Splitting the spectrum: stratifying autism using cluster analysis on predicted normative patterns

Daan In de Braekt (6537391)
Supervisor: Daan van Rooij
2nd Supervisor: Rianne van Lambalgen

February 2022

BSc Thesis (15ECTS)
Artificial Intelligence (Kunstmatige Intelligentie)
Utrecht University

Abstract

Early diagnosis can be of huge importance for the treatment of ASD symptoms in children. Unfortunately, biological markers that reliably classify this neurodevelopmental disorder are still lacking, partly due to the vast heterogeneity in both symptom manifestation and severity. Therefore, Machine Learning approaches using neuroimaging techniques such as structural MRI (sMRI) have become an increasingly useful tool for analyzing underlying brain structures and patterns that may be helpful in finding such biomarkers. One of those approaches adopted is the stratification of the data of ASD cohort, such that the resulting subcohorts are more homogeneous and easier to analyse.

In this thesis, we aimed to stratify sMRI data from people with ASD using two distinct approaches of cluster analysis. We first proved the theoretical prerequisites of both methods, and show their respective optimization objectives. Then, we adopted a normative modelling approach to quantify deviations of ASD subjects from a 'normal' pattern that was inferred using the data from neurotypical subjects. This normative model functioned as the input to the clustering algorithms. As the MRI data was segmented into different clinical measures and brain regions, we then analysed the resulting clusters statistically and identified brain regions associated with each cluster.

Across both methods, we found that clusters correlating with ASD severity deviated most in the regions connecting the occipital, temporal and parietal lobe with the frontal lobe. Non-correlated clusters were associated mostly with regions in the frontal lobe. Moreover, cortical volume measures did not show up as highly deviating variables in any of the found clusters.

Contents

1 Introduction **4**

2 k-means clustering **7**
2.1 Setting definitions . . . 7
2.2 The k-means objective . . . 7
2.3 The k-means algorithm . . . 9
2.3.1 Lloyd's algorithm . . . 9
2.3.2 Some properties of Lloyd's algorithm . . . 9
2.3.3 k-means++ . . . 11
2.4 Determining k . . . 12
2.4.1 Elbow method . . . 12
2.4.2 Silhouette . . . 12

3 Spectral clustering **15**
3.1 Graph construction . . . 15
3.2 Graph partitioning . . . 16
3.2.1 The mincut problem . . . 16
3.2.2 Graph Laplacians . . . 16
3.2.3 Reformulating Ncut . . . 18
3.2.4 Relaxing our objective . . . 20
3.3 Proof of theorem 3.8 . . . 20
3.4 The clustering algorithm . . . 23
3.4.1 From continuous to discrete . . . 23
3.4.2 Retrieving a clustering . . . 23
3.4.3 The algorithm . . . 23

4 (H)DBSCAN **24**
4.1 Challenging assumptions . . . 24
4.1.1 The k-means assumptions . . . 24
4.1.2 A density-based approach . . . 25
4.1.3 An estimated density-based approach . . . 26
4.2 Algorithms . . . 26
4.2.1 DBSCAN . . . 26
4.2.2 Varying ε . . . 27
4.2.3 HDBSCAN . . . 28
4.2.4 Minimum cluster size . . . 29
4.3 Optimizing HDBSCAN . . . 30
4.3.1 Excess of Mass . . . 30
4.3.2 Formulating an optimization objective . . . 31
4.3.3 Optimizing our objective . . . 32

5 Experiment **34**
5.1 The ENIGMA-ASD dataset . . . 35
5.1.1 Data preprocessing . . . 35
5.2 Normative modelling . . . 35
5.2.1 Predicting the normal patterns . . . 36
5.2.2 Choosing a kernel . . . 36
5.2.3 Quantifying deviation . . . 37
5.3 Spectral clustering . . . 37
5.4 HDBSCAN . . . 38
5.5 (Statistical) cluster analysis . . . 38
5.6 Python . . . 38

6 Results **40**
6.1 Spectral clustering . . . 40
6.1.1 Cosine affinity . . . 40

6.1.2 Squared Euclidean affinity . . . 40
6.2 HDBSCAN . . . 41
6.3 General observations . . . 42

7 Conclusion **46**

A Mathematical prerequisites **54**
A.1 Linear algebra . . . 54
A.1.1 Vector space . . . 54
A.1.2 Matrices . . . 56
A.2 Relations . . . 58
A.3 Graphs . . . 59

B Gaussian Process Regression **60**

C Effect of `n_neighbors` on mean silhouette score **62**

D TOP rankings for all clusters **63**

E Creation of toy dataset **67**

1 Introduction

Artificial Intelligence is in the midst of revolutionizing modern healthcare, as data-driven (Machine Learning) technologies are contributing to the diagnosis of patients, making recommendations about better treatments and predictions about patients' future health [71]. A major benefit of these approaches is their ability to detect subtle patterns in data overlooked by human researchers, and the possibility to combine information from multiple sources efficiently [93]. As such, they show promise for exploring biological concepts that may be explained by various factors rather than one specific process. One of many research fields in which such patterns are investigated is that of neurodevelopmental disorders, such as Attention Deficit Hyperactivity Disorder and Autism Spectrum Disorder.

Autism Spectrum Disorder (ASD), commonly just called 'autism', is a neurodevelopmental disorder that is marked by sensory atypicalities, restricted and repetitive behaviors as well as persistent deficits in social communication and interaction [6]. Notable symptoms include, among other things, making little or inconsistent eye contact with others, repeating certain behaviors, being upset by slight changes in a routine, having lasting interests in specific topics, and experiencing difficulties with having conversations [59]. ASD is a lifelong condition that develops early in life, with symptoms usually emerging between 12 and 24 months of age. Around 2% of children in the Netherlands are estimated to be affected [86], and prevalence among males is around three times higher compared to females.

An early diagnosis can make a huge difference in the lives of children with ASD: a growing body of evidence suggests that early intervention and childhood ASD therapies can have a (significant) positive impact on communicative, sensory motor, social and cognitive skills [29]. Unfortunately, current methods of classification are mostly symptom-based, relying on parents or caregivers to look for developmental milestones and contacting doctors if they recognize any concerning signs. In particular, no objective biological measure or test has been found to reliably classify ASD. Therefore, lots of time and effort has been put into finding such 'biomarker'. Examples include performance during behavioral tasks (e.g. tracking eye movements as in [70]), nutritional supplements during pregnancy (e.g. vitamine B12 concentrations at birth [66]), and immune activation (e.g. viral or bacterial maternal infections during pregnancy [62]). In particular, research using structural MRI [78], a neuroimaging technique, has gained popularity as a powerful tool to investigate structural brain alterations in people with ASD [18].

