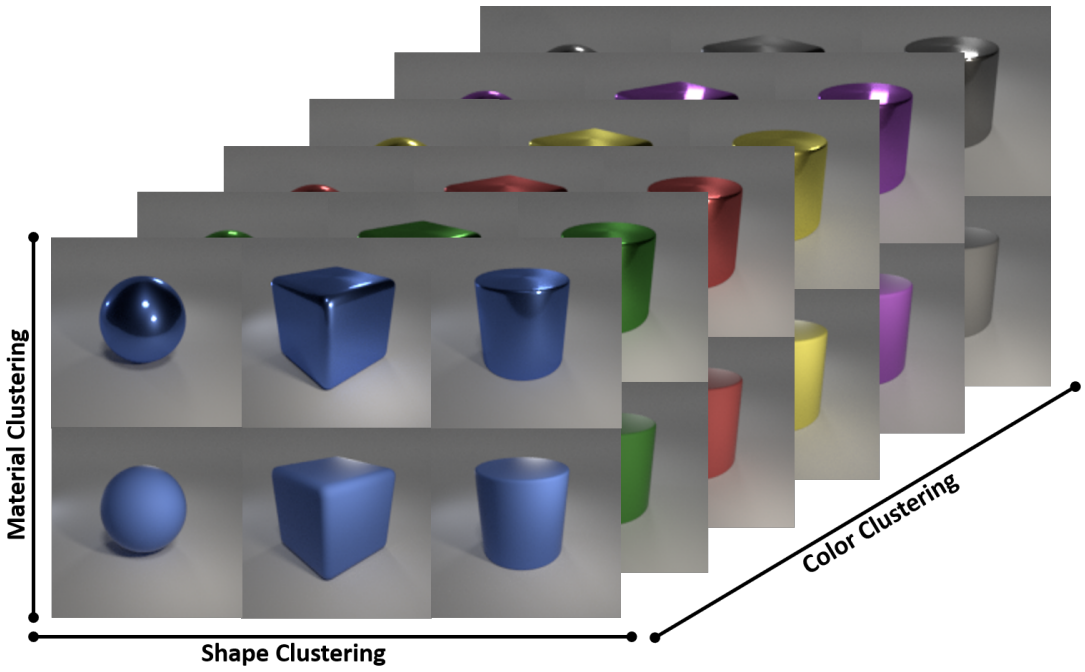
4.3 Connections and Differences among the Methods

We have already introduced the general common properties of the contributed methods in Section 4.1. We have also already mentioned further direct connections in the methods summaries above. The most prominent connection is between SubKmeans and Nr-Kmeans. SubKmeans can be considered as a special case of Nr-Kmeans, where we configure the method to have only a single *clustered* space and a complementing *noise* space. Yet, the general goal of both methods is very different. SubKmeans aims to find a single subspace with a clustering structure hidden in the data, whereas Nr-Kmeans aims to find multiple non-redundant clusterings in different subspaces of the data. Another obvious connection is between Nr-Kmeans and Nr-DipMeans, which is by design as Nr-DipMeans is an extension of Nr-Kmeans.

However, there exist deeper connections and relationships among the contributed methods. Furthermore, there exist connections to the techniques introduces in Chapter 2. In the remainder of this section, we discuss these connections and relationships in more detail.

4.3.1 Eigen-Decomposition – A Central Operation

The eigen-decomposition is an essential part of the optimization procedure for the three contributed clustering methods SubKmeans, Nr-Kmeans, and Nr-DipMeans. The way we use it to optimize a rotation matrix is also connected to the optimization of PCA, presented in Section 2.2.1. All four methods use the eigen-decomposition to find orthogonal directions in the data space.



(a) Objects Data Set

(b) Color Centroids

(c) Material Centroids

(d) Shape Centroids

Figure 4.11: Figure (a) shows a toy data set with images of objects with three different, non-redundant classes: three different shapes (cylinder, sphere, cube), two different materials (dull rubber, reflective metal), and six different colors (yellow, green, red, blue, gray, purple). The plots (b) - (d) show the reconstructed centroids of the ENRC clustering result. We can see that each centroid captures one class and, at the same time, represents an average of the classes of the other clusterings.

The biggest difference between the application in the clustering algorithms compared to PCA is the optimization goal. In PCA, we want to find the direction of highest variance for the whole data set, whereas in the clustering algorithms we want to find directions for which the clustering structure for the given clusterings is most prominent. In case of the used *k*-means-like loss of the clustering algorithms, this is equivalent to the question for the directional trade-off between: (a) the direction in which the data points are closest to their respective centroid for a given clustering and (b) the direction in which the data points are far from the centroids of the other clusterings or the *noise* space centroid. For PCA and SubKmeans, this can be directly seen by comparing the Eq. 2.10 of PCA and Eq. 4.4 of SubKmeans. The trace function in the first equation contains only the scatter matrix of the whole data set, whereas the equation of SubKmeans contains the difference between the sum of scatter matrices of all clusters and the scatter matrix of the whole data set.

The optimization of a rotation itself is also used in ENRC. However, in this method, we use gradient descent and not an eigen-decomposition, which has the advantage that we can use mini-batch updates. The disadvantage is that it might need many mini-batch updates until the rotation is (sufficiently approximately) optimized.

4.3.2 DeepECT and ENRC

The contributed algorithms DeepECT and ENRC also share interesting connections. The most evident once is that both algorithms rely on autoencoders for the nonlinear transformation, whereas the other contributed methods use a linear transformation matrix, as explained above. The autoencoder allows more complex transformations but at the cost of being less explainable and being more complex to optimize.

