NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

Data Science and Information Technologies

SPECIALIZATION

Big Data and Artificial Intelligence

Master's Thesis

Subspace Clustering: A Possibilistic Approach

Stavroula G. Eleftheraki

ATHENS

OCTOBER 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

Επιστήμη Δεδομένων και Τεχνολογίες Πληροφορίας

ΕΙΔΙΚΕΥΣΗ

Μεγάλα Δεδομένα και Τεχνητή Νοημοσύνη

Μεταπτυχιακή Εργασία

Συσταδοποίηση Υποχώρων: Μία Possibilistic Προσέγγιση

Σταυρούλα Γ. Ελευθεράκη

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2020

Master's Thesis

Subspace Clustering: A Possibilistic Approach

Stavroula G. Eleftheraki

S.N.: DS1180005

SUPERVISORS:

Konstantinos D. Koutroumbas, Research Director, National Observatory of Athens

Athanasios A. Rontogiannis, Research Director, National Observatory of Athens

EXAMINATION COMMITTEE:

Konstantinos D. Koutroumbas, Research Director, National Observatory of Athens

Athanasios A. Rontogiannis, Research Director, National Observatory of Athens

Ioannis Panagakis, Associate Professor, National and Kapodistrian University of Athens

OCTOBER 2020

Διπλωματική Εργασία

Συσταδοποίηση Υποχώρων: Μία Possibilistic Προσέγγιση

Σταυρούλα Γ. Ελευθεράκη

Α.Μ.: DS1180005

ΕΠΙΒΛΕΠΟΝΤΕΣ:

Κωνσταντίνος Δ. Κουτρούμπας, Διευθυντής Ερευνών, Εθνικό Αστεροσκοπείο Αθηνών

Αθανάσιος Α. Ροντογιάννης, Διευθυντής Ερευνών, Εθνικό Αστεροσκοπείο Αθηνών

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Κωνσταντίνος Δ. Κουτρούμπας, Διευθυντής Ερευνών, Εθνικό Αστεροσκοπείο Αθηνών

Αθανάσιος Α. Ροντογιάννης, Διευθυντής Ερευνών, Εθνικό Αστεροσκοπείο Αθηνών

Ιωάννης Παναγάκης, Αναπληρωτής Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΟΚΤΩΒΡΙΟΣ 2020

ABSTRACT

Subspace clustering is the problem of modeling a collection of data points lying in one or more subspaces in the presence of noise, outliers and missing data. To the best of our knowledge, all the algorithms associated to this problem follow a hard-clustering philosophy. The study presented in this thesis explores the effectiveness of the possibilistic approach, giving rise to a novel iterative algorithm, called sparse adaptive possibilistic K-subspaces (SAP K-subspaces). SAP K-subspaces algorithm generalizes the sparse possibilistic c-means algorithm (SPCM) [24]. Hence, it inherits the ability to handle reliably data corrupted by noise and containing outliers, as well as data points near the intersections of subspaces. In addition, the new algorithm is suitably initialized with more clusters than those actually exist in the data set and has the ability to gradually eliminate the unnecessary ones in order to conclude with the true clusters, formed by the data. Moreover, it adopts the low-rank approach, introduced in [10], in order to estimate the dimension of the involved subspaces. Experiments on both synthetic and real data illustrate the effectiveness of the proposed method.

SUBJECT AREA: Unsupervised Machine Learning

KEYWORDS: clustering-alternating minimization, cluster elimination, low-rank, sparsity, subspace clustering, parameter adaptation, principal component analysis, possibilistic clustering

ΠΕΡΙΛΗΨΗ

Ως συσταδοποίηση υποχώρων ορίζεται το πρόβλημα της μοντελοποίησης δεδομένων που βρίσκονται σε έναν ή και περισσότερους υποχώρους υπό την παρουσία θορύβου και περιέχουν ακραίες παρατηρήσεις και ελλιπή δεδομένα. Εξ όσων γνωρίζουμε, όλοι οι αλγόριθμοι που επιλύουν αυτό το πρόβλημα υποθέτουν ότι μια παρατήρηση ανήκει αυστηρά σε έναν υποχώρο. Η παρούσα διατριβή εξετάζει την περίπτωση όπου ένα σημείο μπορεί ταυτόχρονα και ανεξάρτητα να ανήκει σε παραπάνω από έναν υποχώρο. Ως αποτέλεσμα έχουμε την δημιουργία ενός καινούργιου αλγορίθμου, του sparse adaptive possibilistic K-subspaces (SAP K-subspaces). Ο αλγόριθμος αυτός αποτελεί γενίκευση του αλγορίθμου sparse possibilistic c-means algorithm (SPCM) [24], πράγμα που σημαίνει ότι μπορεί να διαχειριστεί με αξιοπιστία δεδομένα τόσο με θόρυβο και ακραίες τιμές όσο και δεδομένα τα οποία βρίσκονται σε τομές υποχώρων. Επίσης, ο καινούργιος αλγόριθμος αρχικοποιείται με περισσότερες συστάδες από τις πραγματικές, έχοντας την δυνατότητα απαλοιφής των περιττών συστάδων και τελικά την εύρεση αυτών που σχηματίζονται απο τα δεδομένα. Επιπλέον, υιοθετεί μια προσέγγιση εύρεσης γινομένου πινάκων χαμηλής τάξης για την εκτίμηση της διάστασης των υποχώρων [10]. Πειράματα σε συνθετικά και αληθινά δεδομένα επιβεβαιώνουν την αποτελεσματικότητα του αλγορίθμου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μη Επιβλεπόμενη Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: συσταδοποίηση-εκ περιτροπής ελαχιστοποίηση, απαλοιφή συστάδων, χαμηλή τάξη, αραιότητα, συσταδοποίηση υποχώρων, προσαρμογή παραμέτρων, ανάλυση κύριων συνιστωσών, συσταδοποίηση με βάση τα ενδεχόμενα

ACKNOWLEDGEMENTS

I would like to express my very great appreciation to Dr Konstantinos D. Koutroumbas and Dr Athanasios A. Rontogiannis, Research Directors of National Observatory of Athens, for introducing me to the amazing world of unsupervised machine learning, provided me guidance during the development of this research work. Moreover, I wish to thank Dr Ioannis Panagakis, Associate Professor of National and Kapodistrian University of Athens, for his kind words and comments about this work. I would also like to thank my parents for their unconditional and endless support. Finally, thanks should be given to my friend Dr Costas Leon, ex-lecturer of Democritus University of Thrace, for his continued support and encouragement throughout this study.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

Clustering is a well known multivariate data analysis method, which aims at grouping a set of entities so that similar entities are assigned in the same group/cluster and less similar entities are assigned to different clusters (provided, of course, that the entities tend to form aggregations). It is common for the entities to be represented by a set of L proper selected *features*. In this vein, each entity is associated with an L-dimensional feature vector, while the space where these vectors live is called *feature space*. Assuming that there is structure hidden on it (the data form clusters), our goal is to effectively detect it and conclude about the nature of the entities under study.

There exist many clustering algorithms in the bibliography based on very diverse criteria. Although a complete categorization is impossible, roughly speaking, we can distinguish four major categories: (a) *sequential algorithms*, where the resulting clusters are obtained after one or few sequencial passes on the data set, (b) *cost function optimization algorithms* where the problem is formulated as a cost function optimization, (c) *hierarchical algorithms* where a sequence of clusterings is generated and (d) all the other clustering algorithms that do not fit (in general) the previous categories, for instance, *competitive learning algorithms*, *graph theory based algorithms*, *spectral clustering algorithms* and *subspace clustering algorithms*.

The definition of a cluster is an important issue a clustering algorithm has to face. Thus, some clustering algorithms use all of its points, while others use a set of parameters associated with it. In the last case, the set of parameters define a representative structure associated with the cluster under study, each cluster has a representative, the so called *cluster representative*. There are various types of cluster representatives. The most popular is the point representative, where a a cluster is represented by a point in the feature space (this is suitable for the case where compact clusters around a point are expected). A point representative does not necessary belong to this set of data. In K-means for instance, representatives are averages of different groups of data.

In this thesis, we study the problem of subspace clustering, that is, the problem of grouping high-dimensional data that lie in multiple subspaces. More specifically, the problem here is twofold: first, the clusters themselves need to be determined and second, their associated subspaces need also be determined. The algorithms that fall into this category (typically) represent the clusters by a set of parameters defining the subspace where

they live. Generally, there are four types of methods for subspace clustering: (a) *algebraic methods*, which are based on linear and polynomial algebra concepts, (b) *statistical methods*, which adopt approaches from the field of statistics, (c) *spectral based methods*, which utilize spectral graph theory tools and (d) *self-expressive methods*, which rely on properties of data points lying in a union of linear subspaces. Currently, the last type of methods is the most effective.

A major issue in the problem of clustering is the definition of the relation of a data point with a cluster. There are three main philosophies here namely, (a) the *hard clustering philosophy*, where each data point belongs to a single cluster, (b) the *fuzzy clustering philosophy*, where a point may be shared among more than one cluster and (c) *possibilistic clustering philosophy*, where what matters is the relation of a data point with the clusters. Note that in the frist two cases, the relation of a given data vector $\boldsymbol{x}$ with a certain cluster C_j is affected by the relation of $\boldsymbol{x}$ with the other clusters. On the other hand, in the third case, the relation of the data point with a cluster is independent from its relation with another cluster.

Oddly enough, all the existing subspace clustering methods in the literature follow a hard-clustering philosophy. In contrast to that, in this thesis, we developped a possibilistic clustering algorithm called *sparse adaptive possibilistic K-subspaces* (SAP K-subspaces). Since the algorithm derives from the optimization of a suitably defined cost function, SAP K-subspaces is a cost function optimization clustering algorithm. The proposed algorithm is robust to noise and outliers as it constitutes a generalization of the *sparse possibilistic c-means algorithm* (SPCM) [24]. In addition, starting with an overestimated number of clusters, the proposed algorithm has the ability to adapt both the number of subspaces (clusters) and their dimensions.

The rest of this thesis is organised as follows. Chapter 2 explores several dimensionality reduction methods. Chapter 3 provides some usefull definitions and information about prior work in subspace clustering. Chapter 4 presents the proposed sparse adaptive possibilistic K-subspaces clustering algorithm and its main components. Chapter 5 carefully tests SAP K-subspaces algorithm on artificial data of both linear and affine subspaces, as well as on real data of motion segmentation in video sequences. Chapter 6 summarises the main findings of this work. Chapter 7 highlights areas for further research. Appendix A provides the adopted notation and all the required mathematical concepts. Finally, Appendix B presents a brief review of the main concepts of spectral clustering.

2. DIMENSIONALITY REDUCTION METHODS

In this chapter, we study dimensionality reduction methods. In the concept of subspace clustering, dimensionality reduction methods are used in order to estimate the dimension of the involved subspaces.

2.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a well-known multivariate analysis technique that can be used for data modeling, compression and visualization purposes. Although there has been a long time since it was introduced [11, 19], it is still one of the most widely techniques when it comes to dimensionality reduction. Machine learning workers that usually have to deal with huge amount of data, often utilize PCA in order to reduce the number of their features/variables, thus reduce complexity over a particular problem. On the other hand, statisticians that have only a couple of variables for their cause, often use it as a tool to introduce independence which is a desired property of many statistical methods.

2.1.1 Statistical view of PCA

Given a set of variables $X = \{x_1, x_2, ..., x_L\}$ we seek to find the orthonormal axes, columns of $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_L] \in \mathbb{R}^{L \times L}$, that explain better the variability of the data set. Those axes are called *principal axes*. This procedure can be seen as a rotation of the initial axes (see figure 1). By projecting the data onto this new coordinate system, we not only reformulate the initial variables to new uncorrelated ones but also, we can identify the less important of them based on their associated variance information associated (eigenvalues of $\boldsymbol{\Sigma}_X$). The new produced variables are called *principal components*. Thus, the initial data set X is transformed to $Y = \{y_1, y_2, ..., y_L\}$.

In the sequel, we describe in some detail the rationale behind PCA. Let $\boldsymbol{x} = [x_1, x_2, ..., x_L]^\top \in \mathbb{R}^L$ be a random vector that follows a multivariate normal distribution with $\mathsf{E}(\boldsymbol{x}) = \boldsymbol{0}$ (centered at the origin) and covariance matrix $\mathsf{Var}(\boldsymbol{x}) = \boldsymbol{\Sigma}_X \in \mathbb{R}^{L \times L}$. Our goal is to find a new orthonormal basis $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_L\}$ that explains as good as possible the variability of the data set. We consider only the case where the eigenvalues of $\boldsymbol{\Sigma}_X$ are distinct from each

other.

Let $y_c = \boldsymbol{u}_c^\top \cdot \boldsymbol{x}$, where $\boldsymbol{u}_c$ the c-row of the matrix $\boldsymbol{U}$, be the projection of a data point on the direction defined by $\boldsymbol{u}_i$. Thus, the first principal direction, $\hat{\boldsymbol{u}}_1$, results through maximization of $\text{Var}(y_1) = \text{E}(y_1^2) - \text{E}(y_1)^2$ ①. Assuming centered data, $\text{E}(y_1) = \text{E}(\boldsymbol{u}_1^\top \boldsymbol{x}) = \boldsymbol{u}_1^\top \text{E}(\boldsymbol{x}) = 0$, ① gives $\text{Var}(y_1) = \text{E}(y_1^2) = \text{E}((\boldsymbol{u}_1^\top \boldsymbol{x})^2) = \text{E}(\boldsymbol{u}_1^\top \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{u}_1) = \boldsymbol{u}_1^\top \text{E}(\boldsymbol{x}\boldsymbol{x}^\top)\boldsymbol{u}_1 = \boldsymbol{u}_1^\top \text{Var}(\boldsymbol{x})\boldsymbol{u}_1 = \boldsymbol{u}_1^\top \boldsymbol{\Sigma}_X \boldsymbol{u}_1$.

The optimization problem we have to solve is the following,

$$\max_{\boldsymbol{u}_1} \boldsymbol{u}_1^\top \boldsymbol{\Sigma}_X \boldsymbol{u}_1 \quad \text{s.t.} \quad \|\boldsymbol{u}_1\|_2^2 = \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1. \tag{2.1.1}$$

The constraint of eq. 2.1.1 prohibits us from having $\hat{\boldsymbol{u}}_1 = +\infty$. For ease, we set squared euclidean norm equal to one.

From Lagrange multipliers, we reformulate the problem to that of obtaining

$$\{\hat{\boldsymbol{u}}_1\} = \underset{\boldsymbol{u}_1}{\text{argmax}}\, \boldsymbol{u}_1^\top \boldsymbol{\Sigma}_X \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1). \tag{2.1.2}$$

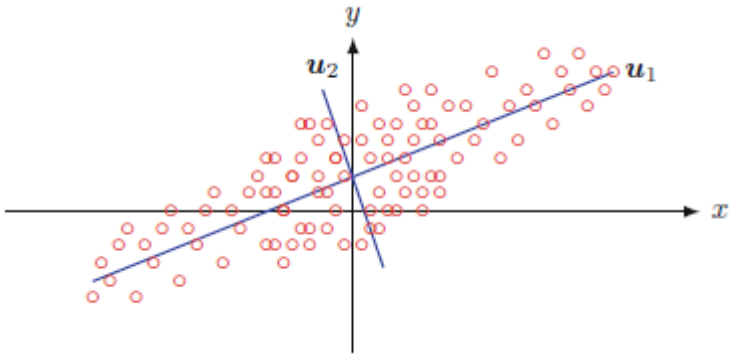


Figure 1: Principal axes over a 2-dimensional data set

Source: [23]

By taking the partial derivative of eq. 2.1.2 and setting equal to zero, we have that

$$\boldsymbol{\Sigma}_X \boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1. \tag{2.1.3}$$

Therefore, the direction along which the data exhibit the maximum spreadness is the eigenvector of $\boldsymbol{\Sigma}_X$, $\hat{\boldsymbol{u}}_1$ associated with its largest eigenvalue.

Matrix $\boldsymbol{\Sigma}_X$ is of size $L \times L$. Thus, there are L eigenvalues and L eigenvectors to compute. From theory we know that eigenvalues are roots of a polynomial and that they can be real

or complex. However, since $\mathbf{\Sigma}_X$ is symmetric, its eigenvalues are real and non-negative. Moreover, its eigenvectors can be chosen to be orthogonal[1].

It is desirable for the second principal component to be uncorrelated with the first that is,

$$\mathsf{Corr}(y_2, y_1) = 0 \Leftrightarrow \frac{\mathsf{Cov}(y_2, y_1)}{\sqrt{\mathsf{Var}(y_2)}\sqrt{\mathsf{Var}(y_1)}} = 0 \Leftrightarrow \frac{\mathsf{E}(y_2 y_1) - \mathsf{E}(y_2)\mathsf{E}(y_1)}{\sqrt{\mathsf{Var}(y_2)}\sqrt{\mathsf{Var}(y_1)}} = 0. \tag{2.1.4}$$

The above is true for

$$\begin{aligned} \mathsf{E}(y_2 y_1) &= 0 \\ \mathsf{E}(\boldsymbol{u}_2^\top \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{u}_1) &= 0 \\ \boldsymbol{u}_2^\top \mathsf{E}(\boldsymbol{x}\boldsymbol{x}^\top)\boldsymbol{u}_1 &= 0 \\ \boldsymbol{u}_2^\top \mathbf{\Sigma}_X \boldsymbol{u}_1 &= 0. \end{aligned} \tag{2.1.5}$$

Observe that $\mathsf{Cov}(y_2, y_1) = 0$ which means that the two first principal components are uncorrelated[2].

Combining eq. 2.1.3 and 2.1.5, we have that $\lambda_1 \boldsymbol{u}_2^\top \boldsymbol{u}_1 = 0$. We know that $\lambda_1 > 0$ because $\mathbf{\Sigma}_X$ is symmetric. Therefore, $\boldsymbol{u}_2^\top \boldsymbol{u}_1 = 0$ which leads us to the conclusion that uncorrelated principal components imply perpendicular directions.

To obtain the second principal axis, we must solve the following problem:

$$\max_{\boldsymbol{u}_2} \boldsymbol{u}_2^\top \mathbf{\Sigma}_X \boldsymbol{u}_2 \quad \text{s.t.} \quad \boldsymbol{u}_2^\top \boldsymbol{u}_2 = 1 \text{ and } \boldsymbol{u}_2^\top \boldsymbol{u}_1 = 0. \tag{2.1.6}$$

Using Langrange multipliers, the problem becomes that of

$$\{\hat{\boldsymbol{u}}_2\} = \operatorname*{argmax}_{\boldsymbol{u}_2} \boldsymbol{u}_2^\top \mathbf{\Sigma}_X \boldsymbol{u}_2 + \lambda_2(1 - \boldsymbol{u}_2^\top \boldsymbol{u}_2) + \kappa \boldsymbol{u}_2^\top \boldsymbol{u}_1. \tag{2.1.7}$$

We take the partial derivative of 2.1.7 and we set it equal to zero.

$$2\mathbf{\Sigma}_X \boldsymbol{u}_2 - 2\lambda_2 \boldsymbol{u}_2 + \kappa \boldsymbol{u}_1 = 0 \tag{2.1.8}$$

Multiplying from the left with $\boldsymbol{u}_1^\top$ we have.

$$\begin{aligned} 2\boldsymbol{u}_1^\top \mathbf{\Sigma}_X \boldsymbol{u}_2 - 2\lambda_2 \boldsymbol{u}_1^\top \boldsymbol{u}_2 + \kappa \boldsymbol{u}_1^\top \boldsymbol{u}_1 &= 0 \\ \xRightarrow[\boldsymbol{u}_2^\top \boldsymbol{u}_1 = 0]{\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1} 2\boldsymbol{u}_1^\top \mathbf{\Sigma}_X \boldsymbol{u}_2 + \kappa &= 0 \\ \xRightarrow{2.1.3} 2\lambda_1 \boldsymbol{u}_1^\top \boldsymbol{u}_2 + \kappa &= 0 \\ \kappa &= 0 \end{aligned}$$

[1] In the case of distinct eigenvalues, the eigenvectors are orthogonal. In the case where some eigenvalues coincide, the eigenvectors can be chosen to be orthogonal.

[2] uncorrelated does not implie independence.

For $\kappa = 0$, eq. 2.1.8 gives

$$\boldsymbol{\Sigma}_X \boldsymbol{u}_2 = \lambda_2 \boldsymbol{u}_2. \tag{2.1.9}$$

Despite the fact that $\kappa = 0$, it is $\boldsymbol{u}_2^\top \boldsymbol{u}_1 = 0$ because $\boldsymbol{\Sigma}_X$ is symmetric and $\lambda_1 \neq \lambda_2$. Thus, λ_2 and $\hat{\boldsymbol{u}}_2$ are the second largest eigenvalue and eigenvector respectively. The proof continues by showing that λ_L and $\hat{\boldsymbol{u}}_L$ are the L-largest eigenvalue and eigenvector respectively.