A large barrier in all this research, however, has been the wild variety in combinations of symptoms that arise and the severity in which they manifest themselves among people with ASD. In fact, differences between different parts of the 'spectrum' are so strong, researchers have suggested it might be more accurate to speak about many 'autisms' instead of just one [26, 35]. This heterogeneity is seen in many neurodevelopmental disorders [53], and makes understanding and analysing any data gathered from people with ASD more complex [35].

A possible data-driven approach to chart this heterogeneity is to stratify (or: subdivide) the ASD cohort into different subcohorts that are internally more homogeneous and easier to analyse. The predominant framework for this is a ML approach called cluster analysis - or 'clustering' in short. It is the task of of grouping together data into a number of groups, called clusters, such that data belonging to the same group is more similar to each other than to data in other groups [9].

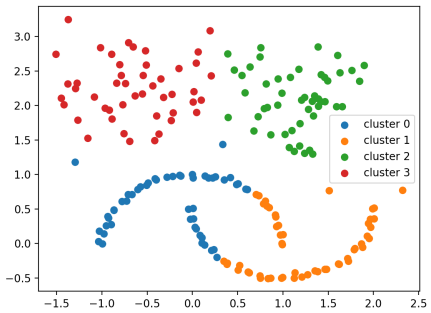
This definition of cluster analysis is, undoubtedly, rather vague. Hence, many interpretations have prevailed over the years, all different in their notions of how and when data can be considered similar, how clusters should look, and consequently, how a good clustering is constructed. Given their notion of 'good', of course.

What all these approaches have in common, however, is that they rely only on the input data, without extra pieces of information such as labels or predefined loss functions that need to be optimized in some manner. In other words, there is no 'best possible clustering' available for the data to which the algorithms can compare their solution. As such, cluster analysis can be considered a form of unsupervised Machine Learning, and is therefore especially attractive for exploratory data analysis, for seeing patterns in the data not known before, and, hopefully, for gaining insight into the structure and distribution of the data that is researched.

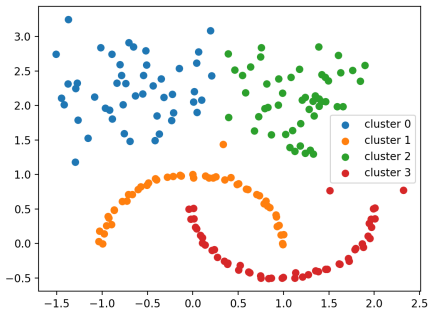
This all brings us to this thesis. Our first goal for this piece will be to stratify structural MRI data from people with ASD – or rather, a normative model derived from this data – into clinical subgroups, using cluster analysis. The data we will use stem from the ENIGMA-ASD working group (http://enigma.ini.usc.edu/ongoing/enigma-asd-working-group/) and is segmented into different brain regions of interest

[93], as we will further discuss in section 5. Hence, our second goal will be to identify brain regions of interest associated with the resulting clusters and draw connections with ASD symptoms and/or severity.

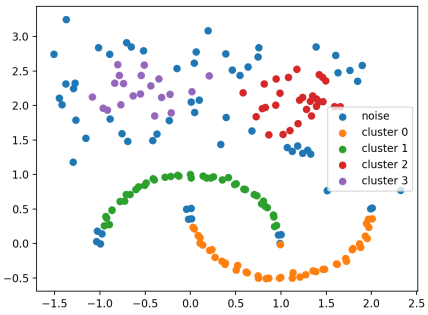
To get started, we will first formalize the notion of a clustering (in section 2.1), and then see three different clustering approaches. Of these three, k-means clustering will be the first – also in section 2 –, and is without doubt the most well known algorithm in the clustering world. It assumes that every cluster is defined by some middle point, called a centroid, and consequently clusters its input data to the clusters belonging to the centroids that are closest to those data. It is easy to implement and understand, and, in most implementations, very quick to converge. It also, generally, does not work very well.[1] We will use k-means not as a standalone tool, but as the last step in our next algorithm, called spectral clustering (section 3). In spectral clustering, we view the data as a graph such that the vertices represent the points and the edges model their pairwise (dis)similarities. Then, this graph is transformed into a lower-dimensional representation such that k-means has fewer problems detecting useful clusters. Lastly, we discuss a somewhat newer approach to clustering called density-based clustering in section 4, which assumes clusters are found in regions where data points are concentrated, separated by regions of sparsity. Other than spectral and k-means, density-based clustering algorithms have the possibility to detect outliers, and can end up not classifying some data to any cluster at all. HDBSCAN[2], the density-based algorithm that we will discuss, has the added benefit of being able to detect clusters of different densities, and having few (i.e. one) parameters to tweak.



(a) k-means



(b) Spectral clustering



(c) HDBSCAN

Figure 1.1: Sample clusterings of a generated two-dimensional toy data set, using said methods (generated with code provided in appendix E). All points are contained in the data set and are colored according to the cluster they belong to. Notice that the blue points in the HDBSCAN clustering are considered 'noise' by the algorithm.

As inherent to its unsupervised nature, the absence of a data-specific loss function prevents us from making simple quantitive claims about the workings of and the comparison between different clustering approaches – although this has been tried[3]. To still get a feel for how these approaches function, researchers often look

[1] See [79] for some examples.

[2] Hierarchical Density-Based Spatial Clustering of Applications with Noise.

[3] There exist external validation measures such as the (adjusted) Rand Index [69] that evaluate clusterings based on a chosen comparison clustering; these tend to overlook the fact that often, multiple alternative clusterings may be equally useful [33]. On the other hand,

at simpler low-dimensional data and analyse the results by eye. As for our three methods, sample clusterings using a toy data set are shown in figure 1.1. Although this data is undoubtedly messy, we can see two moon-shaped chains of points that clearly belong together. Furthermore, we see a cloud of points in the 'upper half' of the data with some slightly denser regions in both the top left and top right – although this last claim may already be up to debate. Looking at the sample clusterings, we see that both spectral clustering and HDBSCAN follow this intuition, while k-means seems to favor distinct, linear borders between the clusters[4]. Furthermore, we see HDBSCAN also classifies some noise. Given the fact it may detect clusters of distinct shapes and strutures, the freedom of parameters it provides and its ability to detect outliers if possible, we hypothesize HDBSCAN will have a slight advantage over spectral clustering in the high-dimensional setting, in the sense that its results will be the most homogenous and therefore the most straightforward to interpret.