A second connection between both algorithms is the choice to optimize the centroids using gradient-descent-based strategies. However, there is a difference in the specific optimization strategy. DeepECT introduces a loss term that penalizes the squared error distance between a leaf node's centroid and the average of the data points of the mini-batch assigned to this leaf node. The optimization of this loss is left to the specific gradient-based optimization algorithm chosen by the user. On the other hand, ENRC utilizes the centroid optimization procedure of mini-batch *k*-means [Scu10]. This optimization strategy updates the centroids directly with the mean vector of the assigned data points, but the learning rate depends inversely on the number of data points assigned to the cluster.

4.3.3 Compatibility to Variants and Extensions of *k*-means

k-means is a simple but proven algorithm that lends itself to be modified and adapted to different situations. Over the years, this has led to many proposed extensions and variants. A common property of SubKmeans, Nr-Kmeans, and to some degree of Nr-DipMeans is that they are compatible with many of these extensions.

Some extensions have orthogonal concerns and can be plugged in with minimal modification of our proposed algorithms. For instance, we can use the widely used initialization procedure of *k*-means++ [AV07] to seed the initial centroids in the beginning. Alternatively, we could use the method *k*-means|| [Bah+12], which parallelizes the centroid initialization procedure. Both have stochastic guarantees that they lead on average to better local minima than with random initializations. Further, we could speed up the assignment step by exploiting the triangle inequality [Ham10; Phi02; Elk03], which allows us to minimize the number of clusters a data point has to be compared to. Yet another way to speed up the assignment step is to utilize kd-trees [Moo99; PM99].

The above-mentioned *k*-means extensions are only a small and by far not exhaustive set of examples, which can be adopted for the three algorithms in a straightforward manner. Other extensions and variations can also be adopted and incorporated but need more elaborated modifications of the respective algorithm. An example of such a modification is the automated estimation of the number of clusters *k*. In this regard, Nr-DipMeans is an extension of Nr-Kmeans that draw ideas from Dip-means [KL12] and from G-means [HE04]. But we can also propose extensions for either SubKmeans or Nr-Kmeans that draw exclusively from one of these papers. Of course, the same is true for other publications. For instance, instead of hypothesis testing, we could extend the cost function by a term that penalizes complexity like in X-means [PM+00]. In fact, we worked on such an idea in the paper 'Automatic Parameter Selection for Non-Redundant Clustering Algorithms' (Section 3.3, Item viii.). It proposes an extension of Nr-Kmeans that utilizes the minimum description length principle [Ris78] to estimate both the number of subspaces, as well as the number of clusters within each subspace. Further examples are the loss function variations of *k*-means like Fuzzy-c-means (Section 2.1.2) or different metrics like *k*-medians (Section 2.1.1). Indeed, we have already started to explore both directions. In a bachelor thesis [Sel19], we have combined SubKmeans with Fuzzy-c-means, which allows finding a fuzzy clustering located in

a subspace of the data space. In another bachelor thesis [Grö17], we have explored the idea to use other M-estimators [Mar+19]—a robust generalization of maximum likelihood estimators, e.g., the median—to estimate centroids and rotation matrix of SubKmeans, which allows these variants to be more robust against outliers and noise within a data set.

4.3.4 From SubKmeans to ENRC

In this section, we show the link between SubKmeans and ENRC.

We can see this relationship by approximating SubKmeans with a loss function that we can optimize with gradient descent. First, we cast the rotation matrix V as a single-layer linear autoencoder. This also ensures that V remains approximately orthogonal. We can do this by penalizing the squared euclidean distance between a data point and its reconstruction after rotation and reverse rotation:

$$\min \left\| \mathbf{x} - VV^{\mathsf{T}}\mathbf{x} \right\|_2^2 \qquad w.r.t \quad V. \tag{4.12}$$

At first glance, this looks like the PCA-like linear autoencoder loss function in Eq. 2.12. Yet, in contrast to Eq. 2.12, here we have defined V as a $\mathbb{R}^{d \times d}$ matrix, which does not introduce a dimensional reduction, i.e., a bottleneck in the embedded space. Instead, we need to split up the transformed space like the projection matrices P_c and P_n do in the SubKmeans objective (discussed in Section 4.2.1). At the same time, this split needs to be differentiable in order to be optimized by gradient-based methods. Similar to ENRC—as briefly discussed in Section 4.2.5.2 and in more detail in the ENRC paper [Mik+20]—we can introduce a differentiable variable β_i for each dimension i of the rotated space with $\beta_i \in [0; 1]$. This weight indicates how much this dimension belongs to the *clustered* space. Consequently, we define the weight for the *noise* space as $1 - \beta_i$. The distance measure between a data point and the cluster center becomes the weighted Euclidean distance, as defined in Eq. 4.10. Restricting the β_i-weight to the domain $[0; 1]$ is a problem for gradient-based methods. Either we could clip the values into this range after each parameter update, or we could use an unconstrained parameter b_i and transform it into the $[0; 1]$ domain, for instance, via the sigmoid function: $\frac{1}{1+e^{-b}}$. We can define

the combined loss term of the *clustered* space and *noise* space as follows:

$$\sum_{i=1}^{k} \left\| V^\mathsf{T}\mathbf{x} - \boldsymbol{\mu}_i \right\|_{\beta}^2 + \left\| V^\mathsf{T}\mathbf{x} - \boldsymbol{\mu}_{\mathcal{D}} \right\|_{(1-\beta)}^2, \tag{4.13}$$

which we have to optimize w.r.t $V, \beta, \boldsymbol{\mu}_0, \cdots, \boldsymbol{\mu}_k, \boldsymbol{\mu}_{\mathcal{D}}$. In contrast to SubKmeans, do we not need to keep the centroids in the original space, but the gradient-based optimization function allows us to define them as parameters in the rotated space. We can simply add the loss terms Eq. 4.12 and Eq. 4.13 to get a loss term that we can optimize by using an off the shelf gradient-based optimizer. For each provided mini-batch, we first assign each data point to the closest centroid based on the weighted euclidean distance and then run the optimizer for one optimization step. We repeat this procedure until we converge.