In case where the data are not centered, we estimate the actual $\boldsymbol{\mu}$ with $\mathrm{E}(\boldsymbol{x})$ and we subtract it from every data point $\boldsymbol{x}_i$. We continue by factorizing $\boldsymbol{\Sigma}_X$. Bear in mind that if the different variables are measured in different units instead of $\boldsymbol{\Sigma}_X$ we use the correlation matrix of X which is nothing but the scaled version $\boldsymbol{\Sigma}_X$.

2.1.2 Geometric view of PCA

Any point of an affine subspace S can be written as a linear combination of its associated linear subspace basis $\boldsymbol{U}$ plus a fixed vector $\boldsymbol{\mu}$:

$$S = \{\boldsymbol{x} \in \mathbb{R}^L : \boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{U}\boldsymbol{y}, \boldsymbol{\mu} \in \mathbb{R}^L, \boldsymbol{U} \in \mathbb{R}^{L \times d}, \boldsymbol{y} \in \mathbb{R}^d\}. \tag{2.1.10}$$

Note that for $\boldsymbol{\mu} = \boldsymbol{0}$ the subspace is linear.

In real world applications the data do not lie perfectly in subspaces due to noise. Thus, for one data point we have that $\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{U}\boldsymbol{y}_i + \boldsymbol{\eta}_i$, where $\boldsymbol{\eta}_i \in \mathbb{R}^L$ the associated noise.

Given a set of data points $X = \{\boldsymbol{x}_i \in \mathbb{R}^L\}_{i=1}^N$, our goal is to find the subspace (affine or linear) that fits the data best. This can be carried out by minimizing the sum of squares error,

$$\min_{\boldsymbol{\mu}, \boldsymbol{U}, \{\boldsymbol{y}_i\}_{i=1}^N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i\|_2^2, \tag{2.1.11}$$

where $\boldsymbol{\mu} \in S$.

Nonetheless, this problem has not a unique solution. Firstly, there are infinitely many $\boldsymbol{\mu}$ vectors that satisfy the above equation. Suppose that $\boldsymbol{y}_i$ is shifted. Then,

$$\|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i\|_2^2 = \|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i + \boldsymbol{U}\boldsymbol{y}_0 - \boldsymbol{U}\boldsymbol{y}_0\|_2^2 = \|\boldsymbol{x}_i - \boldsymbol{\mu}^* - \boldsymbol{U}(\boldsymbol{y}_i + \boldsymbol{y}_0)\|_2^2 \tag{2.1.12}$$

where $\boldsymbol{\mu}^* = \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_0 \in S$.

To avoid this ambiguity, we set $\mathrm{E}(\boldsymbol{y}_1) = \mathrm{E}(\boldsymbol{y}_2) = ... = \mathrm{E}(\boldsymbol{y}_L) = \frac{\sum_{i=1}^N \boldsymbol{y}_i}{N} = \boldsymbol{0}$.

Secondly, there is an infinitely number of possible bases for S. For ease, we search for the orthonormal solution to the problem such $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_d$.

In the light of the above, we reformulate the optimization problem as,

$$\min_{\boldsymbol{\mu},\boldsymbol{U},\{\boldsymbol{y}_i\}_{i=1}^N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^{N} \boldsymbol{y}_i = \boldsymbol{0} \text{ and } \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_d. \tag{2.1.13}$$

From Langrange multipliers it is

$$\min_{\boldsymbol{\mu},\boldsymbol{U},\{\boldsymbol{y}_i\}_{i=1}^N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i\|_2^2 + \boldsymbol{\kappa} \sum_{i=1}^{N} \boldsymbol{y}_i + \text{trace}((\boldsymbol{I}_d - \boldsymbol{U}^\top \boldsymbol{U})\boldsymbol{\Lambda}), \tag{2.1.14}$$

where $\boldsymbol{\kappa} \in \mathbb{R}^d$ and $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^\top \in \mathbb{R}^{d \times d}$.

By taking the derivative of 2.1.14 w.r.t. $\boldsymbol{\mu}$ and setting it equal to zero, it is easy to show that

$$\hat{\boldsymbol{\mu}} = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i}{N}. \tag{2.1.15}$$

For $\boldsymbol{y}_i$ to be an extremum it should hold that

$$\begin{aligned} -2\boldsymbol{U}^\top(\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{y}_i) + \boldsymbol{\kappa} = 0 \\ -2\boldsymbol{U}^\top(\boldsymbol{x}_i - \boldsymbol{\mu}) + 2\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{y}_i + \boldsymbol{\kappa} = 0. \end{aligned} \tag{2.1.16}$$

Summing over i eq. 2.1.16, we get that $\boldsymbol{\kappa} = \boldsymbol{0}$. Thus, it is

$$\hat{\boldsymbol{y}}_i = \boldsymbol{U}^\top(\boldsymbol{x}_i - \boldsymbol{\mu}). \tag{2.1.17}$$

Combining 2.1.11 and 2.1.17 and setting $\boldsymbol{x}_i^* = \boldsymbol{x}_i - \boldsymbol{\mu}$, yields to

$$\begin{aligned} \min_{\boldsymbol{U}} \sum_{i=1}^{N} \|\boldsymbol{x}_i^* - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i^*\|_2^2 &= \min_{\boldsymbol{U}} \sum_{i=1}^{N} \|(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}_i^*\|_2^2 \\ &= \min_{\boldsymbol{U}} \sum_{i=1}^{N} \boldsymbol{x}_i^{*\top}(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)^\top(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}_i^* \\ &= \min_{\boldsymbol{U}} \sum_{i=1}^{N} \boldsymbol{x}_i^{*\top}(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}_i^* \\ &\overset{\boldsymbol{U}^\top\boldsymbol{U}=\boldsymbol{I}_d}{=} \min_{\boldsymbol{U}} \sum_{i=1}^{N} \boldsymbol{x}_i^{*\top}(\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}_i^* \\ &= \min_{\boldsymbol{U}} \text{trace}((\boldsymbol{I}_L - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{X}^*\boldsymbol{X}^{*\top}) \\ &= \min_{\boldsymbol{U}} \text{trace}(\boldsymbol{I}_L\boldsymbol{X}^*\boldsymbol{X}^{*\top} - \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{X}^*\boldsymbol{X}^{*\top}), \end{aligned} \tag{2.1.18}$$

where $\boldsymbol{x}_i^* = \boldsymbol{x}_i - \boldsymbol{\mu}$ and $\boldsymbol{X}^*$ the centered data matrix with columns the $\boldsymbol{x}_i^*$ vectors.

Observe that the first term inside trace is independent of $\boldsymbol{U}$. Additionaly, from theory $\text{trace}(\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{X}^*\boldsymbol{X}^{*\top}) = \text{trace}(\boldsymbol{U}^\top\boldsymbol{X}^*\boldsymbol{X}^{*\top}\boldsymbol{U})$. Thus, our new optimization problem becomes the following,

$$\max_{\boldsymbol{U}} \text{trace}(\boldsymbol{U}^\top\boldsymbol{X}^*\boldsymbol{X}^{*\top}\boldsymbol{U}) \quad \text{s.t.} \quad \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_d. \tag{2.1.19}$$

The Langrangian is given by the formula

$$\max_{\boldsymbol{U}} \text{trace}(\boldsymbol{U}^\top \boldsymbol{X}^* \boldsymbol{X}^{*\top} \boldsymbol{U}) + \text{trace}((\boldsymbol{I}_d - \boldsymbol{U}^\top \boldsymbol{U})\boldsymbol{\Lambda}), \tag{2.1.20}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^\top \in \mathbb{R}^{d \times d}$.

Taking the derivative of 2.1.20 w.r.t. $\boldsymbol{U}$ and set it equal to zero.

$$\begin{aligned} \boldsymbol{X}^* \boldsymbol{X}^{*\top} \boldsymbol{U} + (\boldsymbol{X}^* \boldsymbol{X}^{*\top})^\top \boldsymbol{U} - \boldsymbol{U}\boldsymbol{\Lambda}^\top - \boldsymbol{U}\boldsymbol{\Lambda} = 0 \\ \boldsymbol{X}^* \boldsymbol{X}^{*\top} \boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}. \end{aligned} \tag{2.1.21}$$

Muliplying from the right with $\boldsymbol{U}^\top$ yields to $\boldsymbol{X}^* \boldsymbol{X}^{*\top} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$.

Muliplying from the left with $\boldsymbol{U}^\top$ yields to $\boldsymbol{U}^\top \boldsymbol{X}^* \boldsymbol{X}^{*\top} \boldsymbol{U} = \boldsymbol{\Lambda}$.

Therefore, the objective function can be written as,

$$\text{trace}(\boldsymbol{U}^\top \boldsymbol{X}^* \boldsymbol{X}^{*\top} \boldsymbol{U}) = \text{trace}(\boldsymbol{\Lambda}). \tag{2.1.22}$$

Because both $\boldsymbol{X}^* \boldsymbol{X}^{*\top}$ and $\boldsymbol{\Lambda}$ are symmetric, matrix $\boldsymbol{U}$ is the orthogonal matrix of $\boldsymbol{X}^* \boldsymbol{X}^{*\top}$ with columns its d-eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix with the corresponding d-eigenvalues of $\boldsymbol{X}^* \boldsymbol{X}^{*\top}$ on the main diagonal. From eq. 2.1.22, we conclude that the objective function takes its maximum value for the matrix $\hat{\boldsymbol{U}}$, whose columns are the eigenvectors that correspond to the d-largest eigenvalues of $\boldsymbol{X}^* \boldsymbol{X}^{*\top}$. Those eigenvectors are the top d *left singular vectors* of $\boldsymbol{X}^*$ which are equivalent to those obtained by the classical PCA approach in the section 2.1.1.

2.2 Alternating Iteratively Reweighted Least Squares (AIRLS)

In the previous section (2.1.2), we show that the problem of dimensionality reduction can be seen as that of fitting a lower dimensional subspace S to the data set.

In [10] a new algorithm, called Alternating Iteratively Reweighted Least Squares (AIRLS) is proposed, whose aim is to reduce the dimension of a data set $X = \{\boldsymbol{x}_i \in \mathbb{R}^L\}_{i=1}^N$ corrupted by noise. Specifically, the algorithm tries to express a low-rank matrix $\boldsymbol{X} \in \mathbb{R}^{L\times N}$ (with columns the data vectors) as a product of two low-rank matrices $\boldsymbol{U} \in \mathbb{R}^{L\times L}$ and $\boldsymbol{Y} \in \mathbb{R}^{L\times N}$ ($\boldsymbol{X} = \boldsymbol{U}\boldsymbol{Y}$), where $\boldsymbol{U}$ will contain the basis vectors of a lower dimensional space (subspace) and $\boldsymbol{Y}$ the projections of the data points on this subspace.

On every iteration, we try to fit a subspace S of dimension lower than that of the ambient space (regularizer effect), subject to the constraint that the distance of the points from that subspace is less or equal to the variance associated with that subspace by reducing this way the number of possible solutions to the problem.

In mathematical terms, it is

$$\min_{\boldsymbol{U},\boldsymbol{Y}} \left\| \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{Y}^\top \end{bmatrix}_{(L+N)\times L} \right\|_{r,2} \quad \text{s.t.} \quad \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Y}\|_F^2 \leq \eta, \tag{2.2.1}$$

where η a small constant defining the variance of S, $\boldsymbol{X} \triangleq [\boldsymbol{x}_1 - \boldsymbol{\mu}, \boldsymbol{x}_2 - \boldsymbol{\mu}, ..., \boldsymbol{x}_N - \boldsymbol{\mu}] \in \mathbb{R}^{L\times N}$ the matrix with columns the centered data vectors and $\boldsymbol{Y} \triangleq [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_m] \in \mathbb{R}^{L\times N}$ the matrix with columns the projections of those vectors onto the subspace S.

The previous optimization problem can be equivalently written as,

$$\min_{\boldsymbol{U},\boldsymbol{Y}} \sum_{c=1}^{L} (\|\boldsymbol{u}_c\|_2^2 + \|\boldsymbol{y}_c\|_2^2)^{\frac{r}{2}} \quad \text{s.t.} \quad \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Y}\|_F^2 \leq \eta, \tag{2.2.2}$$

where $\boldsymbol{u}_c \in \mathbb{R}^{L\times 1}$ and $\boldsymbol{y}_c \in \mathbb{R}^{N\times 1}$ the column parts of the concatenated matrix, parameter r defines the $\ell_{r,2}^r$ norm (see Appendix A).

By Langrange theorem, we reformulate the problem to the following

$$\{\hat{\boldsymbol{U}}, \hat{\boldsymbol{Y}}\} = \operatorname*{argmin}_{\boldsymbol{U},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Y}\|_F^2 + \lambda_2 \sum_{c=1}^{L} (\|\boldsymbol{u}_c\|_2^2 + \|\boldsymbol{y}_c\|_2^2 + z^2)^{\frac{r}{2}}, \tag{2.2.3}$$

where λ_2 denotes the Langrange multiplier and z^2 a small positive constant that alleviates singular points i.e., points where the gradient is not continuous.

Regularizer $\lambda_2 \geq 0$ tries to "vanish" - shrink to zero - the columns of the concatenated matrix $\begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{Y}^\top \end{bmatrix}$ reducing thus their rank. Therefore, the data matrix $\boldsymbol{X}$ will be expressed as a product of two low-rank matrices.

The solution of the previous optimization problem will give the basis $\boldsymbol{U}$ of the subspace (that is not orthonormal), as well as the matrix of the projections of the N data points, $\boldsymbol{Y}$.

Unfortunately, the non-separable nature of the regularizing term prohibits us from obtaining a closed form solution. To tackle the above problem, the authors created an algorithm that follows the *Block Successive Upperbound Minimization* (BSUM) framework in order to alternatingly minimize 2.2.3.

Algorithm 1 Alternating Iteratively Reweighted Least Squares (AIRLS) Denoising Algorithm

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$, initial basis $\boldsymbol{U}_0 \in \mathbb{R}^{L \times L}$, initial projections $\boldsymbol{Y}_0 \in \mathbb{R}^{L \times N}$, regularizer λ_2, norm's parameter r, smoothing parameter z.

1: $\boldsymbol{D}_{(\boldsymbol{U}_0, \boldsymbol{Y}_0)} = \text{diag}\big((\|\boldsymbol{u}_1\|_2^2 + \|\boldsymbol{y}_1\|_2^2)^{\frac{r}{2}}, (\|\boldsymbol{u}_2\|_2^2 + \|\boldsymbol{y}_2\|_2^2)^{\frac{r}{2}}, ..., (\|\boldsymbol{u}_L\|_2^2 + \|\boldsymbol{y}_L\|_2^2)^{\frac{r}{2}}\big)$

2: **repeat**

3: $\quad \boldsymbol{U}_{k+1} = \boldsymbol{X}\boldsymbol{Y}_k^\top \big(\boldsymbol{Y}_k \boldsymbol{Y}_k^\top + \lambda_2 \boldsymbol{D}_{(\boldsymbol{U}_k, \boldsymbol{Y}_k)}\big)^{-1}$

4: $\quad \boldsymbol{Y}_{k+1} = (\boldsymbol{X}^\top \boldsymbol{U}_{k+1} \big(\boldsymbol{U}_{k+1}^\top \boldsymbol{U}_{k+1} + \lambda_2 \boldsymbol{D}_{(\boldsymbol{U}_{k+1}, \boldsymbol{Y}_k)}\big)^{-1})^\top$

5: $\quad$ (optional) Pruning

6: $\quad k = k + 1$

7: **until** Convergence

Output: $\boldsymbol{U} = \boldsymbol{U}_{k+1}$, $\boldsymbol{Y} = \boldsymbol{Y}_{k+1}$.

Optionally, a pruning procedure can be placed to take action in step 5 of AIRLS (see algorithm 1). By this way we remove the columns of the concatenated matrix that their elements are approximately zero. This step is crucial for the adaptation of the subspace dimension. Moreover, it reduces the per iteration complexity of the algorithm. Hopefully, the cardinality of the remaining columns will be equal to true dimension of the subspace. In the case where the dimension of the subspace is equal to that of the ambient space L, a good choice of λ_2 maintains the initial dimension of the fitting subspace L.

3. SUBSPACE CLUSTERING: DEFINITIONS AND METHODS

This chapter contextualizes the subspace clustering problem and focuses explicitly on the methods used in the experiments. For theorems and proofs check [23] and the respective cited papers.

3.1 Problem and challenges

Subspace clustering refers to the problem of clustering where the data are spread along various subspaces of the *ambient space*[1]. It constitutes a generalization of the problem that Principal Components Analysis - PCA and Alternating Iteratively Reweighted Least Squares - AIRLS (see chapter 2) solve. More specifically, in those methods, the goal is to estimate the subspace where all the data points live. In contrast, in subspace clustering the aim is to fit one subspace for each cluster of data points.

Assume that we want to partitioning a set of points $X = \{\boldsymbol{x}_i \in \mathbb{R}^L\}_{i=1}^N$ into K clusters $\{C_j\}_{j=1}^K$ each one associated with a subspace S_j, defined as subspaces

$$\{S_j\}_{j=1}^K = \{\boldsymbol{x}_i \in \mathbb{R}^L : \boldsymbol{x}_i = \boldsymbol{\mu}_j + \boldsymbol{U}_j \boldsymbol{y}_{ij}, \boldsymbol{\mu}_j \in \mathbb{R}^L, \boldsymbol{U}_j \in \mathbb{R}^{L \times d_j}, \boldsymbol{y}_{ij} \in \mathbb{R}^{d_j}\}_{j=1}^K \tag{3.1.1}$$

of dimension $\{d_j\}_{j=1}^K \equiv \{\dim(S_j)\}_{j=1}^K$.

The displacement vectors $\{\boldsymbol{\mu}_j\}_{j=1}^K$, the bases $\{\boldsymbol{U}_j\}_{j=1}^K$ that define the subspaces, the projections $\{\boldsymbol{y}_{ij}\}_{i=1}^N j = 1, ..., K$, of the data points to the subspaces, the dimensions of the associated subspaces $\{d_j\}_{j=1}^K$, as well as the *data segmentation* (the assignment of the data points to the clusters) are to be determined. Moreover, notice that for $K = 1$, equation 3.1.1 describes the (single) subspace where all the points of the data set lie and can be estimated by PCA or AIRLS approach presented in chapter 2.

The degree of association of a point $\boldsymbol{x}_i$ with a cluster C_j is quantified via a compatibility degree w_{ij}. Actually, compatibility degrees define the segmentation of the data set. As mentioned earlier, all the existing subspace clustering algorithms adopt the hard philosophy for belongness (each point belongs exclusively to a single cluster); in mathematical term that is, $w_{ij} \in \{0, 1\}$.

[1] If S is a subspace of a vector space V, we call V the *parent space* or *ambient space* of S. Informally, when our data form subspaces their corresponding data space is called ambient space.

Focusing on one cluster C_j, it holds that

$$d_j \leq \sum_{i=1}^{N} w_{ij} < N. \tag{3.1.2}$$

To put it differently, a subspace S_j is compatible with at least d_j points (or $d_j + 1$ for affine subspaces), but it can not be highly compatible with the entire data set.

For the hard membership function, the full set of constraints for w_{ij}'s is as follows

$$\begin{aligned} w_{ij} \in \{0,1\}, i = 1, ..., N, j = 1, ..., K. \\ \sum_{j=1}^{K} w_{ij} = 1, i = 1, ..., N. \end{aligned} \tag{3.1.3}$$

In this case it is $w_{ij} \in \{0, 1\}$; that is, a point strictly belongs exclusively to a single cluster.

On the other hand, in the possibilistic setting, the only constraint for w_{ij}'s is:

$$w_{ij} \in (0, 1]. \tag{3.1.4}$$

Observe that in the possibilistic case there is no restriction about the sum of the compatibility degrees associated to a certain data point. This means that a point can be simultaneously highly or almost no compatible with several clusters. This property makes the possibilistic clustering algorithms highly robust to outliers. Note that w_{ij}'s can be arranged so that they form an $N \times K$ matrix $\boldsymbol{W}$, that is $\boldsymbol{W} = [w_{ij}]_{i=1,...,N,j=1,...,k}$.

As discussed in [22], in subspace clustering we face the following challenges:

1. Both the parameters of the subspaces and the segmentation of the data are considered unknown. If the segmentation were known, we would simply apply PCA to each group of points to estimate the associated subspaces. On the other hand, if the subspaces were known, we would associate each point with its closest subspace [2].

2. The distribution of the data around the subspaces is unknown. For instance, in SSC algorithm [8] we have theoritical guarantees about its effectiveness, when the data are uniformly distributed and the clusters are sufficiently separated. Nevertheless, in some cases the subspaces are degenerate with points in their intersections.

[2] In the rest of this work, we assume each subspace corresponds to a single cluster. Thus, knowledge of the subspace implicitly defines the associated cluster.

3. Subspaces can be joint - disjoint or dependent - independent (see definitions A.2.5,A.2.6 and A.2.12-A.2.14). In general, the easiest case in subspace clustering is when the subspaces are considered disjoint or independent, since the subspaces in those cases do not have points in the intersection (see figure 2).