The structure of the three clustering sections are similar: after exploring the main ideas behind the approach, we introduce an objective function that we aim to optimize, and try to 'solve' this optimization. Then, we describe what the actual algorithm looks like that computes the clustering given the input data.

In section 5, we describe our actual experiment. After preprocessing the data, we will construct a so called normative model of the data, using Gaussian Process Regression. In a normative model, we quantify the deviation of these values from what is considered 'normal', rather than analyzing the absolute values. More concretely, for every person with ASD in the data, we predict a normal value based on the data of all neurotypical (non-ASD) persons in the data, and determine the difference between the actual measured value and that prediction. Furthermore, we will run experiments to optimize the parameters needed for the clustering methods.

In section 6, we will present the resulting clusterings of our normative model and interpret them based on their statistical properties as well as the brain regions that characterize them most. We end this chapter by sharing some general observations based on the found clusterings.

In section 7, we will conclude and reflect upon our experiment, and look forward to possible directions for future research.

Notation Across this thesis, we will denote vectors using lower case bold face (e.g. $\mathbf{s}$), matrices by capital (e.g. $\mathbf{M}$) and scalars by non-bold lower case letters (e.g. d). We denote by $\mathbf{1}^n$ the n-dimensional identity vector, i.e. $\mathbf{1}^n := [1, \cdots, 1]^T$, and use $\mathbf{1}^{n\times m}$ to mean the $n \times m$-dimensional identity matrix, i.e.

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}.$$

Analogously, the n-dimensional zero vector and the $n \times m$-dimensional zero matrix are denoted $\mathbf{0}^n$ and $\mathbf{0}^{n\times m}$ respectively. To refer to a specific entry of a matrix $\mathbf{A}$ – e.g. the entry in the i-th row and j-th column – we write $(\mathbf{A})_{ij}$.

Let $A := \{a_1, \dots, a_n\}$ be a set and let B be some other set such that $B \subseteq A$. If not defined otherwise (e.g. in section 3.2.3), the indicator vector of B, denoted $\mathbf{1}^A(B)$, is defined as

$$\mathbf{1}^A(B) := [x_1, \cdots, x_n]^T \text{ , where } x_i := \begin{cases} 1 \text{ if } a_i \in B \\ 0 \text{ if } a_i \notin B \end{cases}.$$

If A is clear from context, we will write $\mathbf{1}(B)$.

Lastly, for two vectors $\mathbf{a}$ and $\mathbf{b}$, we denote their Euclidean distance by $\|\mathbf{a}-\mathbf{b}\|$ and their respective norms by $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$. We have included a short recap of definitions and properties of some mathematical concepts we will use in appendix A.

internal validation measures rely only on intrinsic properties of the data, and are mostly mathematical formulations that encompass some implicit intuition on how clusterings should typically look [82].

[4] In fact, this is inherent to the k-means algorithm, see [79, 56].

2 k-means clustering

2.1 Setting definitions

A partition is a division of a set into subsets such that all elements of the set are contained in exactly one subset. Formally, let $S \subset \mathbb{R}^d$ be a set consisting of elements $\mathbf{s}_1, \ldots, \mathbf{s}_n$. Let furthermore $\Gamma := \{C_1, \ldots, C_k\}$ be a family of sets of size $k \geq 1$. Then Γ is a finite partition of S if and only if the following conditions hold:

1. $\bigcup_{l=1}^{k} C_l = S$, i.e. the union of all sets in C is equal to S,
2. For any $1 \leq i < j \leq k$, we have $C_i \cap C_j = \emptyset$, i.e. no element of S belongs to two different sets in S, and
3. $\emptyset \notin \Gamma$, i.e. the empty set does not belong to Γ.

The number of possible partitions of a set – the cardinality of Γ – is given by the Bell number [36], which can be calculated recursively by

$$B_0 = 1; B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} \cdot B_n.$$

For a set of 20 elements, this already adds up to 51724158235372 possibilities.

In the context of this thesis, we call a Γ a clustering of S, and $C_1, \ldots, C_k$ are called its clusters. The set S is called the (input) data set and its elements are called the (input) data points. Let furthermore, for any set S, $\Pi(S)$ be the set of all finite partitions (i.e. valid clusterings) of S.

This definition of a clustering is undoubtedly loose, which allows for a plethora of clustering approaches.[5] In particular, no conditions are set on which clusterings should be picked. This task is handled entirely by clustering objectives. A clustering objective is a form of mathematical optimization, quantifying how 'good' a clustering is according to that specific clustering approach.

A minimal form of such objective looks as follows:

$$\begin{aligned} \underset{\Gamma}{\text{minimize}} \quad & f(S, \Gamma), \\ \text{subject to} \quad & \Gamma \in \Pi(S). \end{aligned}$$

for some function $f : S \times \Gamma \to \mathbb{R}$ that depends (at least) on S and Γ, which we call the objective function. The function f captures any underlying assumptions made on cluster quality. Consequently, clusterings that minimize these objectives tend to conform to exactly these assumptions. A clustering Γ that optimizes f is called optimal. Furthermore, we say two objectives f and f' are equivalent if an optimal clustering wrt. f is also optimal wrt. f', and vice versa.

2.2 The k-means objective

The k-means objective is based on the assumption that a good clustering minimizes the sum of pairwise squared Euclidean distances between the points of a cluster, i.e. it optimizes the following objective:

$$\begin{aligned} \underset{\Gamma}{\text{minimize}} \quad & f(\Gamma, S) := \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \left\| \mathbf{s} - \mathbf{s}' \right\|^2, \\ \text{subject to} \quad & \Gamma \in \Pi(S). \end{aligned}$$

Another way of formulating this is to assume that all k clusters are represented by k 'middle points' called centroids and that a good clustering has k centroids that minimize the squared distances from the data points to the centroid of their respective clusters. We define a centroid $\mathbf{m}_l \in \mathbb{R}^d$ of a cluster C_l as

$$\mathbf{m}_l := \frac{1}{|C_l|} \sum_{\mathbf{s} \in C_l} \mathbf{s}.$$

[5] In fact, this definition has been loosened even further to allow points to belong to more than one cluster, or to belong to a cluster by some degree. This is called fuzzy clustering; see [96] for an excellent discussion of the corresponding fuzzy k-means algorithm. We will not go in depth here.