To get a gradient-based version of Nr-Kmeans, we can simply add additional *clustered* spaces to the loss term in Eq. 4.13. Of course, we then have to assign each data point to the closest centroid for each clustering for each mini-batch.

Finally, we can modify Eq. 4.12 by adding an additional encoder network before the rotation and an additional decoder network added after the back-rotation. This results in the adapted reconstruction loss of ENRC in Eq. 4.11. The steps above show how we can modify and adapt both SubKmeans and Nr-Kmeans in such a way that it results in a version of the ENRC method.

However, one property that we cannot directly transfer to ENRC with a nonlinear autoencoder is the *noise* space. In the linear case, we have to approximate a rotation in V. This means we also approximately preserve the variance within the data. Yet, if we naively apply the *noise* space loss of SubKmeans and Nr-Kmeans directly to ENRC, we introduce a term in the loss function that incentives the optimization algorithm to embed data points closer and closer to the single centroid of the *noise* space, without the need to preserve the initial variance. A simple illustrative example is that the last layer in the encoder just learns to divide the input by a very large number, and the first nonlinear layer in the encoder learns at the same time to reverse this division by multiplying its input by the same large number. In that way, it shrinks the embedded space significantly and, therefore, also any distance between point-pairs within it. The optimizing procedure is able to reduce the *noise* space loss significantly without actually doing what we expect from it: introducing a variance preserving space that encodes the global, non-cluster-specific variations within a data set. We should note that such shrinkage, in general, is not a problem

in deep learning and actually a *noise* space loss term, where we fix the centroid to be equal to the origin would be similar to the common $L2$ parameter regularization technique.

There are some research directions to prevent this behavior through more elaborate loss functions for the *noise* space that enforce specific shapes. For instance, we could use a Kullback-Leibler divergence loss, forcing the shape to follow a standard normal Gaussian distribution, or we could use adversarial training [Mak+15]. Yet, these modifications would also require more tuning and adjustments between the *noise* space and the *clustered* spaces losses, which in turn makes the optimization procedure more complex. Therefore, we decided to make a *noise* space a non-target in our research for ENRC. However, this can also be seen as a research opportunity.

5 Conclusion

With this cumulative thesis, we have contributed five algorithms: SubKmeans, Nr-Kmeans, Nr-DipMeans, DeepECT, and ENRC. The main objective of these algorithms is to combine cluster analysis with unsupervised feature transformation methods. Each algorithm was designed to assist in the search for structures in an unknown data set. Further, most algorithms allow us to easily visualize the found structures, which in turn enables the user to inspect and validate the findings. The experimental results—found in the respective publication in the appendix—empirically show that the joint optimization procedure is advantageous compared to optimizing feature transformations and clustering objectives separately. Therefore, we claim that all contributed algorithms achieved the primary goal.

However, all contributed algorithms have implicit assumptions and shortcomings, many of which are inherited from *k*-means, like the convexity assumption of clusters or the effect that clusters tend to be of uniform sizes [Wu12, p. 32f]. The problem that we can only find local optima is almost universal for clustering algorithms as they tend to be NP-hard. Other disadvantages stem from the transformation method, like the black-box property of the autoencoder's nonlinear transformation. These drawbacks are by no means a knock-out criteria, but the taken design decisions and implicit assumptions make the algorithms better suited for certain data sets than others. This is an inherent problem of all data mining algorithms and is related to the 'no free lunch theorem' in supervised machine learning [Wol96]. Consequently, there is no universally best algorithm, and each data mining method needs informed users to decide which tools to use in which situation.

5.1 Future Work

The drawbacks of the contributed algorithms are, at the same time, research opportunities for future projects. Each of the contributed algorithms offers concrete

opportunities for modifications and extensions. We have already discussed some of these possible extensions for SubKmeans, Nr-Kmeans, and Nr-DipMeans in Section 4.3.3. Both SubKmeans and Nr-Kmeans, have already directly influenced the works of other researchers, which provides further examples for research opportunities. The authors of [VSS19] propose to use random projections for the optimization of V in SubKmeans. The researchers in [Zha+19] propose a noise-robust clustering algorithm that is heavily influenced by SubKmeans and Nr-Kmeans. In the paper [Le+19], the authors propose to use the cost function of SubKmeans in a supervised setting for the hierarchical encoding of sequential data. A concrete, open research question for ENRC is the open question on how to select the number of subspaces and clusters within, which is more complicated than for its linear counterpart Nr-Kmeans. Also, selecting a suitable autoencoder architecture is an open question, which it shares with DeepECT.

Furthermore, the general goal of joint optimizations of transformation and clustering still provides countless research opportunities. This applies in particular to the combination of unsupervised deep learning and cluster analysis, which offers many low-hanging fruits. Most deep clustering algorithms utilize centroids, which are a straightforward choice but might not be the best way to represent clusters inside a neural network. Therefore, developing algorithms that utilize other representations of these clusters is an exciting research direction. Another factor that is important—not only to deep clustering but to the whole field of deep learning—is to improve the explainability of neural networks. Deep clustering will benefit from every advancement in this direction.

Another direction might be to explore other ways to facilitate other nonlinear transformation methods than autoencoders. One aspect of autoencoders is that they aim to preserve as many details of the encoded object as possible in order to be able to reconstruct it faithfully. This facet is in strong contrast to a clustering algorithm's goal to find more general structures in the data set and ignore minor differences and variances within a cluster [EM19; Häu+18].