4. Data can be corrupted by noise, outliers, and may exhibit missing entries. This calls for robust algorithms under the concept of subspace clustering.

5. In PCA, which corresponds to the $K = 1$ case, the only parameter for estimation is the dimension of the subspace. However, in subspace clustering, we need a model criterion that benefits models with more than one clusters and low dimensions.

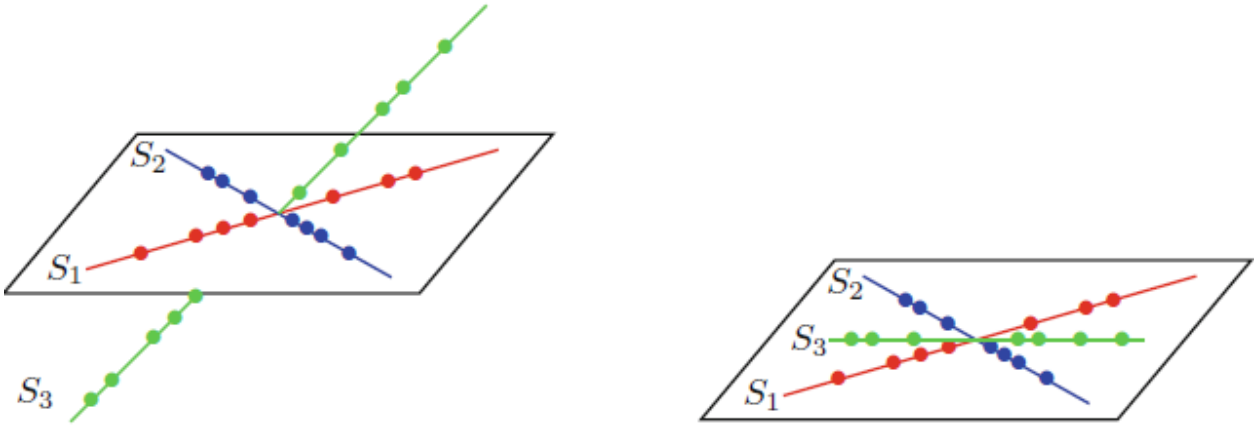


Figure 2: Independent and disjoint subspaces

(a) A set of three independent 1-dimensional subspaces. (b) A set of three disjoint 1-dimensional subspaces. Observe that the first set, in contrast with the second, spannes $\mathbb{R}^3$. In both images, every pair of subspaces intersects only at the origin. Source: [23]

The following sections present some popular subspace clustering methods, which will be used to test the performance of SAP K-subspaces clustering algorithm. Specifically, in section 3.2 we present the K-subspaces algorithm, a cost function based clustering algorithm that is similar to ours but it is non adaptive (in terms of the number of clusters) and it follows the hard philosophy. In section 3.3 we present RANSAC, a robust to outliers agorithm that, as K-subspaces, has the ability to exclude outliers from the clustering procedure. Lastly, in sections 3.4 and 3.5 two state of the art algorithms are presented: Spectral Local Best-Fit Flats (SLBF) and Sparse Subspace Clustering (SSC). Both of them fall into the category of spectral based methods which currently includes the most powerful subspace clustering algorithms.

3.2 K-Subspaces clustering

K-subspaces [1, 21] is a *statistical and iterative method* for subspace clustering that generalizes the classical K-means algorithm [17]. The data instead of forming clusters that spread along all dimensions of the data space, they form clusters that lie along the linear or affine subspaces of the data space. Historically, it conducts an improved version of the K-planes algorithm [5] which was suitable only for hyperplanes detection.

Similarly to K-means, in the K-subspaces clustering algorithm, iteratively we try to minimize the sum of the distances of the data points from their closest cluster representatives. This is carried out by solving the following optimization problem:

$$\begin{aligned} &\min_{\{\boldsymbol{\mu}_j\},\{\boldsymbol{U}_j\},\{\boldsymbol{W}\}} \sum_{j=1}^{K}\sum_{i=1}^{N} w_{ij}\ \text{dist}(\boldsymbol{x}_i, S_j)^2 \\ &\text{s.t. } \boldsymbol{U}_j\boldsymbol{U}_j^\top = \boldsymbol{I}_L, w_{ij} \in \{0,1\}, \sum_{j=1}^{K} w_{ij} = 1 \text{ for } i = 1, ..., N \text{ and } j = 1, ..., K, \end{aligned} \tag{3.2.1}$$

where $\boldsymbol{U}_j$ is the matrix containing the basis vectors of the subspace S_j. The distance of a point $\boldsymbol{x}_i$ from the cluster C_j is defined as the distance of $\boldsymbol{x}_i$ from the associated subspace S_j and is given by the formula,

$$\text{dist}(\boldsymbol{x}_i, S_j)^2 = \|\boldsymbol{x}_i - \boldsymbol{\mu}_j - \boldsymbol{U}_j\boldsymbol{y}_{ij}\|_2^2 \quad \overset{\boldsymbol{U}_j^\top(\boldsymbol{x}_i-\boldsymbol{\mu}_j)=\boldsymbol{y}_{ij}}{=\!=} \quad \|(\boldsymbol{I}_L - \boldsymbol{U}_j\boldsymbol{U}_j^\top)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)\|_2^2. \tag{3.2.2}$$

The first constraint in eq. 3.2.1, means that we seek to find an orthogonal basis of the subspace. The degree of compatibility w_{ij} is equal to 1 when $\boldsymbol{x}_i$ belongs to C_j. Lastly, the constraint about the sum of the degrees means that every point belongs only to one cluster (hard clustering).

As its ancestor (K-means), the K-subspaces algorithm is of iterative nature. For the algorithm to start, one must initialize the orthonormal bases $\{\boldsymbol{U}_j\}_{j=1}^K$, their corresponding dimension $\{d_j\}_{j=1}^K$ and the associated displacement vectors $\{\boldsymbol{\mu}_j\}_{j=1}^K$. The main part of the algorithm can be divided into two stages: segmentation (definition of clusters) and estimation (of the associated subspaces - orthonormal basis, projections and displacement vectors). More specifically, fixing $\{\boldsymbol{U}_j\}_{j=1}^K, \{\boldsymbol{\mu}_j\}_{j=1}^K$, we assign the data points to their closest clusters (in terms of their distance of a point from a subspace). Then, fixing the partition of the data, we re-estimate (update) the parameters $\{\boldsymbol{U}_j\}_{j=1}^K, \{\boldsymbol{\mu}_j\}_{j=1}^K$. Repeat until a termination criterion is met.

Algorithm 2 K-Subspaces

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$, number of subspaces K, subspaces dimension $\{d_j\}_{j=1}^{K}$

1: **Initialization:** Initialize the bases $\{\boldsymbol{U}_j\}_{j=1}^{K}(\boldsymbol{U}_j \in \mathbb{R}^{L \times d_j})$ and the displacement vectors $\{\boldsymbol{\mu}_j\}_{j=1}^{K}(\boldsymbol{\mu}_j \in \mathbb{R}^{L})$.

2: **repeat**

3: **for** $j = 1 : K$ **do**

4: **for** $i = 1 : N$ **do** ▷ Segmentation part

5: $$w_{ij} = \begin{cases} 1, & \text{if } j = \underset{l=1,\ldots,K}{\operatorname{argmin}} \|(\boldsymbol{I}_L - \boldsymbol{U}_l \boldsymbol{U}_l^{\top})(\boldsymbol{x}_i - \boldsymbol{\mu}_l)\|_2^2 \\ 0, & \text{else} \end{cases}$$

6: **end for**

7: $\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} w_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{N} w_{ij}}$ ▷ Estimation part

8: $\boldsymbol{U_j} =$ top d_j eigenvectors of $\sum_{i=1}^{N} w_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^{\top}$

9: $\boldsymbol{Y}_j = [\boldsymbol{U}_j^{\top}(\boldsymbol{x}_1 - \boldsymbol{\mu}_j),\ldots,\boldsymbol{U}_j^{\top}(\boldsymbol{x}_N - \boldsymbol{\mu}_j)]$

10: **end for**

11: **until** convergence

Output: Compatibility degree matrix $\boldsymbol{W}$, $\{\boldsymbol{U}_j, \boldsymbol{Y}_j, \boldsymbol{\mu}_j\}_{j=1}^{K}$.

Observe that for $K = 1$ the problem becomes that of geometric PCA, while for $d_j = 0$, there are no bases and projections ($\{\boldsymbol{U}\}_{j=1}^{K}$, $\boldsymbol{Y}_j$ respectively), thus the algorithm becomes that of K-means. The latter occurs when the data instead of enlogated flat-shaped clusters form compact and hyperspherically-shaped clusters.

The algorithm gradually evolves as it tries to come closer to a local minimum of the cost function in 3.2.1. Furthemore, It is proved that the algorithm will always converge to a local minimum [23]. However, the ”quality” of the convergence depends on the initialization. In addition, K-subspaces has the ability of performing well at small noise levels. As someone can easily observe, the major problem with K-subspaces is that the number of the flats and their dimension have to be known a priori, a case that is barely met in practice. Moreover, the l_2-norm makes the algorithm sensitive to outliers.

3.3 RANdom SAmple Consensus (RANSAC)

RANSAC belongs to the category of the *statistical methods* of subspace clustering. It was introduced in [9] in order to smooth and interpret data sets that contain outliers. The algorithm is suitable for both linear and affine subspaces. In its original form, RANSAC can be used for the identification of a single subspace associated with the whole data set.

To define a linear d-dimensional subspace, d independent vectors are needed. Subsequently, the algorithm iteratively picks at least d random data points and estimates the subspace $\hat{S}$ via PCA. Based on that model, if any of the residuals of the entire data set is less than or equal to a threshold η that particular point is considered as an *inlier* (consensus test). The process is then repeated until the cardinality of the inliers is equal or over a user-defined number N_{min}. The rest of the data are marked as outliers.

Algorithm 3 RANdom SAmple Consensus (RANSAC)

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$, subspace dimension d, maximum number of iterations k, residuals threshold η, cardinality of inliers threshold N_{min}.

$i = 1$
repeat
  $X_i \leftarrow d$ points at random
  $\hat{S}_i \leftarrow$ PCA(X_i)
  $X_{inliers} \leftarrow \{\boldsymbol{x} \in X : \text{dist}(\boldsymbol{x}, \hat{S}_i) \leq \eta\}$
  if $|X_{inliers}| \geq N_{min}$ **then**
    $i = k$
  else
    $i = i + 1$
  end if
until $i = k$
$\hat{S} \leftarrow$ PCA($X_{inliers}$)

Output: $\hat{S}$, $X_{inliers} \in \mathbb{R}^{L \times |inliers|}$.

Nonetheless, the goal in the subspace clustering is to estimate K subspaces. Thus, in order to utilize RANSAC in this framework, we run RANSAC and we repeat until no outliers are remained. The procedure terminates returning the estimation of each inliers' set subspace (cluster) and the reassignment of every point to its closest cluster. The process described is known as **RANSAC-on-Subspaces** and it prevails over RANSAC-on-Union which estimates all the subspaces at once but with a high computational cost [25]. Notice that by using homogeneous coordinates (definition A.2.10), the problem can be generalized to that of K affine subspaces.

Because of its sequential nature, RANSAC-on-Subspaces can handle both dependent and independent subspaces (definitions A.2.5 and A.2.13). It is extremely robust to outliers and the number of subspaces K does not have to be known a priori. However, the choice of η and N_{min} plays an important role in the determination of K. Its main disadvantage is that the probability of getting d inliers reduces exponentially with the number of subspaces. Consequently, the number of maximum iterations has to become larger as the number of subspaces and their dimension increases. Lastly, the dimensions of each subspace have to be known and equal.

In cases where the dimensions differ, one can start the estimation of the subspaces in an increasing or decreasing dimension order. Nevertheless, as discussed in [25], this may cause some serious problems. If one begins by estimating the subspace with the highest dimension, it is expected that data which belong to subspaces of lower dimension will be assigned to it (model over-fit). On the other hand, if one begins with that of the lowest dimension, the first estimated subspace is quite possible to be assigned to data that belong to intersections of subspaces or data that belong to the subspace of the highest dimension. All these side effects get empowered by the algorithm's hard logic and the fact that in each iteration inliers are removed.

3.4 Spectral Local Best-fit Flats (SLBF)

SLBF [26] is a *spectral-based method* for subspace clustering [3]. The ultimate challenge with this kind of methods is to define a good affinity matrix. In subspace clustering, two points which belong to different clusters can be really close to each other if they both lie near intersections of subspaces. The opposite is likely to happen as well, two points that belong to the same group can be far from each other if that group forms an elongated flat structure.

To cope with this problem, SLBF uses as similarity metric the Gaussian kernel function but with point to subspace distance instead of point to point. Firstly, the algorithm estimates for each point (center) $\boldsymbol{x}_i$ a set of neighbors X_i and afterwards using that it defines a subspace $\hat{S}_i$. To define a neighborhood, for a given center $\boldsymbol{x}_i$ algorithm 4 gradually increases the neighborhoods' size till it finds the first local minimum of the average errors.

The average error of a set X_i with center the point $\boldsymbol{x}_0$, is given by the formula:

$$\beta_2(X_i) = \sqrt{\frac{\|\boldsymbol{X}_i - \boldsymbol{H}_j \boldsymbol{X}_i\|_F^2}{|X_i|(\max\limits_{\boldsymbol{x}_i \in X_i} \|\boldsymbol{x}_i - \boldsymbol{x}_0\|_2)^2}}, \tag{3.4.1}$$

where $\boldsymbol{X}_i$ the matrix with columns the elements of X_i, $|X_i|$ the cardinality of X_i and $\boldsymbol{H}_j = \boldsymbol{U}_j(\boldsymbol{U}_j^\top \boldsymbol{U}_j)^{-1}\boldsymbol{U}_j^\top$ the corresponding hat matrix of the model[4].

For the formation of the affinity matrix $\boldsymbol{A}$, the following equations are used:

$$\alpha_{ij} = \mathsf{exp}(-d_{i,j}/2\sigma_i^2) + \mathsf{exp}(-d_{i,j}/2\sigma_j^2) \tag{3.4.2}$$

where

$$d_{i,j} = \sqrt{\mathsf{dist}(\boldsymbol{x}_i, \hat{S}_i), \mathsf{dist}(\boldsymbol{x}_j, \hat{S}_j)} \tag{3.4.3}$$

and

$$\sigma_i = \sqrt{\sum_{\boldsymbol{x} \in X_i} \mathsf{dist}(\boldsymbol{x}, \hat{S})}. \tag{3.4.4}$$

Parameter σ_i measures how well the neighborhood X_i fits its closest subspace $\hat{S}$, where $\{\hat{S}\} = \underset{\hat{S}_h}{\mathsf{argmin}}\, \mathsf{dist}(\boldsymbol{x}, \hat{S}_h)$, $h = 1, ..., N$. To compute the point to subspace distance, the formula 3.2.2 is used.

[3] For a small introduction in spectral clustering see Appendix B.

[4] Here, as model we mean the subspace S_i that we fit to the set X_i.

The process continuous with the computation of the diagonal matrix and the normalization of the affinity. Finally, the produced matrix is factorized and K-means is applied to the rows of the matrix composed by the top k eigenvectors [5] of the normalized affinity multiplied by the corresponding eigenvalue matrix.

Note that K-means takes as an input parameter the number of the clusters which is equal to the cardinality of the chosen eigenvectors. Perhaps, one may overcome this issue by using *adaptive compact clustering algorithms* or by using K-means combined with *model selection techniques* in order to estimate more reliably the number of clusters.

SLBF is a *local spectral method*, because it constructs the affinity based on neighborhooding information around the data points. Other methods in the literature, called *global spectral methods*, like Spectral Curvature Clustering (SCC) [6], use information from the entire data set to compute the affinity. However, the complexity of the latest rises as the dimension and the number of clusters increases.

Algorithm 4 Neighborhood Size Selection for HLM by Randomized Local Best Fit Flats

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L\times N}$, a point $\boldsymbol{x}_0 \in \mathbb{R}^L$, start size S, step size T, mean shifts parameters (optional): κ, m.

1: (optional) Update $\boldsymbol{x}_0$ as the center of its κ-nearest neighbors while repeating m times
2: $k = -1$
3: **repeat**
4: $\quad k = k + 1$
5: $\quad$ Let $\boldsymbol{X}_k$ be the data matrix with the $S + kT$ nearest neighbors of $\boldsymbol{x}_0$ as columns
6: $\quad \hat{S} \leftarrow$ PCA($\boldsymbol{X}_k$) ▷ best fit flat
7: $\quad \beta_2(k) \leftarrow \beta_2(X_k)$ according to 3.4.1
8: **until** $k > 1$ and $\beta_2(k-1) <$ min$\{\beta_2(k-2), \beta_2(k)\}$ ▷ if there is no noise, $k >= 1$

Output: $\boldsymbol{X}_{k-1}$.

[5] The top k eigenvectors correspond to the k largest eigenvalues.

Algorithm 5 Spectral Local Best-Fit Flats (SLBF)

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$, the parameters used in algorithm 4.

1: For all data $\{\boldsymbol{x}_i\}_{i=1}^{N}$, run algorithm 4 and get $\hat{S}_i$

2: Construct the $N \times N$ affinity matrix $\boldsymbol{A}$ using the eq. 3.4.2, 3.4.3 and 3.4.4

3: Construct the $N \times N$ diagonal matrix $\boldsymbol{D} = \text{diag}(d_1, d_2, ..., d_N)$, where $d_i = \sum_{i=1}^{N} \alpha_{ij}$

4: Normalize $\boldsymbol{A}$ by setting $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ and factorize it

5: Construct the $N \times k$ matrix $\boldsymbol{U}$ with columns the top k eigenvectors of $\tilde{\boldsymbol{A}}$ and the $k \times k$ diagonal matrix $\boldsymbol{\Lambda}$ that holds the corresponding eigenvalues on its diagonal

6: Run K-means over the rows of $\boldsymbol{U}\boldsymbol{\Lambda}^{1/2}$

Output: Compatibility degree matrix $\boldsymbol{W}$.

An advantage of the SLBF algorithm is that the adopted similarity measure (eq. 3.4.2), allows the handling of both affine and linear subspaces, due to its angle free definition. Another key advantage of the algorithm, is that it determines the size of the neighborhood automatically via algorithm 4. Clearly, since the algorithm works with neighbors, outliers are expected to be rejected, as they lie far from the region where the majority of the points lies.

Nonetheless, the neighborhood data do not always belong to the same subspace. For instance, the neighbors of a point in intersections of subspaces, is likely to belong to more than one group. The number of subspaces has to be known. Additionaly, the dimension of the subspaces has to be known a priori and equal for all subspaces, as a subspace is fitted to each neighborhood of every point. It goes without saying that this is another drawback of SLBF, since PCA is performed several times until the convergence of algorithm 4.

3.5 Sparse Subspace Clustering (SSC)

Until now, we have seen iterative, statistical and spectral methods for subspace clustering. The algorithm presented in this section belongs to the class of the *self-expressive methods* of subspace clustering. Those methods provide up to now one of the most effective solution to the subspace clustering problem [23].

Self-expressive methods where developed in order to overcome the issues of both local and global spectral methods. Furthemore, they provide theoritical guarantees that ensure reliable results. They are divided into two categories. Those who are based on *sparse representation techniques* and those who are based on *low-rank representation techniques*. SSC [8] is a member of the first category.

3.5.1 Uncorrupted data

In this subsection we examine the case of the uncorrupted data.

The main idea behind the algorithm is that every point $\boldsymbol{x}_i$ in a union of K subspaces $S_1 \cup S_2 \cup ... \cup S_K$ can be written as a linear or affine combination of other points in the data set. This is called *self-expressiveness property*. For now, assume that those subspaces are linear. To express each point as a linear combination of other points, one has to solve for every point the following linear system:

$$\underset{L\times 1}{\boldsymbol{x}_i} = \underset{L\times N}{\boldsymbol{X}}\underset{N\times 1}{\boldsymbol{c}_i} = \boldsymbol{x}_1 c_{i1} + \boldsymbol{x}_2 c_{i2} + ... + \boldsymbol{x}_N c_{iN}, \text{ where } \boldsymbol{c}_i \triangleq [c_{i1}, c_{i2}, ..., c_{iN}]^\top \text{ s.t. } c_{ii} = 0, i = 1, ..., N. \tag{3.5.1}$$

Vector c_i holds values that allow us, multiplied with the data matrix $\boldsymbol{X}$, to produce a linear combination that constructs $\boldsymbol{x}_i$. The constraint $c_{ii} = 0$ prohibits us from writing $\boldsymbol{x}_i$ as a linear combination of itself. Matrix $\boldsymbol{X}$ can be seen as a dictionary used to construct the data from the data themselves.