Then, the reformulated k-means objective looks as follows:

$$\begin{aligned} \underset{\Gamma}{\text{minimize}} \quad & f'(\Gamma, S, \mathbf{m}_1, \ldots, \mathbf{m}_k) := \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l} \|\mathbf{s} - \mathbf{m}_l\|^2, \\ \text{subject to} \quad & \Gamma \in \Pi(S), \\ & \mathbf{m}_1, \ldots, \mathbf{m}_k \in \mathbb{R}^d. \end{aligned}$$

At first glance, this raises the question: which objective is the *right* k-means objective? Thankfully, we need not worry, as we will now show that the two objectives are in fact equivalent.

Theorem 2.1. *Let $S \subset \mathbb{R}^d$, $\Gamma = \{C_1, \ldots, C_k\}$ a clustering of S and $\mathbf{m}_1, \ldots, \mathbf{m}_k$ be the centroids of the clusters in Γ. Then*

$$\sum_{l=1}^{k} \frac{1}{2|C_l|} \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2 = \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l} \|\mathbf{s} - \mathbf{m}_l\|^2.$$

Proof. For any $1 \leq l \leq k$, we have

$$\begin{aligned} \sum_{\mathbf{s} \in C_l} \|\mathbf{s} - \mathbf{m}_l\|^2 &= \sum_{\mathbf{s} \in C_l} (\|\mathbf{m}_l\|^2 - 2\mathbf{s}^T \mathbf{m}_l + \|\mathbf{s}\|^2) && \text{(lemma A.1)} \\ &= |C_l| \cdot \|\mathbf{m}_l\|^2 - 2 \sum_{\mathbf{s} \in C_l} \mathbf{s}^T \mathbf{m}_l + \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 \\ &= |C_l| \cdot \|\mathbf{m}_l\|^2 - 2|C_l| \cdot \|\mathbf{m}_l\|^2 + \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 && \text{(substituting } \sum_{\mathbf{s} \in C_l} \mathbf{s}_i^T = |C_l| \cdot \mathbf{m}_l^T) \\ &= -|C_l| \cdot \|\mathbf{m}_l\|^2 + \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2. \end{aligned}$$

Furthermore, it holds that

$$\begin{aligned} \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2 &= \sum_{\mathbf{s}, \mathbf{s}' \in C_l} (\|\mathbf{s}\|^2 - 2\mathbf{s}^T \mathbf{s}' + \left\|\mathbf{s}'\right\|^2) && \text{(lemma A.1)} \\ &= \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \|\mathbf{s}\|^2 - 2 \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \mathbf{s}^T \mathbf{s}' + \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \|\mathbf{s}\|^2 \\ &= |C_l| \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 - 2 \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \mathbf{s}^T \mathbf{s}' + |C_l| \sum_{\mathbf{s}' \in C_l} \left\|\mathbf{s}'\right\|^2 \\ &= 2|C_l| \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 - 2 \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \mathbf{s}^T \mathbf{s}' \\ &= 2|C_l| \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 - 2|C_l| \sum_{\mathbf{s} \in C_l} \mathbf{s}^T \mathbf{m}_l && \text{(substituting } \sum_{\mathbf{s}' \in C_l} \mathbf{s}' = |C_l| \cdot \mathbf{m}_l) \\ &= 2|C_l| \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2 - 2|C_l|^2 \|\mathbf{m}_l\|^2 && \text{(substituting } \sum_{\mathbf{s} \in C_l} \mathbf{s}^T = |C_l| \cdot \mathbf{m}_l^T) \\ &= 2|C_l|(-|C_l| \cdot \|\mathbf{m}_l\|^2 + \sum_{\mathbf{s} \in C_l} \|\mathbf{s}\|^2). \end{aligned}$$

Combining the two results, we get

$$\frac{1}{2|C_l|} \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2 = \sum_{\mathbf{s} \in C_l} \|\mathbf{s} - \mathbf{m}_l\|^2.$$

Hence,

$$\sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l} \|\mathbf{s} - \mathbf{m}_l\|^2 = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{\mathbf{s}, \mathbf{s}' \in C_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2,$$

which is what we wanted to show. □

Corollary 2.2. *The objectives f and f' are equivalent.*

Proof. Follows directly from theorem 2.1. □

We will refer to *the* k-means objective instead of specifying whether it is f or f'. Based on corollary 2.2, this should not lead to confusion. Note that the centroids do not have to correspond to any datapoint in S, but may be any point in $\mathbb{R}^d$. There exists another clustering objective, called k-medoid, that requires centroids to actually belong to the input data set. Analogous to the notions of mean and median, k-medoid is considered more robust to outliers than the mean, and is more flexible concerning the choice of distance measures [72].

2.3 The k-means algorithm

As it turns out, minimizing f or f' is NP-hard, even for smaller instances like $k = 2$ [48], meaning there does not exist an algorithm - yet - running in polynomial time that finds a clustering that is optimal wrt. the objective.

For a possible salvage, imagine that we knew the optimal set of centroids $\mathbf{m}_1^*, \ldots, \mathbf{m}_k^*$. Then, we could easily find the optimal clustering by assigning

$$\mathbf{s} \in C_l \iff \arg\min_i \left\|\mathbf{s} - \mathbf{m}_i^*\right\|^2 = l.$$

This suggests we can implement a more heuristic approach to approximate the optimal value for f', by repeatedly picking new centroids, and assigning the points accordingly, hoping to get a value for f' that is lower than the one before. This is exactly the intuition behind Lloyd's algorithm.

2.3.1 Lloyd's algorithm

Lloyd's algorithm is used so often that, in many cases, the term k-means clustering refers to the convergence of this algorithm rather than to the clustering that optimizes the 'actual' objective f' [72]. In this simple iterative process, we start by randomly selecting – 'seeding' – initial centroids in the dataspace $\mathbb{R}^d$. We then assign all points to the cluster closest to some centroid, and subsequently update the centroids to be the mean vectors of these clusters. The algorithm stops when it is converged, i.e. when the clusters have not changed upon one iteration.