5.2 Final Remarks

This thesis has contributed five algorithms to the area of cluster analysis with joint feature transformation. Each contributed algorithm has its own exciting applications and solves specific problems in the area of unsupervised learning. At the

same time, each algorithm is only but a small contribution to the vast and ever-growing field of data mining. Nevertheless, the hope is that these algorithms will help practitioners and inspire researchers to answer research questions.

List of Figures and Tables

Further References

[Aar13] Y. B. and Aaron C. Courville and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[Aka98] H. Akaike. "Information theory and an extension of the maximum likelihood principle". In: *Selected papers of hirotugu akaike*. Springer, 1998, pp. 199–213.

[AR16] C. Aggarwal and C. Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2016. ISBN: 9781498785778.

[AV07] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. Ed. by N. Bansal, K. Pruhs, and C. Stein. SIAM, 2007, pp. 1027–1035.

[Bah+12] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. "Scalable k-means++". In: *Proceedings of the VLDB Endowment* 5.7 (2012), pp. 622–633.

[Bal12] P. Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[Bar12] D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[BB95] L. Bottou and Y. Bengio. "Convergence properties of the k-means algorithms". In: *Advances in neural information processing systems*. 1995, pp. 585–592.

[BBV04] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Ber06] P. Berkhin. "A survey of clustering data mining techniques". In: *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[Bez81] J. C. Bezdek. "Pattern recognition with fuzzy objective function algorithms". In: (1981).

[Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[Bro20] W. J. Broad. *A.I. Versus the Coronavirus*. The New York Times. 2020. URL: https://www.nytimes.com/2020/03/26/science/ai-versus-the-coronavirus.html (visited on 05/11/2020).

[Car+20] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. *Exploring Neural Networks with Activation Atlases*. OpenAI. 2020. DOI: 10.23915/distill.00015. URL: https://distill.pub/2019/activation-atlas/ (visited on 09/29/2020).

[Cas16] A. Castrounis. *Data Science and Big Data, Explained*. KDnuggets. 2016. URL: https://www.kdnuggets.com/2016/11/big-data-data-science-explained.html (visited on 05/24/2020).

[CHU07] C. Chen, W. Härdle, and A. Unwin. *Handbook of Data Visualization*. Springer Handbooks of Computational Statistics. Springer Berlin Heidelberg, 2007. ISBN: 9783540330370.

[Cui09] Y. Cui. "Non-redundant clustering, principal feature selection and learning methods applied to lung tumor image-guided radiotherapy". PhD thesis. Northeastern University, 2009.

[DP12] T. H. Davenport and D. Patil. "Data Scientist: The Sexiest Job of the 21st Century". In: *Harvard business review* 90.5 (2012), pp. 70–76.

[Dun73] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of Cybernetics* 3.3 (1973), pp. 32–57. DOI: 10.1080/01969727308546046.

[Elk03] C. Elkan. "Using the triangle inequality to accelerate k-means". In: *ICML*. Vol. 3. 2003, pp. 147–153.

[EM19] B. Epstein and R. Meir. "Generalization bounds for unsupervised and semi-supervised learning with autoencoders". In: *arXiv preprint arXiv:1902.01449* (2019).

[Ern06] M. Ernst. *Choosing a venue: conference or journal?* University of Washington. 2006. URL: https://homes.cs.washington.edu/~mernst/advice/conferences-vs-journals.html (visited on 05/24/2020).

[FPS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3 (1996), pp. 37–37.

[FS19] P. Fränti and S. Sieranoja. "How much can k-means be improved by using better initialization and repeats?" In: *Pattern Recognition* 93 (2019), pp. 95–112. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.04.014.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[GGR07] P. Grünwald, A. Grunwald, and J. Rissanen. *The Minimum Description Length Principle*. Adaptive computation and machine learning. MIT Press, 2007. ISBN: 9780262072816.

[GH15] A. Gandomi and M. Haider. "Beyond the hype: Big data concepts, methods, and analytics". In: *International Journal of Information Management* 35.2 (2015), pp. 137–144. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2014.10.007.

[GJW82] M. Garey, D. Johnson, and H. Witsenhausen. "The complexity of the generalized Lloyd - Max problem (Corresp.)" In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 255–256.

[GMW07] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial and Applied Mathematics, 2007. ISBN: 9780898716238.

[Goo20a] Google. *Top publications – Artificial Intelligence*. 2020. URL: https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_artificialintelligence (visited on 12/18/2020).

[Goo20b] Google. *Top publications – Data mining & Analysis*. 2020. URL: https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_dataminininganalysis (visited on 12/18/2020).

[GR12] J. Gantz and D. Reinsel. "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east". In: *IDC iView: IDC Analyze the future* 2007.2012 (2012), pp. 1–16.

[Grö17] B. Gröttrup. "Robust Clustering with Dimensionality Reduction: Outlier-Resistant Variations of SubKmeans". Bachelor's thesis. Ludwig-Maximilians-Universität München, 2017.

[Guo+17] X. Guo, L. Gao, X. Liu, and J. Yin. "Improved Deep Embedded Clustering with Local Structure Preservation". In: *Proceedings of the 26th IJCAI, Melbourne, Australia, August 19-25, 2017*. 2017, pp. 1753–1759. DOI: 10.24963/ijcai.2017/243.

[Ham10] G. Hamerly. "Making k-means even faster". In: *Proceedings of the 2010 SIAM international conference on data mining*. SIAM. 2010, pp. 130–140.

[Häu+18] P. Häusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers. "Associative Deep Clustering: Training a Classification Network with No Labels". In: *Pattern Recognition - 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings*. Ed. by T. Brox, A. Bruhn, and M. Fritz. Vol. 11269. Lecture Notes in Computer Science. Springer, 2018, pp. 18–32. DOI: 10.1007/978-3-030-12939-2_2.

[HE04] G. Hamerly and C. Elkan. "Learning the k in k-means". In: (2004).