Assume the case where all the data come from one subspace of dimension d. We need to have at least d data points to define that subspace. Most of the times, it is true that $N > d$. Thus, we have that rank$(\boldsymbol{X})$ = rank$(\boldsymbol{X}|\boldsymbol{x}_i)$ and $N -$ rank$(\boldsymbol{X}) \neq 0$, where rank$(\boldsymbol{X}) = d$. By generalizing this to the problem of multiple subspaces, we have infinitely many solutions for the eq. 3.5.1.

Among all the solutions of 3.5.1, there exist one sparse solution $\boldsymbol{c}_i$, whose nonzero entries correspond to data coming from the same subspace as $\boldsymbol{x}_i$. Ideally, the cardinality of those nonzero entries is equal to the dimension of the corresponding subspace. In that case, $\boldsymbol{c}_i$ is said to be *subspace preserving* or the *subspace-sparse solution* of 3.5.1. Nevertheless, any sparse solution is not necessarily subspace preserving.

Consequently, we search for a solution $\boldsymbol{c}_i$ with restriction over the number of its nonzero entries. Thus, the problem becomes that of

$$\text{min } \|\boldsymbol{c}_i\|_0 \quad \text{s.t.} \quad \boldsymbol{x}_i = \boldsymbol{X}\boldsymbol{c}_i,\ c_{ii} = 0. \tag{3.5.2}$$

However, this problem is in general NP-hard [2]. Therefore, to restrict the number of the possible solutions and impose sparsity, the authors in [8] propose the solution of the following convex relaxation optimization problem:

$$\text{min } \|\boldsymbol{c}_i\|_1 \quad \text{s.t.} \quad \boldsymbol{x}_i = \boldsymbol{X}\boldsymbol{c}_i,\ c_{ii} = 0. \tag{3.5.3}$$

Considering simultaneously the problem 3.5.3 for all data points, we have in matrix notation the following,

$$\text{min } \|\boldsymbol{C}\|_1 \quad \text{s.t.} \quad \boldsymbol{X} = \boldsymbol{X}\boldsymbol{C},\ \text{diag}(\boldsymbol{C}) = 0, \tag{3.5.4}$$

where $\boldsymbol{C} \triangleq [\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_N] \in \mathbb{R}^{N\times N}$ is the matrix with columns the sparse vectors that reconstruct each column of $\boldsymbol{X}$ respectively. The last constraint, where $\text{diag}(\boldsymbol{C}) \in \mathbb{R}^N$, is analogous to $c_{ii} = 0$ in 3.5.3 but for every point-column of $\boldsymbol{X}$.

Solving 3.5.4 is the first step of SSC. Remember that the data in subspace clustering form (in general) enlogated flat-shaped clusters. Hence, optionally, the columns of $\boldsymbol{C}$ are normalized. This eliminates scale differences between the largest edge weights since spectral clustering tends to focuse on the largest connections of the graph. Large edge weights are created when a point with small euclidean norm is matched with other points that have way larger euclidean norms and vice versa (euclidean norms of data vectors).

The algorithm proceeds by creating the similarity graph of the data set. The affinity matrix is given by the formula $\boldsymbol{A} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$, to ensure that $\boldsymbol{A}$ is symmetric and positive, where $|C|$ is C with normalized columns. By this, we eliminate cases where a point $\boldsymbol{x}_i$ can be chosen among others to reconstruct a point $\boldsymbol{x}_q$ but the opposite does not hold. Ideally, it is expected that a point would be similar only to points from the same subspace. Likewise, it is expected that vertices corresponding to points from different subspaces are not connected by any edge.

Finally, the normalized Laplacian matrix is computed (eq. 3.1.2). It follows, its factorization and the application of K-means to the rows of the matrix composed by the bottom k normalized eigenvectors which correspond to the k smallest eigenvalues of the Laplacian.

Let as now consider the affine case. Using the homogeneous coordinates of a data set (definition A.2.10), we are able to tackle the problem of affine subspace clustering. Roughly, one can describe the homogeneous coordinates as a mathematical tool that allows us to handle the projective space, the space that any parallel affine subspaces can meet at one point in infinity. Based on this, any d-dimensional affine subspace S in $\mathbb{R}^L$ can be described as a $(d+1)$-dimensional linear subspace in $\mathbb{R}^{L+1}$ that includes S and the origin (see figure 3). Thus, to fit an affine subspace $d+1$ points are needed, where d the dimension of the affine subspace.

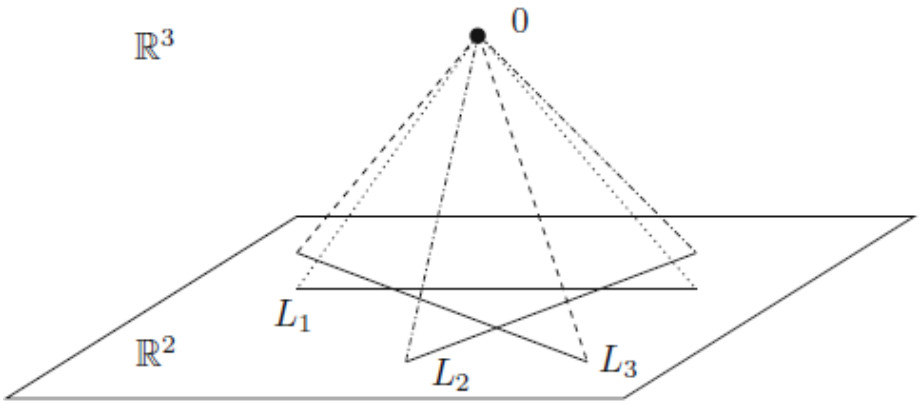


Figure 3: Homogeneous representation

Three affine 1-dimensional subspaces represented by three linear 2-dimensional subspaces. Source: [23]

Nonetheless, this way of representing affine subspaces may lead to indistinguishability between the subspaces as their in-between dimension may increase. To illustrate this, as written in [23], imagine the linear subspaces that correspond to the affine lines $x = 1$ and $x = -1$. They fall into one single subspace.

We have that any point $\boldsymbol{x}_i$ of an affine subspace $S \subset \mathbb{R}^L$ with $d < L$ can be written as,

$$\boldsymbol{x}_i = \boldsymbol{x}_0 c_{i0} + \boldsymbol{x}_1 c_{i1} + ... + \boldsymbol{x}_d c_{id}, \quad \sum_{j=0,\, j\neq i}^{d} c_{ij} = 1, \tag{3.5.5}$$

where $\{\boldsymbol{x}_i\}_{i=0}^{d}$ an affinely independent set (definition A.2.12).

Equivalently with 3.5.1, to express each entry as an affine combination of other points, one has to solve for every point the following linear system:

$$\underset{L\times 1}{\boldsymbol{x}_i} = \underset{L\times NN\times 1}{\boldsymbol{X}\ \boldsymbol{c}_i} \quad \text{s.t.} \quad \underset{1\times NN\times 1}{\mathbf{1}^\top\ \boldsymbol{c}_i} = 1,\ c_{ii} = 0. \tag{3.5.6}$$

After reformulating the problem, one observes two differences. Firstly, here the subspace-sparse solution has $d+1$ nonzero entries and secondly, there is an additional constraint about the scalars that forces their sum to be equal to one.

Finally, the optimization problem becomes,

$$\min \|\boldsymbol{C}\|_1 \quad \text{s.t.} \quad \boldsymbol{X} = \boldsymbol{X}\boldsymbol{C},\ \mathbf{1}^\top\boldsymbol{C} = \mathbf{1}^\top,\ \text{diag}(\boldsymbol{C}) = 0. \tag{3.5.7}$$

Convex programming tools that solve this optimization program can be found in [3, 4, 12].

Algorithm 6 Sparse Subspace Clustering (SSC) for Uncorrupted Data

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L\times N}$.

1: Solve the optimization program 3.5.4 or 3.5.7 for linear or affine subspaces accordingly and get $\boldsymbol{C}$
2: (Optional) Normalize the columns of $\boldsymbol{C}$ as $\boldsymbol{c}_i \leftarrow \frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|_\infty}$
3: Create a weighted similarity graph of the data set with affinity matrix $\boldsymbol{A} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$
4: Construct the $N \times N$ normalized Laplacian matrix via eq. B.0.1 and factorize it
5: Construct the $N \times k$ matrix $\boldsymbol{U}$ with columns the bottom k normalized eigenvectors
6: Run K-means over the rows of $\boldsymbol{U}$

Output: Compatibility degree matrix $\boldsymbol{W}$.

A significant advantage of SSC is that it guarantees a subspace preserving solution when uniformly distributed subspaces are: independent, disjoint or even random (for the relevant theorems see [23]). Under those conditions, there will be no connections between the points-edges of different subspaces. Thus, the number of the subspaces is equal to that of the connected components in the similarity graph. However, note that those conditions are not always met in practice. Furthemore, the algorithm is suitable for both affine and linear subspaces and most importantly their dimension does not have to be known a priori.

Unlike SLBF and other local spectral methods, SSC does not need to take points around the neighborhood of an entry to construct the affinity matrix. Assuming dependency between the clusters, in most of the cases it is likely to get a closer to the truth result, since the points that lie near intersections of subspaces are likely to be assigned to points of the same subspace-group.

The SSC rationale is extended to the concept of corrupted data as well (outliers, noise, large errors and missing data).

3.5.2 Corrupted data

In this subsection we study SSC in the precence of outliers and noise.

Typically, when the data contain outliers the first step is to separate outliers from inliers and continue by grouping the inliers set. The data set can be described as,

$$\boldsymbol{X} = [\boldsymbol{X}_{inliers}, \boldsymbol{X}_{outliers}]\boldsymbol{\Gamma}, \tag{3.5.8}$$

where $\boldsymbol{\Gamma}$ a permutation matrix. Our goal is to detect the points-columns of $\boldsymbol{X}_{outliers}$.

To do that a similar philosophy to that of RANSAC (presented in 3.3) is considered [20]. After solving the optimization program 3.5.4 or 3.5.7 and given a threshold the algorithm tests if a point $\boldsymbol{x}_i$ is an outlier or not. In the former case, that point is removed from the data set. This can be done by checking the sparseness of its corresponding sparset solution $\boldsymbol{c}_i$. Iteratively, for each point the following test is applied,

$$\|\boldsymbol{c}_i\|_1 > \theta(\gamma)\sqrt{L}, \tag{3.5.9}$$

where $\gamma = \frac{N-1}{L}$ the density of the data inside the ambient space $\mathbb{R}^L$ and θ a threshold ratio function with

$$\theta(\gamma) = \begin{cases} \sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{\gamma}}, & 1 \leq \gamma \leq e \\ \sqrt{\frac{2}{\pi e}}\frac{1}{\log\gamma}, & \gamma \geq e. \end{cases} \tag{3.5.10}$$

With e we denote the base of natural logarithms. If $\boldsymbol{x}_i$ satisfies 3.5.9, it is considered an outlier. Reliable results are guaranteed in case of uniformly distributed data. Apart from the density, no other assumption about the nature of the subspaces has to be made.

Assume now that the data points are corrupted by additive noise. Then, each point can be written as,

$$\boldsymbol{x}_i = \boldsymbol{x}_i^* + \boldsymbol{\eta}_i^*, \tag{3.5.11}$$

where $\boldsymbol{x}_i^*$ the noiseless version of $\boldsymbol{x}_i$ and $\boldsymbol{\eta}_i^*$ the noise.

However, using the self-expressiveness property given in 3.5.1, we have that

$$\boldsymbol{x}_i^* = \sum_{j=1}^{N} \boldsymbol{x}_j^* c_{ij}. \tag{3.5.12}$$

By combining 3.5.11 with 3.5.12, each point can be written as,

$$\boldsymbol{x}_i = \boldsymbol{X}\boldsymbol{c}_i + \boldsymbol{\eta}_i, \quad \text{where} \quad \boldsymbol{\eta}_i \triangleq \boldsymbol{\eta}_i^* - \sum_{j=1}^{N} \boldsymbol{\eta}_j^* c_{ij} \in \mathbb{R}^L. \tag{3.5.13}$$

Hence, the problem is reformulated to that of finding the sparsest solutions of

$$\min \|\boldsymbol{C}\|_1 + \frac{\lambda}{2}\|\boldsymbol{H}\|_F^2 \quad \text{s.t.} \quad \text{diag}(\boldsymbol{C}) = 0, \tag{3.5.14}$$

where $\boldsymbol{H} \triangleq [\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, ..., \boldsymbol{\eta}_N] = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{C} \in \mathbb{R}^{L \times N}$, $\boldsymbol{X}$ the data matrix whose columns are the noiseless versions of $\boldsymbol{x}_i$'s, $\|\|_F$ the Frobenius norm and λ the regularization parameter.

The Frobenius norm forces the entries of $\boldsymbol{H}$ to small values. Keep in mind that if the data come from affine subspaces, we should add to the problem the constraint $\mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top$. Observe that in contrast with eq. 3.5.4, here the constraint $\boldsymbol{X} = \boldsymbol{X}\boldsymbol{C}$ is not valid. Actually, it has been replaced by the request to minimize the quantity $\|\boldsymbol{H}\|_F^2$. Thus, for a small enough λ one may get $\boldsymbol{C} = 0$. According to the authors in [8] to avoid this, λ should be chosen as $\lambda = \frac{\alpha}{\kappa}$, where $\kappa = \min_i \max_{i \neq j} |\boldsymbol{x}_i^\top \boldsymbol{x}_j|$ and $\alpha > 1$. Lastly, the optimization problem is convex, thus it can be solved via any convex programming tool referenced before.

Algorithm 7 Sparse Subspace Clustering (SSC) with Outliers

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$.

1: Solve the optimization program 3.5.4 or 3.5.7 for linear or affine subspaces accordingly and get $\boldsymbol{C}$
2: For each point $\boldsymbol{x}_i$ test if 3.5.9 holds and get $\boldsymbol{X}_{inliers}$
3: Repeat step 1 by setting $\boldsymbol{X} = \boldsymbol{X}_{inliers}$
4: (optional) Normalize the columns of $\boldsymbol{C}$ as $\boldsymbol{c}_i \leftarrow \frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|_\infty}$
5: Create a weighted similarity graph of the data set with affinity matrix $\boldsymbol{A} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$
6: Construct and factorize the $N \times N$ normalized Laplacian matrix $\tilde{A}$ (see B.0.1)
7: Construct the $N \times k$ matrix $\boldsymbol{U}$ with columns the normalized eigenvectors corresponding to the k smallest eigenvalues of $\tilde{A}$
8: Run K-means over the rows of $\boldsymbol{U}$

Output: Compatibility degree matrix $\boldsymbol{W}$.

Algorithm 8 Sparse Subspace Clustering (SSC) for Noisy Data

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L \times N}$.

1: Solve the optimization program 3.5.14 and get $\boldsymbol{C}$
2: (optional) Normalize the columns of $\boldsymbol{C}$ as $\boldsymbol{c}_i \leftarrow \frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|_\infty}$
3: Create a weighted similarity graph of the data set with affinity matrix $\boldsymbol{A} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$
4: Construct and factorize the $N \times N$ normalized Laplacian matrix $\tilde{A}$ (see B.0.1)
5: Construct the $N \times k$ matrix $\boldsymbol{U}$ with columns the normalized eigenvectors corresponding to the k smallest eigenvalues of $\tilde{A}$
6: Run K-means over the rows of $\boldsymbol{U}$

Output: Compatibility degree matrix $\boldsymbol{W}$.

4. SPARSE ADAPTIVE POSSIBILISTIC K-SUBSPACES

Sparse Adaptive Possibilistic (SAP) K-subspaces belong to the same family of methods with K-subspaces (section 3.2). Both algorithms are cost function optimization-based which means that they result from the minimization of a suitably defined cost functions. In the framework of subspace clustering that cost involves the distance of the points from a subspace. However, those two methods show fundamental differences.

Surprisingly, although an extended bibliography about subspace clustering that follows the hard philosophy exists, the application of a possibilistic's nature algorithm has not yet been investigated. In this vein, we propose a subspace clustering algorithm based on the possibilistic philosophy. This means that we define the weights w_{ij} as shown in 3.1.4.

By introducing a possibilistic approach to the problem of subspace clustering, we allow the data to contribute to the definition of more than one subspace/cluster. It is expected that this way the resulting clusterings will be better than those obtained from a hard algorithm. This is because points that belong to intersections of subspaces contribute to the formation of all relatives subspaces and not to a single one of them, as it happens in the hard clustering case.

Many subspace clustering algorithms require a priori knowledge of the number of clusters K, the dimensions of the associated subspaces $\{d_j\}_{j=1}^{K}$ and their variances $\{\eta_j\}_{j=1}^{K}$. The introduced algorithm adapts all of them during its execution. In what follows, $\hat{K}$ is the initial number of clusters and $\hat{d}$ is the initial dimension of the associated subspaces, where $\hat{d} \geq \mathsf{max}_j\, d_j, j = 1, 2, ..., \hat{K}$.

As K-subspaces conducts a generalization of the K-means algorithm, SAP K-subspaces constitutes a generalization of the SPCM algorithm proposed by [24]. More specificaly, SAP K-subspaces results from the minimization of the following cost function:

$$\mathcal{C}(\{\boldsymbol{\mu}_j\}_{j=1}^{\hat{K}}, \{\boldsymbol{U}_j\}_{j=1}^{\hat{K}}, \{\boldsymbol{Y}_j\}_{j=1}^{\hat{K}}, \boldsymbol{W}) \equiv \sum_{j=1}^{\hat{K}} \Bigg[\sum_{i=1}^{N} w_{ij} \mathsf{dist}(\boldsymbol{x}_i, S_j)^2 + \eta_j \sum_{i=1}^{N} (w_{ij} \ln w_{ij} - w_{ij}) + \\ + \lambda_1 \sum_{i=1}^{N} \|\boldsymbol{w}_i\|_p^p \Bigg] \quad \text{s.t.} \quad w_{ij} \in (0, 1], \tag{4.0.1}$$

where $\|\boldsymbol{w}_i\|_p^p = \sum_{j=1}^{K} w_{ij}^p$ with $p \in (0, 1)$ and $\lambda_1 \geq 0$ the regularization parameter that controls the degree of the imposed sparsity.

In practice, η_j is an unknown rather small number, estimated by the algorithm. It is a measure of variance of the data points of the cluster C_j around its associated subspace S_j.

The distance of a point from the subspace S_j is given by the formula

$$\text{dist}(\boldsymbol{x}_i, S_j)^2 = \|\boldsymbol{x}_i - \boldsymbol{\mu}_j - \boldsymbol{U}_j \boldsymbol{y}_{ij}\|_2^2, \tag{4.0.2}$$

where $\boldsymbol{U}_j \in \mathbb{R}^{L \times \hat{d}}$ the basis of S_j with initial dimension $\hat{d}$.

A rational assumption that is adopted in this framework is that a point $\boldsymbol{x}_i$ is compatible with only a few subspaces S_j. This is reflected to the entries of $\boldsymbol{w}_i$ with those (few) corresponding (to some degree) to those subspaces being positive, while the rest are zero. Sparsity refers to the fact that only a few entries of $\boldsymbol{w}_i$ are non-zero. In the present framework, when $\text{dist}(\boldsymbol{x}_i, S_j)^2$ is higher than the variance η_j of S_j, we want w_{ij} to be zero. By this way, the only points that contribute to the estimation of a subspace are those lie very close to it.

Note that for the point representative case, where instead of S_j we have a point, the cost function is the same with that of SPCM. If in addition to that we set $\lambda_1 = 0$, the cost function becomes that of PCM [13]. The second term of the cost function alone, holds the values of w_{ij} in the interval $(0, 1]$. Nevertheless, here we have a third term of w_{ij}'s that needs to be investigated further.

Due to the fact that the parameters w_{ij} for a certain $\boldsymbol{x}_i$ are not interrelated, the problem breaks down into K independent subproblems, one for each cluster. Thus, we have to solve one optimization problem for each cluster C_j:

$$\begin{aligned} &\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, \{w_{ij}\}_{i=1}^N) \equiv \sum_{i=1}^N w_{ij} \text{dist}(\boldsymbol{x}_i, S_j)^2 + \eta_j \sum_{i=1}^N (w_{ij} \ln(w_{ij}) - w_{ij}) + \lambda_1 \sum_{i=1}^N \|w_{ij}\|_p^p \\ &\text{s.t.} \quad w_{ij} \in (0, 1]. \end{aligned} \tag{4.0.3}$$

In what follows, we extensively explain the main components of the algorithm and their characterestics. More precisely, we start by presenting appropriate initialization techniques. Subsequently, SAP K-subspaces ability of estimating the dimension, the variance and the number of subspaces is examined. We continue by considering the update of the main parameters and the selection of the regularization parameters. Note that the updating of w_{ij}'s is the same as in SPCM [24], while for the updating of $\boldsymbol{U}_j$'s and $\boldsymbol{Y}_j$'s a low-rank approach for dimensionality reduction is adopted, introduced in [10] (see section 2.2).

More specifically, the main processing steps of the proposed algorithm are summarized below.