Algorithm 1 Lloyd's algorithm

1: **input:** $S \subset \mathbb{R}^d$, k.
2: **initialize:** starting centroids $\mathbf{m}_1^0, \ldots, \mathbf{m}_k^0 \in \mathbb{R}^d$, chosen uniformly at random.[6]
3: **while** not converged:
4: assign each data point to the cluster defined by the closest centroid, i.e.

$$\mathbf{s} \in C_l^{(t+1)} \iff \left\|\mathbf{s} - \mathbf{m}_l^{(t)}\right\|^2 \leq \left\|\mathbf{s} - \mathbf{m}_{l'}^{(t)}\right\|^2 \text{ for all } 1 \leq l' \leq k.$$

5: update the centroids by

$$\mathbf{m}_l^{(t+1)} = \frac{1}{|C_l^{(t+1)}|} \sum_{\mathbf{s} \in C_l^{(t+1)}} \mathbf{s}.$$

6: **return** clusters $C_1, \ldots, C_k$.

2.3.2 Some properties of Lloyd's algorithm

Lemma 2.3. *Each iteration of algorithm 1 does not increase the objective function f'.*

[6] For variable x, $x^{(t)}$ refers to the value of x at iteration t

Proof. Let $C^{(t)}, \mathbf{m}_1^{(t)}, \ldots, \mathbf{m}_k^{(t)}$ and $C^{(t+1)}, \mathbf{m}_1^{(t+1)}, \ldots, \mathbf{m}_k^{(t+1)}$ be the clusterings and corresponding centroids achieved after t and $t+1$ iterations respectively. Each iteration of algorithm 1 consists of two parts – step 4 and 5 – and our proof does too. Ultimately, we will show that

$$\begin{aligned} f'(C^{(t)}, S, \mathbf{m}_1^{(t)}, \ldots, \mathbf{m}_k^{(t)}) &= \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l^{(t)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2 \\ &\geq \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l^{(t+1)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2 \qquad (1) \\ &\geq \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l^{(t+1)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t+1)} \right\|^2 \qquad (2) \\ &= f'(C^{(t+1)}, S, \mathbf{m}_1^{(t+1)}, \ldots, \mathbf{m}_k^{(t+1)}), \end{aligned}$$

(1): In step 4 of algorithm 1, all $\mathbf{s} \in S$ are assigned to the cluster of their closest centroid, such that.

$$\sum_{\mathbf{s} \in C_l^{(t+1)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2 \leq \sum_{\mathbf{s} \in C_l^{(t)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2.$$

Thus, it follows directly that

$$\sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l^{(t+1)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2 \leq \sum_{l=1}^{k} \sum_{\mathbf{s} \in C_l^{(t)}} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2.$$

(2): In step 5 of algorithm 1, the centroids are updated as

$$\mathbf{m}_l^{(t+1)} = \frac{1}{|C_l^{(t+1)}|} \sum_{\mathbf{s} \in C_l^{(t+1)}} \mathbf{s}.$$

Let in the following $\mathbf{s} \in C_l^{(t+1)}$. With some algebraic manipulations, we get

$$\begin{aligned} \left\| \mathbf{s} - \mathbf{m}_l^{(t)} \right\|^2 &= \left\| \mathbf{s} - \mathbf{m}_l^{(t+1)} + \mathbf{m}_l^{(t+1)} - \mathbf{m}_l^{(t)} \right\|^2 \\ &= \left\| \mathbf{s} - \mathbf{m}_l^{(t+1)} \right\|^2 + \left\| \mathbf{m}_l^{(t+1)} - \mathbf{m}_l^{(t)} \right\|^2 + 2 \cdot (\mathbf{s} - \mathbf{m}_l^{(t+1)})^T (\mathbf{m}_l^{(t+1)} - \mathbf{m}_l^{(t)}) && \text{(lemma A.1)} \\ &\geq \left\| \mathbf{s} - \mathbf{m}_l^{(t+1)} \right\|^2 + 2 \cdot (\mathbf{s} - \mathbf{m}_l^{(t+1)})^T (\mathbf{m}_l^{(t+1)} - \mathbf{m}_l^{(t)}). && \text{(since } \left\| \mathbf{m}_l^{(t+1)} - \mathbf{m}_l^{(t)} \right\|^2 \geq 0) \end{aligned}$$

Applying this inequality in a summation over all elements in $C_l^{(t+1)}$ and substituting

$$\mathbf{m}_l^{(t+1)} := \frac{1}{|C_l^{(t+1)}|} \sum_{\mathbf{s} \in C_l^{(t+1)}} \mathbf{s}$$

– indicated by † – yields

$$\begin{aligned}
\sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t)}\right\|^2 &\geq \sum_{\mathbf{s}\in C_l^{(t+1)}} \left(\left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot(\mathbf{s}-\mathbf{m}_l^{(t+1)})^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)})\right) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot \sum_{\mathbf{s}\in C_l^{(t+1)}} (\mathbf{s}-\mathbf{m}_l^{(t+1)})^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot (\sum_{\mathbf{s}\in C_l^{(t+1)}} (\mathbf{s}-\mathbf{m}_l^{(t+1)}))^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) && \text{(lemma A.9)} \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot (\sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{s} - \sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{m}_l^{(t+1)})^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot (\sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{s} - \sum_{\mathbf{s}\in C_l^{(t+1)}} \frac{1}{|C_l^{(t+1)}|} \sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{s})^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) && (\dagger) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot (\sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{s} - |C_l^{(t+1)}|\cdot \frac{1}{|C_l^{(t+1)}|} \sum_{\mathbf{s}\in C_l^{(t+1)}} \mathbf{s})^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2 + 2\cdot \mathbf{0}^T(\mathbf{m}_l^{(t+1)}-\mathbf{m}_l^{(t)}) \\
&= \sum_{\mathbf{s}\in C_l^{(t+1)}} \left\|\mathbf{s}-\mathbf{m}_l^{(t+1)}\right\|^2,
\end{aligned}$$

which is what we wanted to show.