[Hef14] J. Hefferon. *Linear Algebra, Free Book*. 2014.

[Hei17] L. Heiler. *Difference of Data Science, Machine Learning and Data Mining*. Data Science Central. 2017. URL: https://www.datasciencecentral.com/profiles/blogs/difference-of-data-science-machine-learning-and-data-mining (visited on 05/24/2020).

[HH85] J. A. Hartigan and P. M. Hartigan. "The Dip Test of Unimodality". In: *Ann. Statist.* 13.1 (Mar. 1985), pp. 70–84. DOI: 10.1214/aos/1176346577.

[HPK11] J. Han, J. Pei, and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011. ISBN: 9780123814807.

[KB15] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015.

[KJ12] B. Kulis and M. I. Jordan. "Revisiting k-means: New algorithms via Bayesian nonparametrics". In: *In Proceedings of the 23rd International Conference on Ma- chine Learning* (2012).

[KKZ09] H.-P. Kriegel, P. Kröger, and A. Zimek. "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering". In: *ACM Trans. Knowl. Discov. Data* 3.1 (2009), 1:1–1:58. DOI: 10.1145/1497577.1497578.

[KL12] A. Kalogeratos and A. Likas. "Dip-means: an incremental clustering method for estimating the number of clusters". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* Ed. by P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. 2012, pp. 2402–2410.

[Kle03] J. M. Kleinberg. "An impossibility theorem for clustering". In: *Advances in neural information processing systems*. 2003, pp. 463–470.

[KM16] R. Kitchin and G. McArdle. "What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets". In: *Big Data & Society* 3.1 (2016), p. 2053951716631130. DOI: 10.1177/2053951716631130.

[Kri+10] H. Kriegel, A. Zimek, L.-M.-U. Muenchen, and S. Clustering. "Subspace Clustering, Ensemble Clustering, Alternative Clustering, Multiview Clustering: What Can We Learn From Each Other?" In: 2010.

[KS96] D. J. Ketchen and C. L. Shook. "The application of cluster analysis in Strategic Management Research: An analysis and critique". In: *Strategic Management Journal* 17.6 (1996), pp. 441–458. DOI: 10.1002/(SICI)1097-0266(199606)17:6<441::AID-SMJ819>3.0.CO;2-G.

[Le+19] H. Le, M. Xu, T. Hoang, and M. Milford. "Hierarchical Encoding of Sequential Data With Compact and Sub-Linear Storage Cost". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 9823–9832. DOI: 10.1109/ICCV.2019.00992.

[Lip15] Z. C. Lipton. *Data Science's Most Used, Confused, and Abused Jargon*. KDnuggets. 2015. URL: https://www.kdnuggets.com/2015/02/data-science-confusing-jargon-abused.html (visited on 05/25/2020).

[Llo82] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[LV07] J. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Information Science and Statistics. Springer New York, 2007. ISBN: 9780387393513.

[Mak+15] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015).

[Mar+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Url: `https://www.tensorflow.org/`. 2015.

[Mar+19] R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibián-Barrera. *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019.

[Mau+17] D. Mautz, W. Ye, C. Plant, and C. Böhm. "Towards an Optimal Subspace for K-Means". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 365–373. DOI: 10.1145/3097983.3097989.

[Mau+18] D. Mautz, W. Ye, C. Plant, and C. Böhm. "Discovering Non-Redundant K-means Clusterings in Optimal Subspaces". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Y. Guo and F. Farooq. ACM, 2018, pp. 1973–1982. DOI: 10.1145/3219819.3219945.

[Mau+20] D. Mautz, W. Ye, C. Plant, and C. Böhm. "Non-Redundant Subspace Clusterings with Nr-Kmeans and Nr-DipMeans". In: *ACM Trans. Knowl. Discov. Data* 14.5 (June 2020). ISSN: 1556-4681. DOI: 10.1145/3385652.

[May16] M. Mayo. *Data Science Basics: Data Mining vs. Statistics*. KDnuggets. 2016. URL: https://www.kdnuggets.com/2016/09/data-science-basics-data-mining-statistics.html (visited on 05/24/2020).

[MH08] L. v. d. Maaten and G. Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[MH18] L. McInnes and J. Healy. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *CoRR* (2018). arXiv: 1802.03426.

[Mik+20] L. Miklautz, D. Mautz, M. C. Altinigneli, C. Böhm, and C. Plant. "Deep Embedded Non-Redundant Clustering". In: *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, New York, USA, February 7 - February 12, 2020*. 2020.

[Moo99] A. W. Moore. "Very fast EM-based mixture model clustering using multiresolution kd-trees". In: *Advances in Neural information processing systems* (1999), pp. 543–549.

[MPB19] D. Mautz, C. Plant, and C. Böhm. "Deep Embedded Cluster Tree". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*. Ed. by J. Wang, K. Shim, and X. Wu. IEEE, 2019, pp. 1258–1263. DOI: 10.1109/ICDM.2019.00157.

[MSM18] G. Montavon, W. Samek, and K.-R. Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2017.10.011.

[Mul+12] E. Muller, S. Gunnemann, I. Farber, and T. Seidl. "Discovering multiple clustering solutions: Grouping objects in different views of the data". In: *2012 IEEE 28th International Conference on Data Engineering*. IEEE. 2012, pp. 1207–1210.

[NBS10] T. T. Ngo, M. Bellalij, and Y. Saad. "The Trace Ratio Optimization Problem for Dimensionality Reduction". In: *SIAM J. Matrix Analysis Applications* 31.5 (2010), pp. 2950–2971. DOI: 10.1137/090776603.

[Oja89] E. Oja. "Neural networks, principal components, and subspaces". In: *International journal of neural systems* 1.01 (1989), pp. 61–68.