- Initialization
- Main processing part (iterative)
  - Segmentation part (estimation of degrees of compatibility, w_{ij}'s)
  - Parameter estimation
    * $\boldsymbol{\mu}_j$'s (displacement vectors)
    * $\boldsymbol{U}_j$'s and $\boldsymbol{Y}_j$'s (bases and projections)
    * cluster elimination/ η_j's update
    * λ_1 selection
- model selection/ λ_2 selection

4.1 Initialization

As in K-means and K-subspaces, we optimize our cost function by following the Lloyd iterative alternating minimization algorithm [15]: the algorithm alternates between (a) the computation of the compatibility degrees of the data points with the clusters and (b) the computation of the parameters associated with each cluster (they define the corresponding subspace). The K-subspaces objective 3.2.1, which is also part of our objective 4.0.1, is non-convex and is known that suffers from many poor local minima. Hence, SAP K-subspaces is extremely sensitive to initialization.

The most dominant method for subspaces initialization is that of *Probabilistic Farthest Insertion* (PFI) [1]. According to PFI we select randomly a point to serve as cluster center. For this center we find a neighborhood and we estimate the subspace of the associated cluster. Usually, this is done by applying PCA in the neighborhood. Having define this subspace, the next point to be selected is the one that lies farthest from the subspaces that have already been generated and its associated subspace is estimated. This procedure continues until $\hat{K}$ initial subspaces are defined. Undoubtedly, the method is suitable only when there are no outliers. Therefore, we decided to take $\hat{K}$ randomly selected points from the data set to serve as cluster centers.

In the case of independent clusters, the SPCM algorithm is preferred over PFI. SPCM finds dense groups of data inside the ambient space. It exhibits immunity to noise/outliers and it is able to detect clusters of different densities. The representatives of the groups serve as neighborhood centers to estimate the corresponding subspaces. Nonetheless, clusters which are not dense enough for the algorithm to find them may exist.

The size of the neighbor sets of points plays an important role in the determination of the subspaces. Small neighborhoods may contain noisy data and large neighborhoods may include points from different subspaces. Both cases can result in bad initialization. To avoid this, we determine a neighborhood using the neighborhood size selection algorithm presented in [26].

[1] The method is called *probabilistic* because the probability of getting a point from the same subspace is small. Actually, the more distinct the clusters are, the smaller it the probability of selecting points from the same subspace becomes.

4.2 Main processing part

In this section the cost function 4.0.3 is studied with respect to the parameters it involves.

4.2.1 Segmentation (update of w_{ij}'s)

In the frame of SAP k-subspaces possibistic clustering algorithm, the w_{ij}'s are updating as in [24], provided that $\boldsymbol{U}_j$'s, $\boldsymbol{Y}_j$'s and $\boldsymbol{\mu}_j$'s are fixed.

From the compatibility degrees we get the segmentation of the data set. By taking the derivate of the cost function 4.0.3 w.r.t. w_{ij} we have that

$$F(w_{ij}) \equiv \frac{\partial \mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})}{\partial w_{ij}} = d_{ij} + \eta_j \ln(w_{ij}) + \lambda_1 p w_{ij}^{p-1}, \tag{4.2.1}$$

where $d_{ij} \equiv \text{dist}(\boldsymbol{x}_i, S_j)^2$, $p \in (0,1)$, $\lambda_1 \geq 0$ and $\eta_j > 0$ are all fixed. We search to find those w_{ij} that satisfy the equation $F(w_{ij}) = 0$. The last two terms of eq. 4.2.1 are continuous and differentiable functions of w_{ij}. Therefore, $F(w_{ij})$ is continuous and differentiable as well.

Proposition 4.2.1. $F(w_{ij}) \neq 0$ *for* $w_{ij} \notin (0,1)$.

Proof. For $w_{ij} \in (1, \infty)$ all the terms of $F(w_{ij})$ are positive. Note that if $w_{ij} \in (-\infty, 0)$, $F(w_{ij})$ is not defined. ∎

Proposition 4.2.2. $F(w_{ij})$ *has two stationary points,* $w_{ij}^* = \left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}}$ *and* $w_{ij}^{**} = +\infty$.

Proof.

$$\begin{aligned}\frac{\partial F(w_{ij})}{\partial w_{ij}} &= \frac{\partial(d_{ij} + \eta_j \ln(w_{ij}) + \lambda_1 p w_{ij}^{p-1})}{\partial w_{ij}} = \\ &= \eta_j w_{ij}^{-1} + \lambda_1 p(p-1) w_{ij}^{p-2} = \\ &= w_{ij}^{-1}[\eta_j - \lambda_1 p(1-p) w_{ij}^{p-1}], \quad w_{ij} > 0.\end{aligned}$$

If we set $\frac{\partial F(w_{ij})}{\partial w_{ij}} = 0$, then $\eta_j - \lambda_1 p(1-p) w_{ij}^{p-1} = 0$ or $w_{ij}^{-1} = 0$. By solving these equations respectively, we get $w_{ij}^* = \left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}} \geq 0$ and $w_{ij}^{**} = +\infty$. ∎

The question that arises is if $F(w_{ij})$ take its minimum at w_{ij}^*.

Proposition 4.2.3. $F(w_{ij})$ *has a unique minimum for* $w_{ij}^* = \left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}}$.

Proof. Assume that $w_{ij} \in (0, +\infty)$. It suffices to show that $\frac{\partial F(w_{ij})}{\partial w_{ij}} \leq 0$ for $w_{ij} \in (0, w_{ij}^*)$ and $\frac{\partial F(w_{ij})}{\partial w_{ij}} > 0$ for $w_{ij} \in (w_{ij}^*, +\infty)$.

The first derivative of $F(w_{ij})$ is

$$\frac{\partial F(w_{ij})}{\partial w_{ij}} = \eta_j w_{ij}^{-1} - \lambda_1 p(1-p) w_{ij}^{p-2}.$$

Clearly, for $w_{ij} \in (0, w_{ij}^*]$ it is true that

$$\begin{aligned} w_{ij} &\leq w_{ij}^* \Leftrightarrow \\ w_{ij} &\leq \left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}} \Leftrightarrow \\ w_{ij}^{1-p} &\leq \frac{\lambda_1 p(1-p)}{\eta_j} \Leftrightarrow \\ &\dots \\ 1 - \frac{\lambda_1 p(1-p)}{\eta_j} w_{ij}^{p-1} &\leq 0 \Leftrightarrow \\ \eta_j w_{ij}^{-1} - \lambda_1 p(1-p) w_{ij}^{p-2} &\leq 0. \end{aligned}$$

Thus, for $w_{ij} \in (0, w_{ij}^*]$, it holds that $\frac{\partial F(w_{ij})}{\partial w_{ij}} \leq 0$. Working in a similar manner, it turns out that, for $w_{ij} \in (w_{ij}^*, +\infty)$ it holds that, $\frac{\partial F(w_{ij})}{\partial w_{ij}} > 0$. Consequently, since $F(w_{ij}) \nearrow$ in $(w_{ij}^*, +\infty)$, w_{ij}^* is the unique minimum of $F(w_{ij})$.

A summary table is provided below.

w_{ij}	0		w_{ij}^*		$+\infty$
$F'(w_{ij})$		$-$		$+$	
$F(w_{ij})$	0	$\searrow$	w_{ij}^*	$\nearrow$	$+\infty$

■

Proposition 4.2.4. *If $F(w_{ij}^*) < 0$, equation $F(w_{ij}) = 0$ has exactly two solutions $w_{ij}^1, w_{ij}^2 \in (0,1)$ with $w_{ij}^1 < w_{ij}^2$.*

Proof. Because $F(w_{ij}^*) < 0$ (from assumption), $F(1) = d_{ij} + \eta_j \ln(1) + \lambda_1 p 1^{p-1} = d_{ij} + \lambda_1 p > 0$, $F(w_{ij}) \nearrow$ and continuous in $(w_{ij}^*, +\infty)$, from Bolzano's theorem there is exactly one $w_{ij}^2 \in (w_{ij}^*, 1)$ such that $F(w_{ij}^2) = 0$.

Moreover, since the logarithm is not defined for $w_{ij} = 0$, we have that

$$\begin{aligned} F(0) &= \lim_{w_{ij}\to 0^+} F(w_{ij}) = \\ &= \lim_{w_{ij}\to 0^+} (d_{ij} + \eta_j \ln(w_{ij}) + \lambda_1 p w_{ij}^{p-1}) = \\ &= d_{ij} + \lim_{w_{ij}\to 0^+} \left[\frac{1}{w_{ij}^{1-p}} (\eta_j w_{ij}^{1-p} \ln(w_{ij}) + \lambda_1 p) \right] \text{①} \end{aligned}$$

$$\lim_{w_{ij}\to 0^+} \frac{1}{w_{ij}^{1-p}} = +\infty \text{ ②}$$

$$\lambda_1 p + \lim_{w_{ij}\to 0^+} (\eta_j w_{ij}^{1-p} \ln(w_{ij})) = \lambda_1 p + \eta_j \lim_{w_{ij}\to 0^+} \frac{\ln(w_{ij})}{w_{ij}^{p-1}} \overset{\text{H}}{=} \lambda_1 p + \eta_j \lim_{w_{ij}\to 0^+} \frac{\frac{1}{x}}{(p-1)x^{p-2}} = \lambda_1 p \text{ ③}$$

Hence, combining ①, ② and ③, it is $F(0) = +\infty$.

Since, $F(0) > 0$, $F(w_{ij}^*) < 0$ (from assumption), $F(w_{ij}) \searrow$ and continuous in $(0, w_{ij}^*]$, from Bolzano's theorem there is exactly one $w_{ij}^1 \in (0, w_{ij}^*]$ such that $F(w_{ij}^1) = 0$. For the two solutions it clearly holds that $w_{ij}^1 < w_{ij}^2$. ■

Proposition 4.2.5. *If $F(w_{ij}) = 0$ has two solutions $w_{ij}^1, w_{ij}^2 \in (0, 1)$ with $w_{ij}^1 < w_{ij}^2$, $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ exhibits a local minimum (w.r.t. w_{ij}) at the largest of them, w_{ij}^2.*

Proof. From proposition 4.2.4, we know that if $F(w_{ij}^*) < 0$ holds, w_{ij}^1 and w_{ij}^2 are two solutions of $F(w_{ij}) = 0$ with $w_{ij}^1 < w_{ij}^* < w_{ij}^2$.

On the other hand, from proposition 4.2.3 it turns out that $F(w_{ij}) \nearrow$ in $(w_{ij}^*, +\infty)$ and $F(w_{ij}) \searrow$ in $(0, w_{ij}^*]$. By definition $F(w_{ij})$ is continuous to those intervals.

Additionaly to $F(0) > 0$,

$$\begin{aligned} F(+\infty) &= \lim_{w_{ij}\to +\infty} F(w_{ij}) = \\ &= \lim_{w_{ij}\to +\infty} (d_{ij} + \eta_j \ln(w_{ij}) + \lambda_1 p w_{ij}^{p-1}) = \\ &= d_{ij} + \eta_j(+\infty) + \lambda_1 p \lim_{w_{ij}\to +\infty} \frac{1}{w_{ij}^{1-p}} = +\infty > 0 \text{ as well.} \end{aligned}$$

Furthermore,

$$\begin{aligned}\mathcal{C}(\mu_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, 0) &= \lim_{w_{ij}\to 0^+} (d_{ij}w_{ij} + \eta_j(w_{ij}\ln(w_{ij}) - w_{ij}) + \lambda_1 w_{ij}^p) = \\ &= \lim_{w_{ij}\to 0^+} \eta_j(w_{ij}\ln(w_{ij}) - w_{ij}) = \\ &= \eta_j \left[\lim_{w_{ij}\to 0^+} \frac{\ln(w_{ij})}{\frac{1}{w_{ij}}} - \lim_{w_{ij}\to 0^+} w_{ij}\right] \overset{\text{H}}{=} \\ &\overset{\text{H}}{=} \eta_j \lim_{w_{ij}\to 0^+} \frac{\frac{1}{w_{ij}}}{-\frac{1}{w_{ij}^2}} = \lim_{w_{ij}\to 0^+} -w_{ij} = 0.\end{aligned}$$

Summing up, we have the following table.

w_{ij}	0		w_{ij}^1		w_{ij}^*		w_{ij}^2		$+\infty$
$F(w_{ij})$		$+$	0		$-$		0	$+$	
$\mathcal{C}$	0	↗	w_{ij}^1	↘		↘	w_{ij}^2	↗	$+\infty$

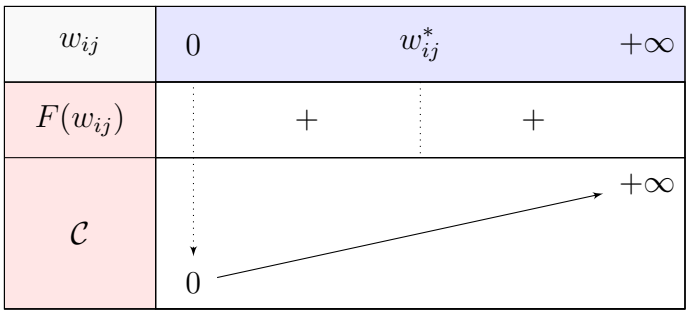
Consequently, $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ exhibits a local minimum for w_{ij}^2, the largest solution among the two that $F(w_{ij}) = 0$. ■

Proposition 4.2.6. *$\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ exhibits its global minimum (w.r.t w_{ij}) at $\widehat{w_{ij}}$, where*

$$\widehat{w_{ij}} = \begin{cases} w_{ij}^2, & \textit{if } F(w_{ij}^*) < 0 \textit{ and } w_{ij}^2 > (\frac{\lambda_1(1-p)}{\eta_j})^{\frac{1}{1-p}} \\ 0, & \textit{otherwise}. \end{cases} \tag{4.2.2}$$

Proof. From the table of proposition 4.2.5, it follows that if $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij}^2) < \mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, 0)$ then $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ has a global minimum at w_{ij}^2. This means that, $w_{ij}^2[d_{ij} + \eta_j \ln(w_{ij}^2) - \eta_j + \lambda_1(w_{ij}^2)^{p-1}] < 0$. However, having in mind that $F(w_{ij}^2) = 0$, the latter becomes $w_{ij}^2[-\lambda_1 p w_{ij}^{p-1} - \eta_j + \lambda_1(w_{ij}^2)^{p-1}] < 0$. Since $w_{ij}^2 > 0$, it is $-\lambda_1 p w_{ij}^{p-1} - \eta_j + \lambda_1(w_{ij}^2)^{p-1} < 0$. Solving w.r.t. w_{ij}^2, we get $w_{ij}^2 > (\frac{\lambda_1(1-p)}{\eta_j})^{\frac{1}{1-p}}$.

In the case where $F(w_{ij}^*) > 0$, observe that $\mathcal{C}(\mu_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij}) \nearrow$ in $(0, +\infty)$. Therefore, the minimum is achieved for $w_{ij} = 0$. The table below illustrates that particular case.



■

All of that being said, for the computation of w_{ij} we proceed as follows. Firstly, we determine the w_{ij}^* and the value of $F(w_{ij}^*)$. Bear in mind that by definition $w_{ij} \in (0, 1]$ (see 3.1.4). We continue by comparing $F(w_{ij}^*)$ with zero. If $F(w_{ij}^*) > 0$, then from propositions 4.2.5 and 4.2.6 we know that $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, 0) = 0$ and $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij}) \nearrow$ in $(0, 1]$. Thus, we impose sparsity by setting manually $w_{ij} = 0$. In the rare case of proposition 4.2.3, where $F(w_{ij}^*) = 0$, we set again $w_{ij} = 0$ since $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, 0) = 0$ and $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij}) \nearrow$ in $(0, w_{ij}^*) \cup (w_{ij}^*, 1]$. If $F(w_{ij}^*) < 0$, we know from proposition 4.2.5 that $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ exhibits a local minimum at the largest solution of $F(w_{ij}) = 0$, denoted by w_{ij}^2. For the estimation of w_{ij}^2, we use the bisection method as in [24]. Lastly, we use the eq. 4.2.2 to get the global minimum of $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, w_{ij})$ w.r.t. w_{ij}.

4.2.2 Update of $\boldsymbol{\mu}_j$'s

To update the displacement vector $\boldsymbol{\mu}_j$ of a subspace S_j, we use the formula deduced from the derivation of the cost function 4.0.3 w.r.t. $\boldsymbol{\mu}_j$.

Eq. 4.0.2 can be written as,

$$\begin{aligned}\text{dist}(\boldsymbol{x}_i, S_j)^2 &= \|\boldsymbol{x}_i - (\boldsymbol{\mu}_j + \boldsymbol{U}_j\boldsymbol{y}_{ij})\|_2^2 = \\ &= \boldsymbol{x}_i^\top \boldsymbol{x}_i - 2\boldsymbol{x}_i^\top(\boldsymbol{\mu}_j + \boldsymbol{U}_j\boldsymbol{y}_{ij}) + (\boldsymbol{\mu}_j + \boldsymbol{U}_j\boldsymbol{y}_{ij})^\top(\boldsymbol{\mu}_j + \boldsymbol{U}_j\boldsymbol{y}_{ij}) = \\ &= \boldsymbol{x}_i^\top \boldsymbol{x}_i - 2\boldsymbol{x}_i^\top\boldsymbol{\mu}_j - 2\boldsymbol{x}_i^\top\boldsymbol{U}_j\boldsymbol{y}_{ij} + \boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j + 2\boldsymbol{\mu}_j^\top\boldsymbol{U}\boldsymbol{y}_{ij} + (\boldsymbol{U}_j\boldsymbol{y}_{ij})^\top(\boldsymbol{U}_j\boldsymbol{y}_{ij}).\end{aligned} \tag{4.2.3}$$

From 4.0.3 and 4.2.3, we have that

$$\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, \{w_{ij}\}_{i=1}^N) = \sum_{i=1}^N w_{ij}[-2\boldsymbol{\mu}_j^\top\boldsymbol{x}_i + \boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j + 2\boldsymbol{\mu}_j^\top\boldsymbol{U}_j\boldsymbol{y}_{ij}] + \text{ terms independent of } \boldsymbol{\mu}_j. \tag{4.2.4}$$

Thus,

$$\frac{\partial\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, \{w_{ij}\}_{i=1}^N)}{\partial\boldsymbol{\mu}_j} = -2\sum_{i=1}^N w_{ij}\boldsymbol{x}_i + 2\boldsymbol{\mu}_j\sum_{i=1}^N w_{ij} + 2\sum_{i=1}^N w_{ij}\boldsymbol{U}_j\boldsymbol{y}_{ij}. \tag{4.2.5}$$

By setting the derivative 4.2.5 equal to zero, we can prove that

$$\mu_j = \frac{\sum_{i=1}^N w_{ij}\boldsymbol{x}_i - \sum_{i=1}^N w_{ij}\boldsymbol{U}_j\boldsymbol{y}_{ij}}{\sum_{i=1}^N w_{ij}}. \tag{4.2.6}$$

Assuming that all the parameters in $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, \{w_{ij}\}_{i=1}^N)$ are kept fixed except $\boldsymbol{\mu}_j$, squared euclidean norm is convex. Therefore, 4.2.6 is the global minimum of 4.2.4.

4.2.3 Update of U_j's and Y_j's

In the sequel, we show how the method expressed in [10] (see section 2.2) can be adapted in the framework of the present problem in order to tackle the problem of the unknown subspace dimension. Note that in this section, we consider the w_{ij}'s and $\boldsymbol{\mu}_j$'s fixed.

Let $m_j = |\boldsymbol{x_i} \in X : w_{ij} > \alpha|$ the estimated number of points that lie on the subspace S_j, for $\alpha \in [0, 1)$. High values of α let us consider only the most compatible points to the cluster C_j in order to estimate their corresponding subspace S_j.

Therefore, minimizing $\mathcal{C}(\boldsymbol{\mu}_j, \boldsymbol{U}_j, \boldsymbol{Y}_j, \{w_{ij}\}_{i=1}^N)$ w.r.t. $\boldsymbol{U}_j$ and $\boldsymbol{Y}_j$ is equivalent to

$$\min_{\boldsymbol{U}_j, \dot{\boldsymbol{Y}}_j} \sum_{i=1}^{m_j} \|\sqrt{w_{ij}}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) - \sqrt{w_{ij}}\boldsymbol{U}_j \boldsymbol{y}_{ij}\|_2^2. \tag{4.2.7}$$

In matrix form the above problem can be rewritten as,

$$\min_{\boldsymbol{U}_j, \dot{\boldsymbol{Y}}_j} \|\dot{\boldsymbol{X}}_{\boldsymbol{j}} - \boldsymbol{U}_j \dot{\boldsymbol{Y}}_j\|_F^2, \tag{4.2.8}$$

where $\dot{\boldsymbol{X}}_{\boldsymbol{j}} \triangleq [\sqrt{w_{1j}}(\boldsymbol{x}_1 - \boldsymbol{\mu}_j), \sqrt{w_{2j}}(\boldsymbol{x}_2 - \boldsymbol{\mu}_j), ..., \sqrt{w_{m_j j}}(\boldsymbol{x}_{m_j} - \boldsymbol{\mu}_j)] \in \mathbb{R}^{L \times m_j}$ and $\dot{\boldsymbol{Y}}_j \triangleq [\sqrt{w_{1j}}\boldsymbol{y}_1, \sqrt{w_{2j}}\boldsymbol{y}_2, ..., \sqrt{w_{m_j j}}\boldsymbol{y}_m] \in \mathbb{R}^{\hat{d} \times m_j}$. With $\boldsymbol{X}_{\boldsymbol{j}}$, we denote the matrix with columns the elements of $\{\boldsymbol{x}_i \in X : w_{ij} > \alpha, i = 1, ..., N\}$.