□

Theorem 2.4. *If $|S| < \infty$, the k-means algorithm terminates after a finite amount of iterations.*

Proof. From lemma 2.3, we know that in each iteration f' does not increase. Furthermore, since S is finite, there exist only a finite amount of partitions of S (namely, B_n). Therefore, the algorithm has to terminate after a finite amount of iterations. □

Even though the algorithm converges, there is no guarantee about its accuracy. In particular, if f'_{OPT} is the value for f' corresponding to the (globally) optimal clustering and f'_L the value corresponding to the clustering found by algorithm 1, then f'_L/f'_{OPT} is unbounded, even when n and k are fixed [5]. In other words, Lloyd's algorithm can produce clusterings that are arbitrarily bad.

Because of this bad approximation, Lloyd's algorithm is often run multiple times in practice, picking new starting centroids each time to add a component of randomisation. Then, the clustering that minimizes f' over all runs is chosen as the end result. Other approaches to introduce randomization include swapping points between clusters after convergence, splitting up clusters with high within-cluster distances or merging nearby clusters together [79].

2.3.3 k-means++

A particularly clever seeding method is proposed in [5], and starts off by picking a point $\mathbf{s} \in S$ uniformly at random as the first initial centroid. Then, the next centroid is assigned from S with probability proportional to its squared distance to the already assigned centroids, i.e. data points that lie further away from the other initial centroids are preferred. The idea is that the initial clusters are spread out as much as possible, to yield a better convergence afterwards.

At any point, let $D(s)$ denote the (squared Euclidean) distance from $\mathbf{s}$ to the closest of already selected starting centroids (so $D(s) = 0$ if $\mathbf{s}$ is itself a centroid). Then, the seeding algorithm is described in algorithm 2.

It is shown in [5] that k-means++ is $O(\log k)$-competitive, i.e. if f'_{++} is the value for f' corresponding to the clustering found by algorithm 2, then $f'_{OPT} \leq O(\log k)\cdot f'_{++}$, which is a dramatic improvement over the

Algorithm 2 k-means++ (adapted from [5])

1: **input:** $S \subset \mathbb{R}^n$, k
2: **initialize:** starting centroid $\mathbf{m}_1^0 \in S$, chosen uniformly at random.
3: **while** less than k centroids are chosen:
4: choose the next centroid $\mathbf{m}_l^0$, selecting $\mathbf{m}_l^0 = \mathbf{s}_l \in S$ with probability

$$\frac{D(\mathbf{s}_l)}{\sum_{\mathbf{s}\in S} D(\mathbf{s})}.$$

5: proceed as in the original k-means algorithm (step 3-6).

standard Lloyd's algorithm. The new initialization method also speeds up convergence considerably (up to 90% in some theoretic datasets).

2.4 Determining k

A major assumption in any implementation of k-means clustering is that the number of clusters k is known a priori. While in some applications this number may be implicitly or explicitly entailed in the data set (e.g. image compression using k-means, where k controls the compression rate [19]. In general, however, determining the amount of clusters can be a difficult task in its own right. We will therefore need some method to pick a good value for k given our data.

A naive approach would be to iterate over various values for k and picking the one that minimizes f', essentially 'brute forcing' the best possible clustering. It can however be easily seen that increasing k will always lower f': if $k = |S|$, then $f'_{OPT} = 0$ holds trivially since any $\mathbf{s} \in S$ can function as its own cluster.

In particular, the higher the value for k, the lesser our algorithm will show the underlying patterns in the data we are looking for, but rather overfits on the given data [2]. The other extreme – $k = 1$ – returns a single lump cluster which contains exactly as little new information. Consequently, there has to be some trade-off between a higher clustering accuracy (i.e. minimizing f' by adding more clusters) and preventing overfitting (i.e. a smaller value for k).

2.4.1 Elbow method

In a common rule-of-thumb translation of this trade-off called the Elbow method, we try to choose the optimal k by looking at a graph plot of the values for f'. The idea is that the first few clusters will improve the fit significantly, since these clusters show some actual underlying patterns in the data, until some optimal value k^*, after which the added information by increasing k drops a lot due to overfitting. This 'cut-off point' is assumed more or less elbow-shaped: a steep drop in f' (for $1 \leq k \leq k^*$) followed by slow decreases for $k > k^*$. In figure 2.1, an example scatter plot is given. In practice, elbows may not be as sharp, and may be hard to unambiguously identify. We also have no way to ensure this visually inspected value for k will actually give us any underlying insights about the dataset.

2.4.2 Silhouette

The silhouette measure encapsulates the notion that good clusterings ensure all clusters are tightly connected (i.e. within-cluster-distances are low) while being well-separated from all other clusters (i.e. high between-cluster-distances). The best value for k according to silhouette is the one that ensures this best.

Let $\Gamma = \{C_1, \ldots, C_k\} \in \Pi(S)$. For any $\mathbf{s} \in C_l$, we define

$$A(\mathbf{s}) := \frac{1}{|C_l|} \sum_{\mathbf{s}'\in C_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2$$

and

$$B(\mathbf{s}) := \min_{l\neq l'} \frac{1}{|C_{l'}|} \sum_{\mathbf{s}'\in C'_l} \left\|\mathbf{s} - \mathbf{s}'\right\|^2.$$

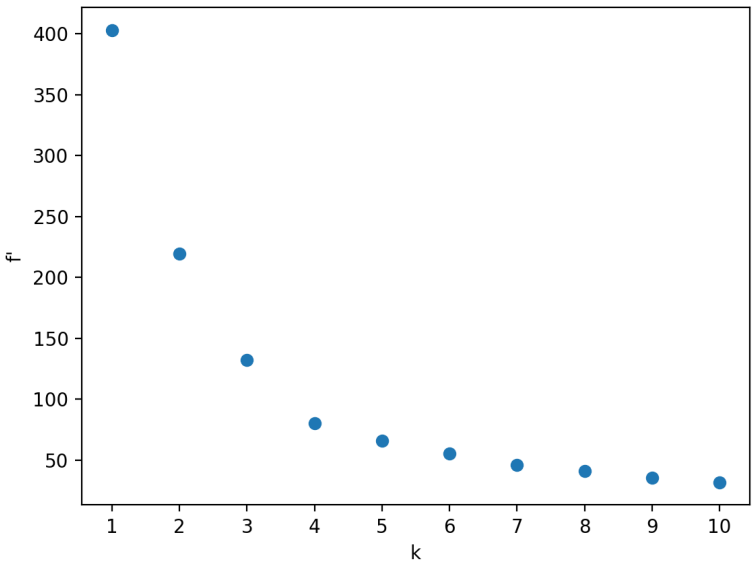


Figure 2.1: Elbow method applied to the toy data set introduced in section 1. The steepest declines for f' are reached for $k = 2$ and $k = 3$.