[Ola+20] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. *Thread: Circuits – What can we learn if we invest heavily in reverse engineering a single neural network?* OpenAI. 2020. DOI: 10.23915/distill.00024. URL: https://distill.pub/2020/circuits/ (visited on 09/29/2020).

[Pas+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Url: https://pytorch.org/. Curran Associates, Inc., 2019, pp. 8024–8035.

[Phi02] S. J. Phillips. "Acceleration of k-means and related clustering algorithms". In: *Workshop on Algorithm Engineering and Experimentation*. Springer. 2002, pp. 166–177.

[PJ92] B. T. Polyak and A. B. Juditsky. "Acceleration of Stochastic Approximation by Averaging". In: *SIAM J. Control Optim.* 30.4 (July 1992), pp. 838–855. ISSN: 0363-0129. DOI: 10.1137/0330046.

[Pla18] E. Plaut. "From Principal Subspaces to Principal Components with Linear Autoencoders". In: *CoRR* abs/1804.10253 (2018). arXiv: 1804.10253.

[PM+00] D. Pelleg, A. W. Moore, et al. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." In: *ICML*. Vol. 1. 2000.

[PM99] D. Pelleg and A. Moore. "Accelerating exact k-means algorithms with geometric reasoning". In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 1999, pp. 277–281.

[Rap17] RapidMiner. *What Are Artificial Intelligence, Machine Learning, and Deep Learning?* KDnuggets. 2017. URL: https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html (visited on 05/23/2020).

[Ris78] J. Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(78)90005-5.

[Ros20] C. Rosenthal. *The perils of Big Data: How crunching numbers can lead to moral blunders*. The Washington Post. 2020. URL: https://www.washingtonpost.com/outlook/2019/02/18/perils-big-data-how-crunching-numbers-can-lead-moral-blunders/ (visited on 05/09/2020).

[Rou87] P. 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.

[Sch78] G. Schwarz. "Estimating the Dimension of a Model". In: *Ann. Statist.* 6.2 (Mar. 1978), pp. 461–464. DOI: 10.1214/aos/1176344136.

[Scu10] D. Sculley. "Web-scale k-means clustering". In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 1177–1178.

[Sel19] S. Selle. "Fuzzy-SubKmeans - Anpassen des SubKmeans-Algorithmus zur Fuzzy Cluster Analyse". Bachelor's thesis. Ludwig-Maximilians-Universität München, 2019.

[SKK+00] M. Steinbach, G. Karypis, V. Kumar, et al. "A comparison of document clustering techniques". In: *KDD workshop on text mining*. Vol. 400. 1. Boston. 2000, pp. 525–526.

[Tau20] T. Taulli. *Machine Learning: What Is It Really Good For?* Forbes. 2020. URL: https://www.forbes.com/sites/tomtaulli/2020/05/23/machine-learning-what-is-it-really-good-for/ (visited on 05/23/2020).

[TBB13] D. T. Truong, R. Battiti, and M. Brunato. "Discovering Non-redundant Overlapping Biclusters on Gene Expression Data". In: *2013 IEEE 13th International Conference on Data Mining*. 2013, pp. 747–756.

[TH12] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[Too14] J. Toonders. *Data Is the New Oil of the Digital Economy*. Wired. 2014. URL: https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/ (visited on 05/12/2020).

[Val18a] I. Valchanov. *Data Science vs Machine Learning vs Data Analytics vs Business Analytics*. KDnuggets. 2018. URL: https://www.kdnuggets.com/2018/05/data-science-machine-learning-business-analytics.html (visited on 05/24/2020).

[Val18b] I. Valchanov. *The What, Where and How of Data for Data Science*. KDnuggets. 2018. URL: https://www.kdnuggets.com/2018/06/what-where-how-data-science.html (visited on 05/23/2020).

[VSS19] T. Vannoy, J. Senecal, and V. Strnadova-Neeley. "Improved Subspace K-Means Performance via a Randomized Matrix Decomposition". In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 2019, pp. 1–5.

[Wax13] C. Waxer. *How data mining can boost your revenue by 300%*. CNN. 2013. URL: https://money.cnn.com/2013/10/28/smallbusiness/data-mining/ (visited on 05/09/2020).

[Wei13] S. Weinberger. *The darker side of data science*. BBC. 2013. URL: https://www.bbc.com/future/article/20130620-the-darker-side-of-big-data (visited on 05/11/2020).

[Wol96] D. H. Wolpert. "The lack of a priori distinctions between learning algorithms". In: *Neural computation* 8.7 (1996), pp. 1341–1390.

[Wu+08] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. S. Steinbach, D. J. Hand, and D. Steinberg. "Top 10 algorithms in data mining". In: *Knowl. Inf. Syst.* 14.1 (2008), pp. 1–37. DOI: 10.1007/s10115-007-0114-2.

[Wu12] J. Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

[XT15] D. Xu and Y. Tian. "A comprehensive survey of clustering algorithms". In: *Annals of Data Science* 2.2 (2015), pp. 165–193.

[XW05] R. Xu and D. C. Wunsch. "Survey of Clustering Algorithms". In: *IEEE Transcations on Neural Networks* 16.3 (2005), p. 645.

[YW18] Y. Yang and H. Wang. "Multi-view clustering: A survey". In: *Big Data Mining and Analytics* 1.2 (2018), pp. 83–107.

[Zei12] M. D. Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[Zha+19] Z. Zhang, C. Gao, C. Liu, Q. Yang, and J. Shao. "Towards Robust Arbitrarily Oriented Subspace Clustering". In: *Database Systems for Advanced Applications - 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22-25, 2019, Proceedings, Part I*. Ed. by G. Li, J. Yang, J. Gama, J. Natwichai, and Y. Tong. Vol. 11446. Lecture Notes in Computer Science. Springer, 2019, pp. 276–291. DOI: 10.1007/978-3-030-18576-3_17.