For the adaptation of d_j a low-rank approach is adopted. The authors of [10] propose a regularizer that imposes column-sparsity jointly on two matrix factors (see section 2.2). In our problem those matrices are $\boldsymbol{U}_j$ and $\dot{\boldsymbol{Y}}_j$. More specifically, the regularizer imposes sparsity on the columns of the concatenated matrix $\begin{bmatrix} \boldsymbol{U}_j \\ \dot{\boldsymbol{Y}}_j^\top \end{bmatrix}$ using the $\ell_{r,2}^r$ norm, where $0 < r \leq 1$. By adopting this technique, the updating problem we have to solve becomes

$$\{\hat{\boldsymbol{U}}, \hat{\boldsymbol{Y}}\} = \underset{\boldsymbol{U}_j, \dot{\boldsymbol{Y}}_j}{\operatorname{argmin}} \frac{1}{2}\|\dot{\boldsymbol{X}}_{\boldsymbol{j}} - \boldsymbol{U}_j \dot{\boldsymbol{Y}}_j\|_F^2 + \lambda_2 \sum_{c=1}^{\hat{d}} (\|\boldsymbol{u}_c\|_2^2 + \|\dot{\boldsymbol{y}}_c\|_2^2 + z^2)^{\frac{r}{2}}, \tag{4.2.9}$$

where $\boldsymbol{u}_c \in \mathbb{R}^{L \times 1}$ and $\dot{\boldsymbol{y}}_c \in \mathbb{R}^{m \times 1}$ are the column parts of the concatenated matrix and z^2 a small positive constant that alleviates singular points i.e., points where the gradient is not continuous.

To tackle the above non-convex problem, we use the AIRLS - Denoising algorithm (algorithm 1) that follows the block successive upper-bound minimization (BSUM) framework in order to alternatingly minimize eq. 4.2.9.

After convergence, $\hat{\boldsymbol{Y}}$ has to be unweighted. This is done by dividing each i-column of $\hat{\boldsymbol{Y}}$ by $\sqrt{w_{ij}}$.

The produced matrix does not hold the projections of the entire data set onto S_j. Therefore, the other $(N - m_j)$-projections are estimated by the formula produced by minimizing eq. 4.0.2.

More specifically, eq. 4.0.2 can be equivalently written as,

$$\begin{aligned} \text{dist}(\boldsymbol{x}_i, S_j)^2 &= (\boldsymbol{x}_i - \boldsymbol{\mu}_j - \boldsymbol{U}_j\boldsymbol{y}_{ij})^\top(\boldsymbol{x}_i - \boldsymbol{\mu}_j - \boldsymbol{U}_j\boldsymbol{y}_{ij}) = \\ &= (\boldsymbol{x}_i^* - \boldsymbol{U}_j\boldsymbol{y}_{ij})^\top(\boldsymbol{x}_i^* - \boldsymbol{U}_j\boldsymbol{y}_{ij}) = \\ &= \boldsymbol{x}_i^{*\top}\boldsymbol{x}_i^* - \boldsymbol{x}_i^{*\top}\boldsymbol{U}_j\boldsymbol{y}_{ij} - \boldsymbol{y}_{ij}^\top\boldsymbol{U}_j^\top\boldsymbol{x}_i^* + \boldsymbol{y}_{ij}^\top\boldsymbol{U}_j^\top\boldsymbol{U}_j\boldsymbol{y}_{ij}, \end{aligned} \tag{4.2.10}$$

where $\boldsymbol{x}_i^* = \boldsymbol{x}_i - \boldsymbol{\mu}_j$.

Since the projection of a point onto a subspace is the point for which $\text{dist}(\boldsymbol{x}_i, S_j)^2$ is minimized, we take the derivative of the above w.r.t. $\boldsymbol{y}_i$,

$$\frac{\partial \text{dist}(\boldsymbol{x}_i, S_j)^2}{\partial \boldsymbol{y}_i} = -\boldsymbol{U}_j^\top\boldsymbol{x}_i^* - \boldsymbol{U}_j^\top\boldsymbol{x}_i^* + 2\boldsymbol{U}_j^\top\boldsymbol{U}_j\boldsymbol{y}_i \tag{4.2.11}$$

and we set it equal to zero.

This yields to the least squares formula

$$\boldsymbol{y}_i = (\boldsymbol{U}_j^\top\boldsymbol{U}_j)^{-1}\boldsymbol{U}_j^\top\boldsymbol{x}_i^*, \tag{4.2.12}$$

which is used for the computation of the other $(N - m_j)$-projections.

4.2.4 Cluster elimination and update of η_j's

An intrinsic feature of the PCM algorithms is the so called mode-seeking property. That is, its cluster representative converges to its closest physical cluster. This means that if two representatives initialize close to the same cluster, they converge independently towards it, giving thus rise to the ”coincident clusters” phenomenon. This is the principle upon which the elimination clusters mechanism is based.

Possibilistic by nature, SAP K-subspaces inherits the mode-seeking property. More precisely, it turned out that given an overestimation of the actual number of clusters, $\hat{K}$, and by selecting properly the value of the parameter Ξ, the algorithm is able to uncover the true structure of the data set by eliminating clusters.

Although we follow a possibilistic approach of subspaces estimation, currently all the applications of subspace clustering are of a hard logic (a point strictly belongs only to one cluster). Therefore, in order to find the membership of the clusters, we assign each point with its closest cluster. More precisely, each of the points is assigned to the cluster for which it exhibits the maximum compatibility degree with. In case of outliers, those x_i's with $w_{ij} = 0, \forall j = 1, 2, ..., K$ are left unassigned. By this way, only a portion of the initial subspaces will be moved closer to the clusters (those that fit best), the rest are going to fit small groups of points that either approximately lie on the same subspace (coincide) or form noise clouds[2].

After assigning points to clusters, the cluster elimination phase takes place. Specifically, a cluster is eliminated if either its cardinality is less than the dimension of the associated subspace d_j or if the dimension of subspace d_j is zero. The latter occurs when λ_2 shrinks toward zero all the columns of the concateneted matrix by showing that this particular group of points does not spread along any subspace.

Next, the variances η_j's of the remaining clusters are updated. More precisely, we estimate the η_j of a cluster C_j, as the variance around its associated subspace S_j, i.e.,

$$\eta_j = \frac{\sum_{\boldsymbol{x}_i \in C_j} \mathsf{dist}(\boldsymbol{x}_i, S_j)^2}{|C_j|}, \tag{4.2.13}$$

where $C_j = \{\boldsymbol{x_i} \in X : w_{ij} = \mathsf{max}_{r=1,...,K}\, w_{ir}\}$.

[2]Points that lie (mainly) on different subspaces, also conduct noise clouds.

4.2.5 λ_1 selection

From propositions 4.2.1 to 4.2.6, it is obvious that for $F(w_{ij}) = 0$ to hold, it should be $F(w_{ij}^*) < 0$, which means that

$$\begin{aligned}
F(w_{ij}^*) &< 0 \\
d_{ij} + \eta_j \ln\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}} + \lambda_1 p\left[\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{\frac{1}{1-p}}\right]^{p-1} &< 0 \\
d_{ij} + \frac{\eta_j}{1-p}\ln\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right) + \lambda_1 p\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right)^{-1} &< 0 \\
d_{ij} + \frac{\eta_j}{1-p}\ln\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right) + \frac{\eta_j}{1-p} &< 0 \\
\frac{(1-p)d_{ij}}{\eta_j} + \ln\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right) + 1 &< 0 \\
\frac{(1-p)d_{ij} + \eta_j}{\eta_j} + \ln\left(\frac{\lambda_1 p(1-p)}{\eta_j}\right) &< 0 \\
\exp\left(-\frac{(1-p)d_{ij} + \eta_j}{\eta_j}\right) &> \frac{\lambda_1 p(1-p)}{\eta_j} \\
\frac{\eta_j}{p(1-p)}\exp\left(-1 - \frac{(1-p)d_{ij}}{\eta_j}\right) &> \lambda_1.
\end{aligned} \tag{4.2.14}$$

Therefore, choices of $\lambda_1 \leq (\geq) \frac{\eta_j}{p(1-p)} \exp\left(-1 - \frac{(1-p)d_{ij}}{\eta_j}\right)$ result to $w_{ij} > 0\ (= 0)$, imposing (not imposing) sparsity.

By setting $d_{ij} = \eta_j$, it is easy to show that eq. 4.2.14 becomes

$$\lambda_1 \leq \frac{\eta_j}{p(1-p)\exp(2-p)}. \tag{4.2.15}$$

Inequality 4.2.14, is equivalent to

$$\lambda_1 = \Xi \frac{\eta_j}{p(1-p)\exp(2-p)}, \quad \text{for } \Xi \in (0,1). \tag{4.2.16}$$

For each cluster C_j we have a different λ_1 parameter.

The algorithm's performance is strictly linked to the choice of Ξ. Choices of Ξ around 1 force w_{ij}'s of points with $d_{ij} \gtrsim \eta_j$ to become zero. Generally, it is better to avoid using a high value of Ξ since it prohibits badly initialized subspaces from moving around the euclidean space and eventually minimize the cost function 4.0.3.

Note that the variances η_j's together with the parameter Ξ implicitly determine the cardinality of the clusters through eq. 4.2.16.

4.3 Model selection (λ_2 selection)

The low-rank promoting parameter λ_2 of AIRLS is selected from a set of values based on the model which achieves the lowest penalized *Root Mean Squared Error* (RMSE). As model we imply the union of subspaces $S_1 \cup S_2 \cup ... \cup S_K$ which fits a particular data set. Since the optimization problem we solve in 4.2.9 is weighted, the values in this set do not have to vary a lot.

The penalized RMSE is combuted using the formula

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^2} + \frac{K}{\hat{K}} + \frac{\delta}{\hat{d}}, \tag{4.3.1}$$

where $\hat{K}$ and $\hat{d}$ the initial number of subspaces and their initial dimension respectively, K the estimated number of clusters and δ the median of a set which includes all the estimated dimensions $\{d_j\}_{j=1}^{K}$. For the computation of $\hat{\boldsymbol{x}}_i$'s it holds $\hat{\boldsymbol{x}}_i = \boldsymbol{H}_j \boldsymbol{x}_i$, where $\boldsymbol{x}_i \in C_j$ and $\boldsymbol{H}_j = \boldsymbol{U}_j(\boldsymbol{U}_j^\top \boldsymbol{U}_j)^{-1}\boldsymbol{U}_j^\top$ the hat matrix.

The second term helps to avoid selecting overfitted models. Typically, this is the case when the data set is corrupted by noise. The third one encourages the selection of models which correspond to subspaces of reduced dimension. Without the last term, the model which achieves the lowest RMSE has (usually) subspaces of dimension $\hat{d}$ (no dimensionality reduction applied). The reason is that a subspace of dimension $\hat{d}$ is closer to its corresponding group of points than the associated subspace with reduced dimension.

Assuming one subspace, the value of λ_2 depends on L, $|C_j|$ and η_j. Therefore, for λ_2 to be the same to all clusters, we assume that $|C_j|$ and η_j are the same $\forall j = 1, 2, ..., K$.

4.4 SAP K-subspaces algorithm

In the following, combining all the individual stages described below, we give the SAP k-subspaces in its algorithmic form.

Algorithm 9 Sparse Adaptive Possibilistic K-Subspaces (SAP K-Subspaces)

Input: Data matrix $\boldsymbol{X} \in \mathbb{R}^{L\times N}$, number of initial subspaces $\hat{K}$, norm's parameter $p \in (0,1)$ to handle outliers, parameter $\Xi \in (0,1)$, the compatibilities cut-off α, the parameters used in algorithm 1.

Initialization: Initialize the bases $\{\boldsymbol{U}_j\}_{j=1}^{\hat{K}}(\boldsymbol{U}_j \in \mathbb{R}^{L\times\hat{d}})$, the projections $\{\boldsymbol{Y}_j\}_{j=1}^{\hat{K}}(\boldsymbol{Y}_j \in \mathbb{R}^{\hat{d}\times N})$, the displacement vectors $\{\boldsymbol{\mu}_j\}_{j=1}^{\hat{K}}(\boldsymbol{\mu}_j \in \mathbb{R}^{L})$ and the variances $\{\eta_j\}_{j=1}^{\hat{K}}$.
repeat
  $K = \hat{K}$
  for $j = 1 : K$ **do**
    $\lambda_1 = \Xi\frac{\eta_j}{p(1-p)\,\exp(2-p)}$
    for $i = 1 : N$ **do** ▷ Segmentation part
      $\text{dist}(\boldsymbol{x}_i, S_j)^2 = \|\boldsymbol{x}_i - \boldsymbol{\mu}_j - \boldsymbol{U}_j\boldsymbol{y}_{ij}\|_2^2$
      Use the distance to update w_{ij}'s, j=1,...,K as described in 4.2.1
    end for
    $\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} w_{ij}\boldsymbol{x}_i - \sum_{i=1}^{N} w_{ij}\boldsymbol{U}_j\boldsymbol{y}_{ij}}{\sum_{i=1}^{N} w_{ij}}$ ▷ Estimation part
    Update $\boldsymbol{U}_j$ and $\boldsymbol{Y}_j$ as described in 4.2.3
  end for
  Perform cluster elimination and re-estimate K as described in 4.2.4
  Update η_j's using the eq. 4.2.13
until convergence

Output: Compatibility degree matrix $\boldsymbol{W}$, $\{\boldsymbol{U}_j,\boldsymbol{Y}_j,\boldsymbol{\mu}_j, d_j\}_{j=1}^{K}$.

Example 1: In the example illustrated in figure 4, we have a 3-dimensional data set that comes from three linear subspaces: two 1-dimensional and one 2-dimensional. Each group consists of 200 points generated as follows: We generate first 200 points uniformly distributed in a ball of radius one in subspace associated with this group. Then, we add zero mean Gaussian noise with covariance matrix $0.02^2\boldsymbol{I}_L$.

Subspaces are initialized based on the neighborhoods of the most distant points of the data set (those with the largest euclidean norm).

In AIRLS and SAP K-subspaces, we set $r = 1$, $z = 0.1$, $p = 0.3$ and $\alpha = 0$. Parameter λ_2 is fine-tuned by searching which value of the set $\{0.01, 0.1, 0.3, 0.5, 0.7, 0.9\}$ achieves the lowest RMSE score of the model.

The initial subspaces of (a)-(b) lie close to the actual ones. Thus, with $\Xi = 0.7$ we successfully impose sparsity to the compatibility degrees which correspond to data that come from different clusters. For comparison reasons, we keep the value of λ_2 obtained from the experiment depicted in (a)-(b). Looking at (c)-(d), it is apparent that even for a small overestimation, $\hat{K} = K + 1$, the algorithm fails to detect the underlying structure of the data set as it estimates four clusters. Specifically, it finds three 1-dimensional subspaces instead of two. This is due to the high value of Ξ that prevents the badly initialized subspace (marked with cyan) from moving. On the other hand, in (e)-(f) the small value of Ξ allows multiple subspaces to (approximately) fall into one single subspace. The subspace that best fits the data is kept and the clusters with cardinality less than the dimension of the corresponding subspace are eliminated. As a result, we have three subspaces.

With all that being said, parameter Ξ should be chosen considering how confident we are that the initialization worked in terms of S_j. Assuming good (unknown) initialization and no overestimated (overestimated) K, high (small) values of Ξ are preferred.

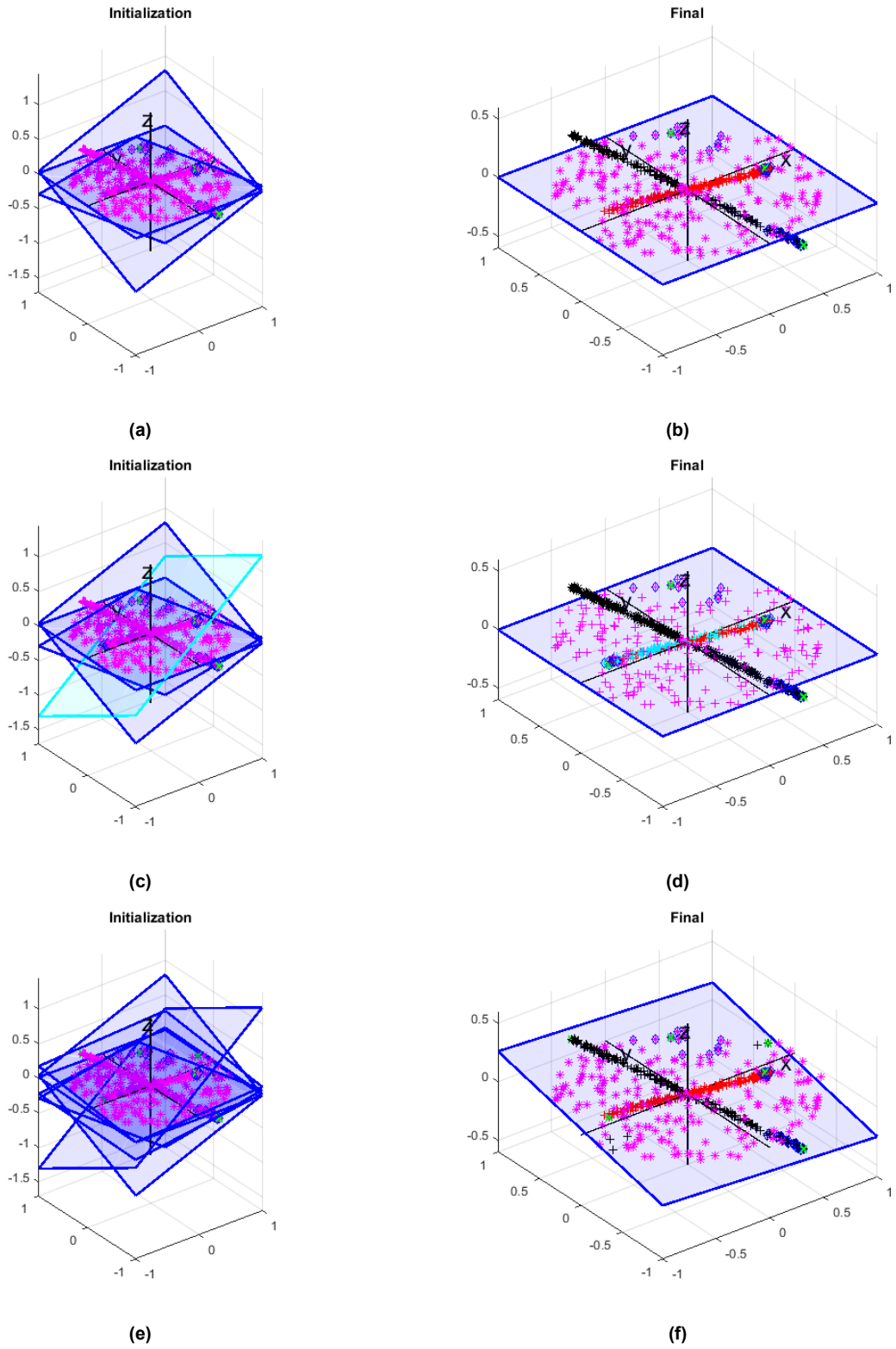


Figure 4: The mode-seeking property of SAP K-subspaces in linear subspaces

Green points are the initialization centers and the blue ones are their neighbors. (a) Three 2-dimensional initial subspaces. (b) The obtained clustering result for $\Xi = 0.7$ and $\lambda_2 = 0.9$. (c) Four 2-dimensional initial subspaces. (d) The obtained clustering result for $\Xi = 0.7$ and $\lambda_2 = 0.9$. (e) Six 2-dimensional initial subspaces. (f) The obtained clustering result for $\Xi = 0.01$ and $\lambda_2 = 0.9$.

Example 2: In this example, we generate subspaces of the same dimension and group size as before. However, this time the data lie on affine subspaces which correspond to three independent clusters (there are no points shared between them). The results are depicted in figure 5.