As $A(\mathbf{s})$ computes squared distances between points in the same cluster as $\mathbf{s}$, it serves as a measure of tightness within a cluster. $B(\mathbf{s})$ on the other hand denotes the sum of distances between $\mathbf{s}$ and the points of the nearest neighboring cluster in Γ. Hence, it serves as a measure of separation between clusters. Now, the silhouette of $\mathbf{s}$ is defined as

$$\text{Sil}(\mathbf{s}) := \frac{B(\mathbf{s}) - A(\mathbf{s})}{\max(A(\mathbf{s}), B(\mathbf{s}))}.$$

It is easily seen that $-1 \leq \text{Sil}(\mathbf{s}) \leq 1$. A good clustering ideally sees $A(\mathbf{s})$ way smaller than $B(\mathbf{s})$ (i.e. $\mathbf{s}$ fits well in cluster C_l, and poorly fits its neighboring cluster), which means $\text{Sil}(\mathbf{s})$ will get closer to 1. On the contrary, $\text{Sil}(\mathbf{s})$ will get closer to -1 if $\mathbf{s}$ better fits to its neighboring cluster C_l' than to its current cluster.

We can extend the notion of silhouette to a measure of the entire clustering, by taking the mean of $\text{Sil}(s)$ over all $\mathbf{s} \in S$. The best value for k maximizes this mean, i.e.

$$k^* := \arg\max_k \overline{\text{Sil}}(C^{(k)}),$$

where $C^{(k)}$ denotes the clustering produced by a clustering algorithm (possibly after some improvement heuristics) with k clusters, and $\overline{\text{Sil}}(C^{(k)}) = \frac{1}{|S|} \sum_{\mathbf{s} \in C^{(k)}} \text{Sil}(\mathbf{s})$.

In theorem 2.1, we have seen minimizing within-cluster squared distances is equivalent to minimizing distances to its centroid. While the original definition of silhouette uses within-cluster-distances, this means we can equivalently implement silhouette using distances from points to their centroids:

$$A'(\mathbf{s}) = \|\mathbf{s} - \mathbf{m}_l\|^2$$
$$B'(\mathbf{s}) = \min_{l \neq l'} \left\|\mathbf{s} - \mathbf{m}_l'\right\|^2$$
$$\text{Sil}'(\mathbf{s}) = \frac{B'(\mathbf{s}) - A'(\mathbf{s})}{\max(A'(\mathbf{s}), B'(\mathbf{s}))},$$

for some $\mathbf{s} \in S$ that is in cluster C_l. This version has been dubbed 'simplified silhouette' due to its computational benefits over the original silhouette (the centroids have already been computed when running k-means) [90].

Lemma 2.5. *After applying k-means clustering, $\text{Sil}'(\mathbf{s}) \geq 0$ for all $\mathbf{s} \in S$.*

Proof. Step 4 of algorithm 1 ensures that $\mathbf{s}$ is classified to the cluster with the closest possible centroid, i.e. $\|\mathbf{s}-\mathbf{m}_l\|^2 \leq \left\|\mathbf{s}-\mathbf{m}'_l\right\|^2$ for any $1 \leq l' \leq k$. Therefore, $B'(\mathbf{s}) \geq A'(\mathbf{s})$, whence $\mathrm{Sil}'(\mathbf{s}) \geq 0$. □

By lemma 2.5 we can simplify Sil' to

$$\mathrm{Sil}'(\mathbf{s}) = 1 - \frac{A'(\mathbf{s})}{B'(\mathbf{s})}.$$

3 Spectral clustering

An alternative way of approaching a clustering problem is to view the data as a graph, such that each vertex represents a point and the edges model the similarity between those points. As such, the choice of finding a partition of the input can be reframed as the problem of finding a 'cut' of the graph into a number – k – of connected components (see appendix A.3). The idea in spectral clustering is then to transform the graph data back into vector data of lower dimension – called embedding –, such that it can be clustered using a regular clustering algorithm such as k-means.

3.1 Graph construction

We again assume some input data $S \subseteq \mathbb{R}^d$. Furthermore, we pick a notion of distance $\delta : S \times S \to \mathbb{R}$, which has to fulfill following properties:

1. $\delta(\mathbf{s}_i, \mathbf{s}_i) = 0$,
2. $\delta(\mathbf{s}_i, \mathbf{s}_j) \geq 0$, and
3. $\delta(\mathbf{s}_i, \mathbf{s}_j) = \delta(\mathbf{s}_j, \mathbf{s}_i)$,

for $\mathbf{s}_i, \mathbf{s}_j \in S$. Besides the squared Euclidean distance, cosine distance is often used here. For any $\mathbf{s}_i, \mathbf{s}_j \in S$, cosine distance $\delta_{\cos}$ is defined as

$$\delta_{\cos}(\mathbf{s}_i, \mathbf{s}_j) = 1 - \cos\theta = 1 - \frac{\mathbf{s}_i^T \mathbf{s}_j}{\|\mathbf{s}_i\| \|\mathbf{s}_j\|},$$

where θ is the angle between $\mathbf{s}_i$ and $\mathbf{s}_j$. Note that the δ chosen does not have to be a proper distance metric [21]. In particular, triangle inequality may be violated (and $\delta_{\cos}$ indeed does this). From here, we construct a graph $G := (V, E, \mathbf{W})$, where the vertices $V = \{1, \ldots, n\}$ represent the input data $\mathbf{s}_1, \ldots, \mathbf{s}_n$, and the weight matrix $\mathbf{W} := (\mathbf{W})_{ij} \in \mathbb{R}^{n \times n}$ is called the affinity matrix.

There are essentially two ways to construct the set of edges E. Firstly, we can connect all vertices to each other, resulting in a complete graph encompassing all relations between all vertices (most often setting $(\mathbf{W})_{ij} := \delta(\mathbf{s}_i, \mathbf{s}_j)$ to specify similarity). Alternatively, we can choose to only connect certain edges based on some threshold on δ, resulting in a sparse matrix $\mathbf{W}$ that allows for easier interpretation and faster calculations.