Appended Papers

The remainder of this dissertation contains the contributed publications. Further, we provide essential information about the publication, such as the title, abstract, the conference or the journal it was published, all authors, the division of work among the authors, and the assigned Digital Object Identifier (DIO).

Paper A: *Towards an Optimal Subspace for K-Means*

Title	Towards an Optimal Subspace for K-Means
Authors	Dominik Mautz, Wei Ye, Claudia Plant, Christian Böhm
Publication Outlet	Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining *Note: CORE ranking: A*; Accepted as an oral presentation; acceptance rate for oral presentations: 10.9%*
DIO-URL	`https://doi.org/10.1145/3097983.3097989`

Division of Work

Dominik Mautz: conceiving the original idea; problem identification and definition; formulating the loss function and optimization strategy; implementation of the algorithm and experiments; experimental design; execution of experiments; review of relevant related work; writing major parts of paper; creating the conference poster, video, and presentation; presenting at the conference.

Wei Ye: experimental setup and evaluation for the comparison methods; writing parts of the experiments section; regular discussions of candidate methods and potential experiments; suggestions for enhancements and improvements of the algorithm.

Claudia Plant: writing parts of the introduction; identification of relevant related work; periodic review of drafts for the paper; assisting in improving the draft; mentoring and supervision.

Christian Böhm: regular discussions of candidate methods and potential experiments; suggestions for enhancements and improvements of the algorithm and the paper; periodic review of drafts for the paper; refinement of the final draft; mentoring and supervision.

Abstract Is there an optimal dimensionality reduction for *k*-means, revealing the prominent cluster structure hidden in the data? We propose SubKmeans, which extends the classic *k*-means algorithm. The goal of this algorithm is twofold: find a sufficient *k*-means-style clustering partition and transform the clusters onto a common subspace, which is optimal for the cluster structure. Our solution is able to pursue these two goals simultaneously. The dimensionality of this subspace is found automatically and therefore the algorithm comes without the burden of additional parameters. At the same time this subspace helps to mitigate the *curse of dimensionality*. The SubKmeans optimization algorithm is intriguingly simple and efficient. It is easy to implement and can readily be adopted to the current situation. Furthermore, it is compatible to many existing extensions and improvements of *k*-means.

Thesis-Reference [Mau+17]

The paper was removed for the publication version of this thesis.

Paper B: *Discovering Non-Redundant K-means Clusterings in Optimal Subspaces*

Title	Discovering Non-Redundant K-means Clusterings in Optimal Subspaces
Authors	Dominik Mautz, Wei Ye, Claudia Plant, Christian Böhm
Publication Outlet	Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining *Note: CORE ranking: A*; Accepted as an oral presentation; acceptance rate for oral presentations: 8.5%*
DIO-URL	`https://doi.org/10.1145/3219819.3219945`

Division of Work

Dominik Mautz: conceiving the original idea; problem identification and definition; formulating the loss function and optimization strategy; implementation of the algorithm and experiments; experimental design; execution of experiments; review of relevant related work; writing the majority of the paper; creating the conference poster, video, and presentation; presenting at the conference.

Wei Ye: suggestions for enhancement and improvements of the algorithm; experimental setup and evaluation for the comparison methods; writing parts of the experiments section; frequent discussions during all phases of development.

Claudia Plant: identification of relevant related work; writing parts of the related work; assisting in improving the draft; frequent discussions during all phases of development; mentoring and supervision.

Christian Böhm: writing parts of the introduction; suggestions for enhancements and improvements of the algorithm; periodic review of drafts for the paper; refining the final draft; frequent discussions during all phases of development; mentoring and supervision.

Abstract A huge object collection in high-dimensional space can often be clustered in more than one way, for instance, objects could be clustered by their shape or alternatively by their color. Each grouping represents a different view of the data set. The new research field of *non-redundant clustering* addresses this class of problems. In this paper, we follow the approach that different, non-redundant *k*-means-like clusterings may exist in different, arbitrarily oriented subspaces of the high-dimensional space. We assume that these subspaces (and optionally a further *noise* space without any cluster structure) are orthogonal to each other. This assumption enables a particularly rigorous mathematical treatment of the non-redundant clustering problem and thus a particularly efficient algorithm, which we call NR-KMEANS (for non-redundant *k*-means). The superiority of our algorithm is demonstrated both theoretically, as well as in extensive experiments.

Thesis-Reference [Mau+18]

The paper was removed for the publication version of this thesis.

Paper C: *Deep Embedded Cluster Tree*

Title	Deep Embedded Cluster Tree
Authors	Dominik Mautz, Claudia Plant, Christian Böhm
Publication Outlet	Proceedings of the IEEE International Conference on Data Mining, ICDM 2019 *Note: CORE ranking: A*; Acceptance rate: 18.4%*
DIO-URL	`https://doi.org/10.1109/ICDM.2019.00157`
Division of Work	Dominik Mautz: conceiving the original idea; problem identification and definition; formulating the loss function and optimization strategy; implementation of the algorithm and experiments; selection and execution of experiments; evaluation of comparison techniques; parts of the results discussion; review of related work; writing the majority of paper; conference presentation. Claudia Plant: suggestions for potential experiments; identification of known related work; regular discussions of candidate methods and experiments; periodic review of drafts for the paper; mentoring and supervision. Christian Böhm: writing parts of the introduction and discussion sections; regular discussions of candidate methods and experiments; suggestions for enhancements and improvements; periodic review of drafts for the paper; refinement of the final draft; mentoring and supervision.