Starting from (a)-(b) clustering, we see that the initial subspaces lie close to the real ones. Thus, we take $\Xi = 0.7$. By setting in addition $\alpha = 0.6$, we do not only impose high sparsity, but also we prohibit points with $w_{ij} <= 0.6$ from contributing to the estimation of S_j. The algorithm successfully detects the clusters. In (c)-(d) and (e)-(f), the number of initial subspaces is increased by one. As we see, this new subspace (marked with cyan) passes through all the clusters. For small values of Ξ and α the algorithm fails to uncover the true data structure. Specifically, the tiny value of Ξ allows the subspaces to move far from the initialized ones, while the zero value of α does not limit the number of points used during subspace estimation. Therefore, the subspace which lies more close to the entire data set gets the majority of the points assigned to it. On the other hand, when $\Xi = 0.7$ and $\alpha = 0.6$, the cluster associated with this new subspace is eliminated as it runs out of points.

Apparently, in the case where the clusters are independent, values of α around 0.5 seem to work properly. In this way, we force the algorithm to find a ”less possibilistic” solution to the problem, in the sense that points with high enough compatibility with a cluster (but less that 0.5) are excluded from the process of estimating the associated subspace.

From both examples, we conclude that the right choice of Ξ is of high importance for SAP K-subspaces, otherwise the mode-seeking property of the algorithm is undermined. In the case of linear subspaces the values of Ξ which achieve the highest accuracy are usually smaller than those used in affine subspaces.

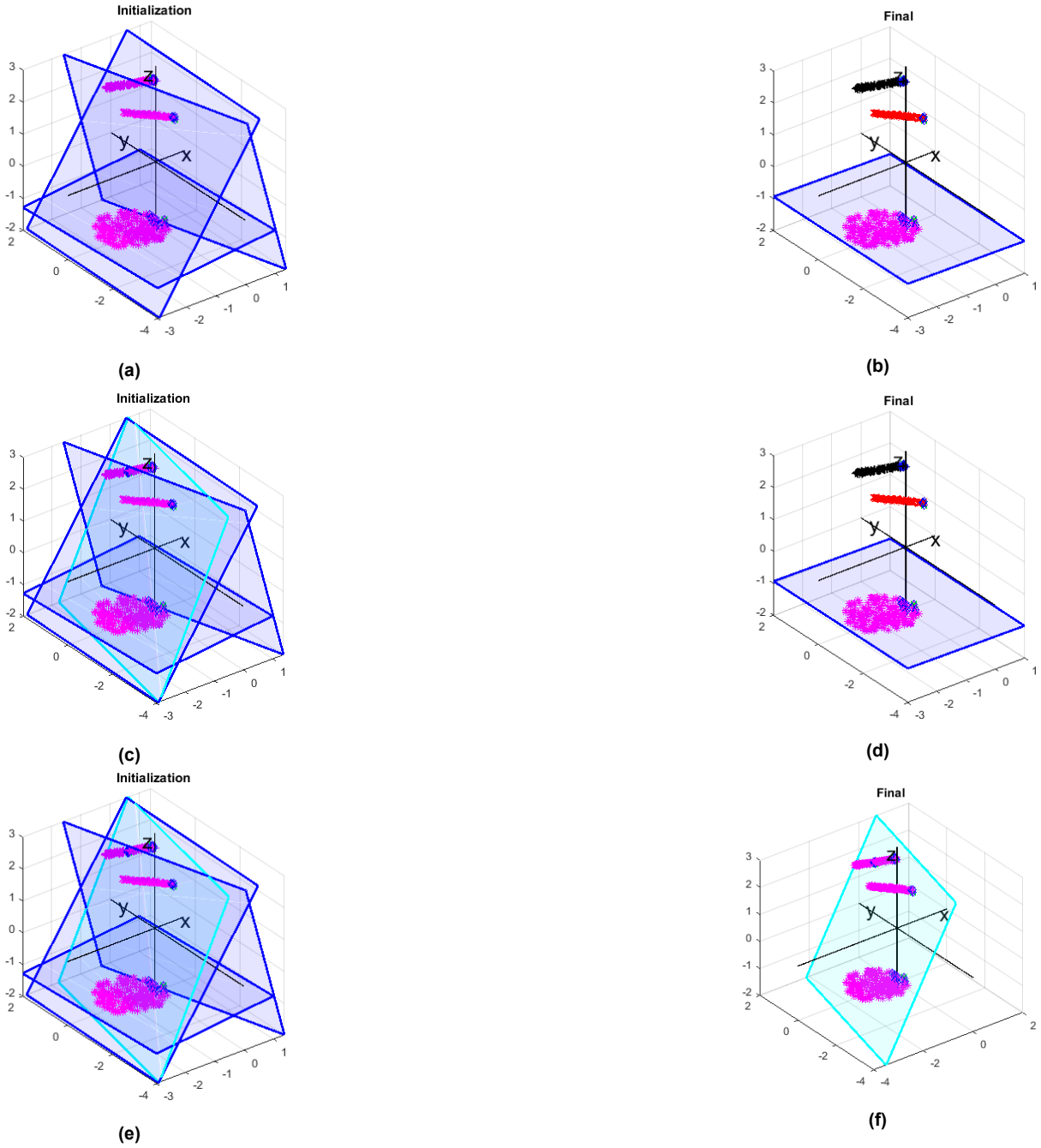


Figure 5: The mode-seeking property of SAP K-subspaces in affine subspaces

Green points are the initialization centers and the blue ones are their neighbors. (a) Three 2**-dimensional initial subspaces. (b) The obtained clustering result for** $\Xi = 0.7$**,** $\alpha = 0.6$ **and** $\lambda_2 = 0.9$**. (c) Four** 2**-dimensional initial subspaces. (d) The obtained clustering result for** $\Xi = 0.7$**,** $\alpha = 0.6$ **and** $\lambda_2 = 0.9$**. (e) Four** 2**-dimensional initial subspaces. (f) The obtained clustering result for** $\Xi = 0.01$**,** $\alpha = 0$ **and** $\lambda_2 = 0.9$**.**

5. EXPERIMENTAL RESULTS

In this Chapter, we conduct experiments on both synthetic and real data sets to illustrate the effectiveness of the proposed algorithm. The adopted performance metric is the rate of misclassified data points:

$$\text{misclassification error\%} = \frac{|\text{misclassified data}|}{|\text{data}|} \times 100\%. \tag{5.0.1}$$

A point is assigned to the cluster for which it exhibits the highest compatibility degree. If that degree is the same for more than one subspace, the choice is made at random.

To match the true labels with the cluster labels, we use a modification of the classical Hungarian algorithm that is suitable for both square and rectangular cost matrices [7]. In order to form pairs, the algorithm creates a $m \times \hat{m}$ cost matrix $\boldsymbol{C}$, where m the number of true clusters and $\hat{m}$ the number of estimated clusters. In the context of clustering, c_{ij} denotes the number of points which simultaneously belong to the i-real and j-estimated cluster respectively. When the rectangular case occurs ($m \neq \hat{m}$), the algorithm matches the min$\{m, \hat{m}\}$ cluster labels with the true ones and all the other data points are marked as misclassified.

In the following experiments, d^K in $\mathbb{R}^L$ means that in this specific experiment we have K subspaces of dimension d inside the L-dimensional Euclidean space. Also, $(d_1, ..., d_K)$ in $\mathbb{R}^L$ means that we have K subspaces of dimension $d_1, ..., d_K$ respectively.

SAP K-subspaces is compared with the algorithms introduced in Chapter 3: K-subspaces, RANSAC, SLBF and SSC.

We use the code of K-subspaces from *http://www.math.sjsu.edu/ gchen/scc.html*, the code of SLBF from *https://sciences.ucf.edu/math/tengz/lbf/* and the codes of RANSAC and SSC from *http://www.vision.jhu.edu/code.htm*. For the SSC algorithm we use the implementation which adopts the alternating direction method of multipliers (ADMM) and we perform clustering using the projections of the data set onto a subspace of dimension $(\max_j d_j + 1)$. Following [8], in SSC we set $\alpha = 800$. RANSAC and SLBF do not support the case where the dimensions of the subspaces differ. Thus, for those algorithms we assume that the dimension of each subspace equals to the maximum among all subspace dimensions ($\max_j d_j + 1$ for affine subspaces). Especially, for the K-affine case in RANSAC we assume that each subspace has the maximum dimension plus 1. The inliers threshold for RANSAC is $\eta = \text{Mean}(\{\eta_j\}_{j=1}^K) \times 0.8$. In SLBF algorithm, we take the clustering that

achieves the lowest average RMSE score. In AIRLS, it is $z = 0.1$, $\hat{d} = \max_j d_j$ and $r = 1$. Parameter λ_2 is fine-tuned as described in section 4.3. The AIRLS algorithm converges when either the relative decrease of the reconstructed data between two successive iterations i.e., $\frac{\|\boldsymbol{U}_k \boldsymbol{Y}_k - \boldsymbol{U}_{k+1}\boldsymbol{Y}_{k+1}\|_F}{\|\boldsymbol{U}_k \boldsymbol{Y}_k\|_F}$, becomes less than 10^{-5} or 500 iterations have been executed.

For the comparison to be valid, we initialize K-subspaces and SAP K-subspaces by the same methods, that is: PFI for the K-linear subspaces problem and SPCM for the K-affine subspaces problem. However, in SAP K-subspaces we overestimate the number of the true clusters. Generally, it is $\hat{K} = 3 * K$, otherwise $\hat{K}$ is given as comment above the tables with the results. In both K-subspaces and SAP K-subspaces, as well in SLBF, the determination of the neighborhoods' size is carried out via algorithm 4. More precisely, we set $S = 2 \times \hat{d}$ and $T = 2$. In SPCM initialization, the estimated cluster centers serve as neighborhood centers. For SPCM, it is $p = 0.5$ and $\Xi = 0.9$. After estimating the neighborhoods, we perform PCA and initialize the parameters: $\{\boldsymbol{\mu}_j\}_{j=1}^{\hat{K}}$, $\{\boldsymbol{U}_j\}_{j=1}^{\hat{K}}$, $\{\boldsymbol{y}_i\}_{i=1}^{N}$ and $\{\eta_j\}_{j=1}^{\hat{K}}$.

For the update of the w_{ij}'s in SAP K-subspaces we set $p = 0.3$. Parameter Ξ varies per experiment. The algorithm stops when either the average penalized RMSE score of the model between two successive iterations (concluded to the same number of clusters) becomes less than 10^{-6} or 250 iterations are reached. At the end of the procedure, all the unassigned points are assigned to their closest cluster.

The experiments were taken in an Intel Core i5, 5200 CPU at 2.20GHz and 6 GB memory, using MATLAB version 2016a.

5.1 Clustering results on simulated data

Synthetic experiments are divided into two parts. In the first part we study the case of the K-linear subspaces and in the second part the case of the K-affine subspaces. For each subspace S_j of dimension d_j in $\mathbb{R}^L$, 200 samples are generated according to the model described below. The data are further corrupted with 5% or 30% uniformly distributed outliers in a cube with edge length the euclidean norm of the most distant point in the data set. The experiments in both cases will be repeated for 30 random trials and the mean, as well as the median misclassification error will be reported.

The data set is formulated by points that follow the noisy Gaussian model:

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{\mu}_j + \boldsymbol{U}_j \boldsymbol{u}_{ij} + \boldsymbol{\eta}_i,\ \boldsymbol{U}_j^\top \boldsymbol{U}_j = \boldsymbol{I}_d \\ \boldsymbol{u}_{ij} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d),\ \boldsymbol{\eta}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_L). \end{aligned} \tag{5.1.1}$$

For the linear case ($\boldsymbol{\mu}_j = \boldsymbol{0}$), in SAP K-subspaces we take $\Xi = 0.1$ and $\alpha = 0$. This is due to the nature of the model that produces the data. Note that in the Gaussian model the majority of the data lie in the intersection of the subspaces. Consequently, we encourage the algorithm to take points from the intersection of the subspaces in order to estimate each subspace.

On the other hand, the clusters in the affine case ($\boldsymbol{\mu}_j \neq \boldsymbol{0}$) are usually well separated. Therefore, we take $\Xi = 0.3$ and $\alpha = 0.5$ to prohibit a subspace from fitting more than one cluster.

Table 1: Mean and median percentage of misclassified points that follow the noisy degenerate spherical Gaussian model with $\sigma = 0.05$ and 5% outliers (Linear subspaces)

Algorithms	K-subspaces	RANSAC	SLBF	SSC	SAP K-subspaces
2^2 in $\mathbb{R}^4$					
Mean	10.2	1.7	3.7	27.5	1.7
Median	1.5	1.0	2.6	21.6	1.0
4^2 in $\mathbb{R}^6$					
Mean	9.2	0.4	1.9	37.1	1.1
Median	2.2	0.2	1.4	39.6	0.5
10^2 in $\mathbb{R}^{15}$					
Mean	8.0	6.4	1.1	46.8	1.7
Median	1.9	0.4	0.7	47.2	0.0
$(4, 5, 6)$ in $\mathbb{R}^{10}$					
Mean	24.0	0.4	1.4	52.9	9.0
Median	25.3	0.3	1.3	54.2	0.2

Table 2: Mean and median percentage of misclassified points that follow the noisy degenerate spherical Gaussian model with $\sigma = 0.05$ and 30% outliers (Linear subspaces)

Algorithms	K-subspaces	RANSAC	SLBF	SSC	SAP K-subspaces
2^2 in $\mathbb{R}^4$					
Mean	26.5	1.2	3.8	33.4	5.1
Median	27.1	1.0	1.5	38.0	1.0
4^2 in $\mathbb{R}^6$					
Mean	27.6	0.5	4.6	37.4	1.4
Median	28.2	0.5	2.4	37.9	0.6
10^2 in $\mathbb{R}^{15}$					
Mean	31.1	20.4	1.3	43.5	13.3
Median	33.5	11.6	1.0	44.6	0.0
$(4, 5, 6)$ in $\mathbb{R}^{10}$					
Mean	41.2	7.6	2.3	51.3	36.8
Median	42.5	0.3	1.8	51.6	33.5
$(4, 5, 6)$ in $\mathbb{R}^{10}$ *					
Mean	39.6	5.9	1.6	50.0	11.6
Median	39.7	0.4	1.4	50.3	0.7

* the score from the experiments with $\hat{K} = 250$ (generally $\hat{K} = 3 * K$)

Table 3: Mean and median percentage of misclassified points that follow the uniform ball model with $\sigma = 0.05$ and 5% outliers (Affine subspaces)

Algorithms	K-subspaces	RANSAC	SLBF	SSC	SAP K-subspaces
2^2 in $\mathbb{R}^4$					
Mean	14.13	3.0	0.3	3.5	8.7
Median	0.4	2.7	0.0	1.4	0.0
4^2 in $\mathbb{R}^6$					
Mean	15.8	2.6	0.2	2.7	9.6
Median	2.9	2.5	0.0	0.7	0.1
10^2 in $\mathbb{R}^{15}$					
Mean	21.6	13.7	0.0	0.1	8.7
Median	2.5	3.7	0.0	0.0	0.0
$(4, 5, 6)$ in $\mathbb{R}^{10}$					
Mean	24.5	0.9	0.0	0.1	10.6
Median	33.3	0.8	0.0	0.0	0.0

Table 4: Mean and median percentage of misclassified points that follow the uniform ball model with $\sigma = 0.05$ and 30% outliers (Affine subspaces)

Algorithms	K-subspaces	RANSAC	SLBF	SSC	SAP K-subspaces
2^2 in $\mathbb{R}^4$					
Mean	13.4	1.7	0.0	2.4	11.7
Median	10.0	1.5	0.0	1.7	5.0
4^2 in $\mathbb{R}^6$					
Mean	28.3	2.1	0.1	1.3	30.5
Median	31.0	2.0	0.0	0.6	32.7
10^2 in $\mathbb{R}^{15}$					
Mean	46.6	14.6	0.0	0.0	47.9
Median	50.0	8.9	0.0	0.0	50.0
$(4, 5, 6)$ in $\mathbb{R}^{10}$					
Mean	35.0	7.1	0.0	0.5	37.9
Median	33.4	0.5	0.0	0.2	33.3

The tables above show the average and the median misclassification error per experiment with fixed number of outliers. At first glance, it is apparent that the problem of the K-affine subspaces is easier than that of the K-linear subspaces for all algorithms except for SAP K-subspaces which is a possibilistic one.

As we increase the number of outliers (Tables 2 and 4), K-subspaces struggles to uncover the underline structure of the data set. This is due to the lack of a mechanism that ignores outliers. In contrast, SAP K-subspaces achieves far better performance owing to the way it updates the compatility degrees. From the experiments of Table 4, we conclude that in affine subspaces RANSAC is more robust than SAP K-subspaces. Nonetheless, as the number of the subspaces and their corresponding dimension increase, RANSAC fails to detect the clusters since the probability of getting d inliers becomes smaller.

Both K-subspaces and SAP K-subspaces are sensitive to initialization. However, SAP K-subspaces is even more sensitive due to the parameters Ξ and η_j. Many badly initialized subspaces are eliminated either because of zero dimension error or because their cardinality becomes less than their dimension. As a result, SAP K-subspaces is not always able to estimate all the clusters. Nevertheless, it manages to find at least one group of points and it marks the rest of the data set as unassigned. Hence, in the experiments we expect to have lower median than average misclassification error.

Increasing the number of the initial (linear) subspaces using the PFI method, helps the algorithm to detect the true number of clusters. In the last experiment of Table 2 we have a data set of 600 observations that we further corrupt it with 180 outliers. What stands out in this experiment is that when we increase the number of the initial subspaces from 6 to 250, the misclassification error falls under the half of its previous value. It is worth mentioning that from the very first update of the clusters, there have been left under 20 of them. This is due to the way we eliminate the clusters. A point is assigned only to one cluster. Therefore, many groups are left empty and the best subspaces are kept from the algorithm for further improvement.

An increasement of $\hat{K}$ in the experiments of Tables 1 and 2, would bring even better results. On the contrary, this is not the case for the experiments of Tables 3 and 4. In those experiments, we initialize the subspaces via SPCM. We know that in the affine case the clusters are well separated. Therefore, if SPCM succeeds to detect the dense areas-parts of the subspaces, SAP K-subspaces will uncover the true structure of the data set.

Unfortunately in Table 4, we see that for an increased number of outliers the results are not satisfactory.

SLBF and SSC achieve a remarkable result in the K-affine subspaces problem. However, SSC does not perform that well in the K-linear subspaces problem. That is because of the clusters as a desired condition for SSC is that the subspaces are uniformly distributed, independent and disjoint.

5.2 Clustering results on motion segmentation data

Motion Segmentation (MS) refers to the task of extracting and clustering point trajectories (pixels) of a video sequence. In MS, different clusters represent different rigit-body motions. Those point trajectories can be described using the corresponding camera projection model. One of the most known camera projection models is the *affine camera model* in which each single rigit-body motion lives in an affine subspace of dimension at most three.

The proposed algorithm is tested on the Hopkins 155 database of motion segmentation, available in *http://www.vision.jhu.edu/data/hopkins155*. The database contains 155 data sets with two to three noisy clusters (motions). Data sets are devided into three main categories: (a) *checkerboard sequences* (104 data sets) containing objects where their motion tranjectories lie on independent 3-dimensional affine subspaces, (b) *traffic sequences* (38 data sets) containing moving cars where their respected trajectories lie on independent 2-dimensional subspaces and (c) *articulated sequences* (13 data sets) containing tranjectories of objects which lie on dependent affine subspaces.

In contrast to the synthetic experiments, the clusters do not have the same cardinality. Especially in the case of three motions, all the data sets form clusters whose cardinalities differ significantly (except for the data set articulated in articulated sequences). Another challenge is the existence of clusters with extremely small cardinality. As a result, SPCM is not able to find all the clusters during initialization. Therefore, we decided to test SAP K-subspaces only with the two motions data sets.

To all the experiments $\Xi = 0.7$ and $\alpha = 0.5$ except for the articulated sequences of which $\Xi = 0.1$, since the subspaces are dependent.

(a) 2R3RTCRT (b) cars1 (c) arm

Figure 6: Sample images from some sequences of the Hopkins 155 database with tracked points superimposed

Figure (a) corresponds to checkerboard sequences, figure (b) corresponds to traffic sequences and figure (c) corresponds to articulated sequences. Source: [23]

Table 5: Mean and median percentage of misclassified points in Hopkins 155 data set (two motions)

	Checkerboard		Traffic		Articulated	
	Mean	Median	Mean	Median	Mean	Median
K-subspaces(3,-)	11.51	2.0	8.1	2.6	10.6	5.4
RANSAC(4,5)	15.0	6.9	12.5	7.5	12.3	7.5
SLBF(3,-)	1.3	0.0	0.2	0.0	0.4	0.0
SSC(-,4)	3.9	0.0	0.1	0.0	0.5	0.0
SAP K-subspaces(3,-)	17.3	18.2	5.3	0.0	6.7	3.9

Table 6: Total computation time in Hopkins 155 data set (two motions)

K-subspaces(3,-)	RANSAC(4,5)	SLBF(3,-)	SSC(-,4)	SAP K-subspaces(3,-)
3.7311s	9.3018s	329.2515s	102.4807s	237.3102s

In the tables above, the parameters (d,p) next to the name of the algorithm indicate the dimension of the subspace (initial for SAP K-subspaces) d and the dimension of the projection p.

Considering Table 5, it is apparent that SAP K-subspaces struggles to detect the clusters of the checkerboard sequences. The reason is that the checkerboard sequences of two motions include the data sets with the most diversed in terms of population groups. K-subspaces does not require the same size of groups. Therefore, its performance in checkerboard sequences is superior than that of SAP K-subspaces. The results of both SLBF and SSC are outstanding. It appears that the inliers threshold, estimated as $\eta =$ Mean$(\{\eta_j\}_{j=1}^{K}) \times 0.8$, is not a proper choice for RANSAC as it has the worst performance.

K-subspaces is the fastest among all the considered algorithms (Table 6). SAP K-subspaces, although an iterative algorithm as K-subspaces, in each iteration has to run AIRLS to estimate the dimension of the subspaces. Thus, it needs more time to converge. However, it is faster than SLBF that is the most time consuming algorithm of all.

6. CONCLUSION

In the present thesis, a novel possibilistic subspace clustering algorithm, called SAP K-subspaces, is proposed. The algorithm is able to estimate the true number of equally populated clusters, as well as the variance and the dimension of the corresponding subspaces. SAP K-subspaces can also be viewed as the subspace extension of SPCM [24]. The proposed algorithm imposes a sparsity constraint on the degrees of compatibility of each point with a cluster exactly as SPCM does. As a result, SAP K-subspaces exhibits immunity to noise and outliers. To adapt the subspaces dimensions $\{d_j\}_{j=1}^{K}$'s, the algorithm adopts a low-rank dimensionality reduction technique presented in [10]. The algorithm is initialized through PFI in case of dependent clusters and through SPCM in case of independent clusters. It turned out, that the ability of SAP K-subspaces to estimate the number of clusters highly depends on a parameter Ξ that controls the degree of the imposed sparsity on the degrees of compatibility of a data point with the clusters. When clustering dependent groups of points, the values of Ξ which unveil the data structure are usually smaller than those used in independent ones. The same appears to hold for parameter α (the threshold which specifies which points will contribute to the estimation of the subspace of a cluster). To fine-tune the regularizer λ_2 (the parameter that reduces the dimension of the subspace associated with a certain cluster), we use a penalized version of the model's RMSE which encourages the choice of dimensionality reduced subspaces and discourages that of overfitted models (models with more subspaces than the actual ones). The algorithm can effectively detect subspaces that intersect with each other and correspond to non-uniformly distributed groups of points. However, those groups have to be of the same size, since the same λ_2 parameter is used for each cluster. When this criterion is met, SAP K-subspaces exhibits in most cases state-of-the-art performance.

7. FUTURE WORK

Interesting avenues of research, include the creation of a scalable possibilistic subspace clustering algorithm which reduces the dimension of the subspaces without imposing constraints on the cardinality of their associated clusters. Lastly, there is a demand for creating reliable model selection criteria which take into account models produced by algorithms which adapt both the number of subspaces and their dimensions.

ABBREVIATIONS - ACRONYMS

ADMM	Alternating Direction Method of Multipliers
AIRLS	Alternating Iteratively Reweighted Least Squares
BSUM	Block Successive Upper-bound Minimization
eq.	equation
i.e.	from the latin phrase ”id est”, that is
MS	Motion Segmentation
NRE	Normalized Reconstruction Error
PCA	Principal Components Analysis
PFI	Probabilistic Farthest Insertion
RANSAC	RANdom SAmple Consensus
RMSE	Root Mean Square Error
SLBF	Spectral Local Best-Fit Flats
SPCM	Sparse Possibilistic C-Means
SSC	Sparse Subspace Clustering
w.r.t.	with respect to

APPENDIX A. DEFINITIONS AND NOTATION

In this appendix, definitions and notations used in the thesis are presented. More of them will be introduced in the relevant chapters as necessary.

A.1 Notation

A.1.1 Sets

X	Data set
C	Cluster, set of data points
S	Subspace, set of data points

A.1.2 Scalars

L	Ambient space dimension
d_j	Dimension of the subspace S_j
N	The cardinality of the data set
N_j	The cardinality of S_j
K	Number of clusters
w_{ij}	Degree of compatibility of a data point $\boldsymbol{x}_i$ with the j-cluster
η_j	Variance of the noise associated with the subspace S_j

A.1.3 Vectors

Vectors are represented as boldface lowercase letters.

$\boldsymbol{x}_i$	$\in \mathbb{R}^L$, denotes the i data point
$\boldsymbol{\eta}_i$	$\in \mathbb{R}^L$, denotes the noise of $\boldsymbol{x}_i$
$\boldsymbol{y}_{ij}$	$\in \mathbb{R}^d$, denotes the orthogonal projection of $\boldsymbol{x}_i$ onto S_j
$\boldsymbol{\mu}_j$	$\in \mathbb{R}^L$, denotes the displacement vector of S_j
$\mathbf{1}$	All-ones vector of any size
$\mathbf{0}$	All-zeros vector of any size

A.1.4 Matrices

Matrices are denoted as boldface uppercase letters.

$\boldsymbol{X}$	$\in \mathbb{R}^{L\times N}$, whose columns are the data vectors $\boldsymbol{x}_i$
$\boldsymbol{X}_j$	$\in \mathbb{R}^{L\times N_j}$, whose columns are the data vectors $\boldsymbol{x}_i \in C_j$
$\boldsymbol{W}$	$\in \mathbb{R}^{N\times K}$, whose (i,j) element is the degree of compatibility w_{ij}
$\boldsymbol{U}_j$	$\in \mathbb{R}^{L\times d}$, with columns d vectors that form a basis for S_j
$\boldsymbol{Y}_j$	$\in \mathbb{R}^{d\times N}$, with columns the orthogonal projections of the data onto S_j
$\boldsymbol{I}_m$	The $m \times m$ identity matrix
$\mathbf{0}_m$	The $m \times m$ zero matrix

A.1.5 Norms

Let $\boldsymbol{\alpha}_j$ a $1 \times m$ vector.

- $\|\boldsymbol{\alpha}_j\|_r = (\sum_{i=1}^{m} |\alpha_{ij}|^r)^{1/r}$ (ℓ_r norm of a vector)
- $\|\boldsymbol{\alpha}_j\|_r^p = (\sum_{i=1}^{m} |\alpha_{ij}|^r)^{p/r}$ (ℓ_r^p norm of a vector)

Let $\boldsymbol{A}$ a $m \times n$ matrix.

- $\|\boldsymbol{A}\|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{m} \alpha_{ij}^2} = \sqrt{\sum_{j=1}^{n} \|\boldsymbol{\alpha}_j\|_2^2}$ (Forbenious norm of a matrix)
- $\|\boldsymbol{A}\|_{r,p} = (\sum_{j=1}^{n} \|\boldsymbol{\alpha}_j\|_r^p)^{1/p}$ ($\ell_{r,p}$ norm of a matrix)
- $\|\boldsymbol{A}\|_{r,p}^r = (\sum_{j=1}^{n} \|\boldsymbol{\alpha}_j\|_r^p)^{r/p}$ ($\ell_{r,p}^r$ norm of a matrix)

A.2 Definitions

A.2.1 Linear Algebra

Definition A.2.1. Let V a non-empty set on which addition and scalar multiplication is defined. In addition, let the field of scalars to be that of the real numbers. Then $(V, +, \cdot)$ or simply V is called a (real) *vector space*. The elements of V are called vectors if the following conditions are satisfied:

1. If $\boldsymbol{x}_1, \boldsymbol{x}_2 \in V$ then $\boldsymbol{x}_1 + \boldsymbol{x}_2 \in V$ (closure +)

2. $\boldsymbol{x}_1 + \boldsymbol{x}_2 = \boldsymbol{x}_2 + \boldsymbol{x}_1, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in V$ (commutative law)
3. $\boldsymbol{x}_1 + (\boldsymbol{x}_2 + \boldsymbol{x}_3) = (\boldsymbol{x}_1 + \boldsymbol{x}_2) + \boldsymbol{x}_3, \forall \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \in V$ (associative law)
4. There exists one element, denoted by $\mathbf{0}_V$, such that $\boldsymbol{x}_1 + \mathbf{0}_V = \boldsymbol{x}_1, \forall \boldsymbol{x}_1 \in V$ (additive identity)
5. $\forall \boldsymbol{x}_1 \in V$, there exists an inversed such that $\boldsymbol{x}_1 + (-\boldsymbol{x}_1) = \mathbf{0}_V$ (additive inversed)
6. For $\kappa \in \mathbb{R}, \boldsymbol{x}_1 \in V$, it is $\kappa \boldsymbol{x}_1 \in V$ (closure $\cdot$)
7. $\kappa(\boldsymbol{x}_1 + \boldsymbol{x}_2) = \kappa \boldsymbol{x}_1 + \kappa \boldsymbol{x}_2, \forall \kappa \in \mathbb{R}, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in V$ (distributive law)
8. $(\kappa + \lambda)\boldsymbol{x}_1 = \kappa \boldsymbol{x}_1 + \lambda \boldsymbol{x}_1, \forall \kappa, \lambda \in \mathbb{R}, \forall \boldsymbol{x}_1 \in V$ (distributive law)
9. $\kappa(\lambda \boldsymbol{x}_1) = (\kappa\lambda)\boldsymbol{x}_1 \forall \lambda, \kappa \in \mathbb{R}, \forall \boldsymbol{x}_1 \in V$ (associative law)
10. $\mathbf{1}\boldsymbol{x}_1 = \boldsymbol{x}_1, \forall \boldsymbol{x}_1 \in V$ (unitary law)

Definition A.2.2. If $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$ are vectors of a vector space V and $\kappa_1, \kappa_2, ..., \kappa_N$ scalars, then $\sum_{i=1}^{N} \kappa_i \boldsymbol{x}_i$ is called a *linear combination* of $\boldsymbol{x}_i$'s.

Definition A.2.3. A set of points $\{\boldsymbol{x}_i\}_{i=1}^{N}$ is said to be *linearly dependent* if there exist scalars $\kappa_0, ..., \kappa_N$ not all zero such that $\sum_{i=0}^{N} \kappa_i \boldsymbol{x}_i = \mathbf{0}_V$. If the scalars are all zero, the set is said to be *linearly independent*.

Definition A.2.4. Let S be a subset of a vector space V ($S \subseteq V$). If S has identical definitions of vector addition and scalar multiplication with that of V, then S is called a *subspace* of V.

Definition A.2.5. K *linear* subspaces of a vector space V are said to be *independent* if the dimension of their sum is equal to the sum of their dimensions,

$$\text{dim}(S_1 \oplus S_2 \oplus ... \oplus S_K) = \text{dim}(S_1) + \text{dim}(S_2) + ... + \text{dim}(S_K).$$

Definition A.2.6. K *linear* subspaces of a vector space V are said to be *disjoint* if every pair of subspaces intersect only at the origin,

$$S_i \cap S_j = \{\mathbf{0}_V\}, \text{ where } i, j = 1, 2..., K \text{ and } i \neq j.$$

Remark. If $S_i \cap S_j = \{\mathbf{0}_V\}$ then $\text{dim}(S_i \oplus S_j) = \text{dim}(S_i) + \text{dim}(S_j)$.

Remark. Independent subspaces are disjoint too but the opposite is not always true (see figure 2).

Definition A.2.7. Let $U = \{u_i\}_{i=1}^{\kappa}$ a non-empty finite subset of a vector space V. In addition, let S be a subspace of V composed by the linear combinations of U. Then, S is said to be *spanned* by U, span $\{U\} = S$.

Definition A.2.8. Let U be a subset of a subspace S. Then, U is said to be a *basis* of S if the following conditions are satisfied:

1. span $\{U\} = S$
2. The elements of U are linearly independent

Remark. $\mathbf{0}_V \notin U$, otherwise the elements of U are linearly dependent.

Remark. There can be more than one bases of S.

A.2.2 Affine Geometry

Definition A.2.9. A subset A of a vector space V is called an *affine subspace* if either $A = \emptyset$ or $A = \boldsymbol{\mu} + L$, where $\boldsymbol{\mu} \in V$ and L a linear subspace of V.

Definition A.2.10. The *homogeneous coordinates* of a point $x = [x_{11}, x_{12}, ..., x_{1L}]^\top \in \mathbb{R}^L$ are defined as $[x_{11}, x_{12}, ..., x_{1L}, 1]^\top$.

Definition A.2.11. If $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$ are vectors of a vector space V and $\kappa_1, \kappa_2, ..., \kappa_N$ scalars with $\sum_{i=1}^{N} \kappa_i = 1$, then $\sum_{i=1}^{N} \kappa_i \boldsymbol{x}_i$ is called an *affine combination* of $\boldsymbol{x}_i$'s.

Definition A.2.12. A set of points $\{\boldsymbol{x}_i\}_{i=1}^{N}$ is said to be *affinely dependent* if there exist scalars $\kappa_0, ..., \kappa_N$ with $\sum_{i=1}^{N} \kappa_i = 1$ not all zero such that $\sum_{i=0}^{N} \kappa_i \boldsymbol{x}_i = \mathbf{0}_V$. If the scalars are all zero, the set is said to be *affinely independent*.

Remark. Linear independence implies affine independence and affine dependence implies linear dependence, but not vice versa.

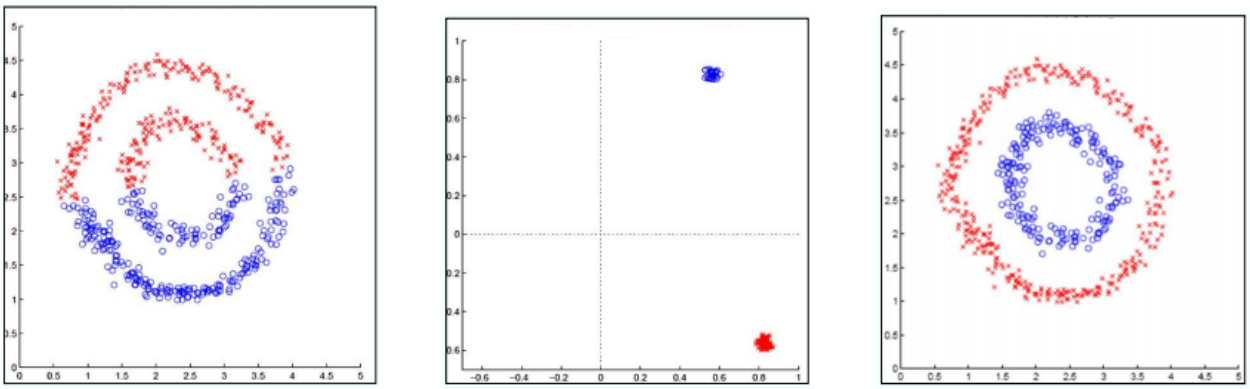
Definition A.2.13. K *affine* subspaces are said to be *independent* if the corresponding K linear subspaces using the homogeneous coordinates are.

Definition A.2.14. K *affine* subspaces are said to be *disjoint* if the corresponding K linear subspaces using the homogeneous coordinates are.

APPENDIX B. SPECTRAL CLUSTERING

The aim of this subsection is to provide background information about the technique used by the algorithms presented in 3.4 and 3.5. For a complete theoritical review see [16].

Spectral clustering is a technique suitable for high dimensional data and they can reveal clusters of any shape, provided that they are not intersected. Basically, what we do with spectral clustering is that we map the data into a new space (usually of a lower dimension), where the clusters (hopefully) become compact and hyperspherically-shaped. Then, we apply suitable clustering techniques, such as the K-means [17].



(a) (b) (c)

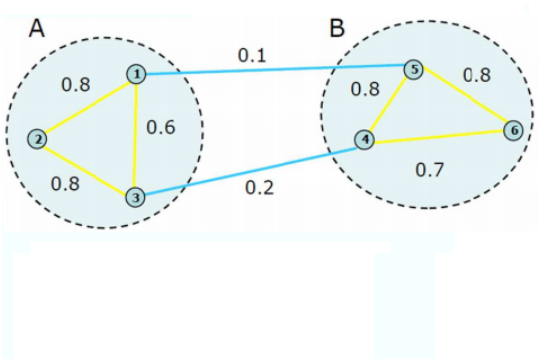
Figure 7: Spectral vs Compact Clustering

2-dimensional data set that forms two clusters. (a) K-means' clustering results when applied to the original data set. (b) The data K-means is applied. (c) The spectral clustering results in the original space. Source: [18]

More precisely, the first step, is to build a *similarity graph* of the data. A similarity graph can be weighted or unweighted. It is common to use a weighted graph, denoted by $G = (V, E)$, where V is a set of N nodes corresponding to N data points and $E \subseteq V \times V$ is a set of edges that connect a subset of those nodes (see figure 8).

From the weights of the graph, the matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ called the *weighted adjacency matrix* of the graph or simply the *affinity matrix* is created. Each element of the affinity matrix, measures the similarity between two data points. For instance, α_{ij} measures the similarity degree between point i and point j. In other words, is an indication of whether those two points should be assigned to the same group or not. An $\alpha_{ij} = 0$, indicates that the points should be assigned to different groups. The matrix $\boldsymbol{A}$ is symmetric ($\alpha_{ij} = \alpha_{ji}$) and the range of its values depends on the similarity measure. A typical similarity measure is

the Gaussian kernel function, $\alpha_{ij} = \exp(\frac{-\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma^2})$ where σ determines the width of the kernel.



(a)

	X_1	X_2	X_3	X_4	X_5	X_6
X_1	0	0.8	0.6	0	0.1	0
X_2	0.8	0	0.8	0	0	0
X_3	0.6	0.8	0	0.2	0	0
X_4	0	0	0.2	0	0.8	0.7
X_5	0.1	0	0	0.8	0	0.8
X_6	0	0	0	0.7	0.8	0

(b)

Figure 8: Similarity graph and affinity matrix

(a) The similarity graph of a 2-dimensional data set that forms two clusters. (b) The associated affinity matrix. Source: [18]

Given the affinity matrix, the *Laplacian matrix* or *graph Laplacian* $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is created. There are many different definitions in the literature about what kind of matrices should be called like this. However, one may think of the graph Laplacian as a matrix that allows us to perform clustering based on its properties. A popular choise is the *normalized Laplacian matrix*. Defined as,

$$\boldsymbol{L} = \boldsymbol{D}^{-1/2}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-1/2}, \text{ where } \boldsymbol{D} = \text{diag}(d_1, d_2, ..., d_N), \; d_i = \sum_{i=1}^{N} \alpha_{ij}. \tag{B.0.1}$$

To continue, $k \ll N$ eigenvectors of $\boldsymbol{L}$ are chosen and stacked into a matrix $\boldsymbol{U} \in \mathbb{R}^{N \times k}$. For the normalized Laplacian, those are the first k eigenvectors from the bottom which correspond to the k smallest eigenvalues. Finally, a compact clustering technique is applied to the rows of $\boldsymbol{U}$ (each row corresponds to a data point).

REFERENCES

[1] Pankaj K. Agarwal and Nabil H. Mustafa. K-means projective clustering. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, page 155–165, New York, NY, USA, 2004. Association for Computing Machinery.

[2] Edoardo Amaldi, Viggo Kann, et al. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

[3] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[5] P. S. Bradley, O. L. Mangasarian, and Panos Pardalos. k-plane clustering, 1999.

[6] Guangliang Chen and Gilad Lerman. Spectral curvature clustering (scc). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[7] Ondrej Drbohlav. Hungarian algorithm for linear sum assignment problem.

[8] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Paris V Giampouras, Athanasios A Rontogiannis, and Konstantinos D Koutroumbas. Alternating iteratively reweighted least squares minimization for low-rank matrix factorization. *IEEE Transactions on Signal Processing*, 67(2):490–503, 2018.

[11] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[12] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l_1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1(4):606–617, 2007.

[13] Raghuram Krishnapuram and James M Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[14] Connor Lane, Benjamin Haeffele, and René Vidal. Adaptive online k-subspaces with cooperative reinitialization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[15] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[16] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[17] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[18] Klas Nordberg. Lecture notes in visual representations for machine learning.

[19] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[20] Mahdi Soltanolkotabi, Ehsan Elhamifar, Emmanuel J Candes, et al. Robust subspace clustering. *The Annals of Statistics*, 42(2):669–699, 2014.

[21] Paul Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

[22] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[23] René Vidal, Yi Ma, and S Shankar Sastry. Principal component analysis. In *Generalized principal component analysis*, pages 25–62. Springer, 2016.

[24] Spyridoula D Xenaki, Konstantinos D Koutroumbas, and Athanasios A Rontogiannis. Sparsity-aware possibilistic clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 24(6):1611–1626, 2016.

[25] Allen Y Yang, Shankar R Rao, and Yi Ma. Robust statistical estimation and segmentation of multiple subspaces. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 99–99. IEEE, 2006.

[26] Teng Zhang, Arthur Szlam, Yi Wang, and Gilad Lerman. Hybrid linear modeling via local best-fit flats. *International journal of computer vision*, 100(3):217–240, 2012.