Abstract The idea of combining the high representational power of deep learning techniques with clustering methods has gained much attention in recent years. Optimizing the representation and clustering simultaneously has been shown to have an advantage over optimizing them separately. However, so far all proposed methods have been using a flat clustering strategy, with the true number of clusters known a priori. In this paper, we propose the **Deep** **E**mbedded **C**luster **T**ree (DeepECT), the first divisive hierarchical embedded clustering method. The cluster tree does not need to know the true number of clusters during optimization. Instead, the level of detail to be analyzed can be chosen afterward and for each sub-tree separately. An optional data-augmentation-based extension allows DeepECT to ignore prior-known invariances of the dataset, such as affine transformations in image data. We evaluate and show the advantages of DeepECT in extensive experiments.

Thesis-Reference [MPB19]

The paper was removed for the publication version of this thesis.

Paper D: *Deep Embedded Non-Redundant Clustering*

Title	Deep Embedded Non-Redundant Clustering
Authors	Lukas Miklautz, Dominik Mautz, Muzaffer Can Altinigneli, Christian Böhm and Claudia Plant
Publication Outlet	Proceedings of the 34th AAAI Conference on Artificial Intelligence *Note: CORE ranking: A*; Acceptance rate: 20.6%*
DIO-URL	`https://doi.org/10.1609/aaai.v34i04.5961`

Division of Work

This contribution was developed with intensive cooperation between the first two authors with daily meetings, pair programming sessions and fine grained distribution of work, resulting in equal contribution of the first two authors to this paper.

Lukas Miklautz: problem identification, definition and refinement; formulating the loss function and optimization strategy; implementation and testing of variations of the method; experimental design; execution of experiments; writing and visualization of major parts of the article review of relevant related work; creating the conference poster and presentation.

Dominik Mautz: conceiving the initial idea; problem identification, definition and refinement; formulating the loss function and optimization strategy; implementation and testing of variations of the method; experimental design; execution of experiments; writing and visualization of major parts of the article; review of relevant related work;

Muzaffer Can Altinigneli: development and implementation of the autoencoders part; regular discussions of candidate methods and experiments; writing and visualizing parts of the experiments sections in the main paper and supplementary.

Christian Böhm: regular discussions of candidate methods and potential experiments; suggestions for enhancements and improvements; periodic review of drafts for the paper; mentoring and supervision.

Claudia Plant: regular discussions of candidate methods and potential experiments; suggestions for enhancement and improvements; periodic review of drafts for the paper; refining the final draft; mentoring and supervision.

Abstract Complex data types like images can be clustered in multiple valid ways. Non-redundant clustering aims at extracting those meaningful groupings by discouraging redundancy between clusterings. Unfortunately, clustering images in pixel space directly has been shown to work unsatisfactory. This has increased interest in combining the high representational power of deep learning with clustering, termed deep clustering. Algorithms of this type combine the non-linear embedding of an autoencoder with a clustering objective and optimize both simultaneously. None of these algorithms try to find multiple non-redundant clusterings. In this paper, we propose the novel Embedded Non-Redundant Clustering algorithm (ENRC). It is the first algorithm that combines neural-network-based representation learning with non-redundant clustering. ENRC can find multiple highly non-redundant clusterings of different dimensionalities within a data set. This is achieved by (softly) assigning each dimension of the embedded space to the different clusterings. For instance, in image data sets it can group the objects by color,material and shape, without the need for explicit feature engineering. We show the viability of ENRC in extensive experiments and empirically demonstrate the advantage of combining non-linear representation learning with non-redundant clustering.

Thesis-Reference [Mik+20]

The paper was removed for the publication version of this thesis.

Paper E: *Non-Redundant Subspace Clusterings with Nr-Kmeans and Nr-DipMeans*

Title	Non-Redundant Subspace Clusterings with Nr-Kmeans and Nr-DipMeans
Authors	Dominik Mautz, Wei Ye, Claudia Plant, Christian Böhm
Publication Outlet	ACM Transactions on Knowledge Discovery from Data (KDD 2018 Special Issue) *Note: This journal paper is an extension of the 2018 SIGKDD paper above. It includes at least 30% new material and was invited for a special issue featuring top papers from the SIGKDD 2018 conference.*
DIO-URL	`https://doi.org/10.1145/3385652`

Division of Work

Dominik Mautz: conceiving the original idea for both algorithms; problem identification and definition; formulating the loss function and optimization strategy; implementation of the algorithms and experiments; experimental design; execution of experiments; review of relevant related work; writing the majority of the article.

Wei Ye: suggestions for enhancement and improvements of the algorithm; experimental setup and evaluation for the comparison methods; writing parts of the experiments section; frequent discussions during all phases of development.

Claudia Plant: identification of relevant related work; writing parts of the related work; assisting in improving the draft; frequent discussions during all phases of development; mentoring and supervision.

Christian Böhm: writing parts of the introduction; suggestions for enhancements and improvements of the algorithm; periodic review of drafts for the paper; refining the final draft; frequent discussions during all phases of development; mentoring and supervision.

Abstract A huge object collection in high-dimensional space can often be clustered in more than one way, for instance, objects could be clustered by their shape or alternatively by their color. Each grouping represents a different view of the data set. The new research field of *non-redundantclustering* addresses this class of problems. In this paper, we follow the approach that different, non-redundant *k*-means-like clusterings may exist in different, arbitrarily oriented subspaces of the high-dimensional space. We assume that these subspaces (and optionally a further *noise space* without any cluster structure) are orthogonal to each other. This assumption enables a particularly rigorous mathematical treatment of the non-redundant clustering problem and thus a particularly efficient algorithm, which we call Nr-Kmeans (for non-redundant *k*-means). The superiority of our algorithm is demonstrated both theoretically, as well as in extensive experiments. Further, we propose an extension of Nr-Kmeans that harnesses Hartigan's dip test to identify the number of clusters for each subspace automatically.

Thesis-Reference [Mau+20]

The paper was removed for the publication version of this thesis.