As for the latter method, a common way to sparsify $\mathbf{W}$ is an m-nearest neighbor graph which connects vertices if at least one of them is an m-nearest neighbor (denoted m-nn) of the other, according to δ.[7] Edges are weighted as follows:

$$\forall i, j : (\mathbf{W})_{ij} = \begin{cases} 1 & \text{if } \mathbf{s}_i, \mathbf{s}_j \text{ are } m\text{-nn of each other} \\ 1/2 & \text{if either } \mathbf{s}_i \text{ or } \mathbf{s}_j \text{ is } m\text{-nn of the other} \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, $\mathbf{W}$ can be sparsified using an ε-neighborhood graph in which only the vertices within ε distance from each other are connected, i.e.

$$\forall i, j : (\mathbf{W})_{ij} = \begin{cases} 1 & \text{if } \mathbf{s}_j \in N(\mathbf{s}_i, \varepsilon) \\ 0 & \text{otherwise,} \end{cases}$$

where $N(\mathbf{s}, \varepsilon)$ is the set of points $\mathbf{s}'$ such that $\delta(\mathbf{s}, \mathbf{s}') \leq \varepsilon$. By property 3 of δ we only need to verify the ε-neighborhood of $\mathbf{s}_i$ to know that of $\mathbf{s}_j$. Furthermore, no matter if a complete or m-nn graph is implemented, we have $(\mathbf{W})_{ij} = (\mathbf{W})_{ji}$, meaning $\mathbf{W}$ is symmetric. Furthermore, we have $(\mathbf{W})_{ij} \geq 0$. For theoretical purposes, these are the only constraints on the construction of $\mathbf{W}$.

For subsets $A, B \subseteq V$, we define $W(A, B) := \sum_{i \in A, j \in B} (\mathbf{W})_{ij}$, and let $W(\{i\}, V)$ denote the degree of vertex i. Any edge (i, j) such that $i \in A, j \in B$ is called a crossing edge between A and B. We then define the corresponding diagonal degree matrix $\mathbf{D} := (\mathbf{D})_{ij} \in \mathbb{R}^{n \times n}$ with $(\mathbf{D})_{ii} := W(\{i\}, V)$ for $1 \leq i \leq n$ and the non-diagonal entries set to zero.

[7] The variable m is chosen instead of the usual k to avoid confusion with the number of clusters k.

3.2 Graph partitioning

As said, the intent of the spectral approach is to reformulate the problem of finding a partition of S to the problem of finding a 'cut' of G. Analogous to the intuition discussed in section 2.4.2, we ideally aim for a partition having high edge weights within components and low edge weights in between. The mincut problem is the most direct way to formalize this.

3.2.1 The mincut problem

A k-cut[8] of a graph G is a partition $A := \{A_1, \ldots, A_k\} \in \Pi(V)$ such that removing the edges between the k sets yields k connected components consisting of the sets in A. A k-mincut of G is a k-cut that minimizes the sum of the weights of these crossing edges. Formally, for a partition A of V, we want to

$$\underset{A}{\text{minimize}} \quad \text{cut}(A) := \frac{1}{2}\sum_{i=1}^{k} W(A_i, V - A_i),$$

where $\frac{1}{2}$ is added so that edge weights are not counted twice. Especially for $k = 2$, finding a mincut can be implemented easily, e.g. by using the Stoer-Wagner algorithm [76, 84] . The cut function for $k = 2$ naturally simplifies to

$$\text{cut}(A, V - A) = \frac{1}{2}(W(A, V - A) + W(V - A, A)) = W(A, V - A).$$

However, for larger k, this problem is, expectedly, NP-hard to solve [73]. Clustering methods based solely on solving this cut function have been proposed, for example by [95]. However, they noticed that this function seems to favor cutting small sets of isolated nodes off the graph, leaving the bulk of the vertices in one huge component. This is undesirable, since we like to cluster reasonably large groups of points instead of a few far-away 'outlier' clusters.

To avoid this, a new minimizing objective was introduced in [73] that besides summing up the weights of crossing edges, additionally weigths the cut function by the sizes of the subgraphs they induce. This objective is called Ncut and is defined as

$$\underset{A}{\text{minimize}} \quad \text{Ncut}(A) := \sum_{i=1}^{k} \frac{\text{cut}(A_i, V - A_i)}{\text{assoc}(A_i)},$$

where $\text{assoc}(A) := \sum_{i \in A} W(\{i\}, V) = \sum_{i \in A} (\mathbf{D})_{ii}$, or the sum of edge weights from vertices in A to all vertices in the graph. $\sum_{i=1}^{k} \frac{1}{\text{assoc}(A_i)}$ reaches its minimum if $\text{assoc}(A_i) = \text{assoc}(A_j)$ for all $1 \leq i, j \leq k$ [84]. If some component A_i in the cut is too small, $\text{assoc}(A_i)$ will be small, leading to a large Ncut summand.

However nice this result from [73] is, it is again NP-hard to solve [87]. We therefore need to relax the Ncut problem in some way to approximate the optimal value for Ncut. To do so, we will need the notion of graph Laplacians.

3.2.2 Graph Laplacians

We define the (unnormalized) graph Laplacian as $\mathbf{L} := \mathbf{D} - \mathbf{W}$. Since $\mathbf{D}$ and $\mathbf{W}$ are symmetric by definition, $\mathbf{L}$ is too. The diagnonal elements $(\mathbf{L})_{ii}$ $(1 \leq i \leq n)$ are equal to

$$(\mathbf{L})_{ii} = (\mathbf{D})_{ii} - (\mathbf{W})_{ii} = W(\{i\}, V) - (\mathbf{W})_{ii} = \sum_{\substack{i,j=1 \\ j \neq i}}^{n} (\mathbf{W})_{ij}$$

which means the diagonal elements of $\mathbf{W}$ do not influence $\mathbf{L}$. In other words, edges from a vertex to itself do not change the unnormalized graph Laplacian.

[8] Here, we do not circumvent the use of name 'k' to emphasize this variable will end up corresponding to the amount of clusters k.

Theorem 3.1. *For every* $\mathbf{u} \in \mathbb{R}^n$*, where* $\mathbf{u} := [u_1, \cdots, u_n]^T$*, we have that*

$$\mathbf{u}^T \mathbf{L}\mathbf{u} = \frac{1}{2} \sum_{i,j=1}^{n} (\mathbf{W})_{ij} (u_i - u_j)^2.$$

Proof. By distributivity of matrix multiplication and definition of $\mathbf{W}$ and $\mathbf{D}$, we have

$$\begin{aligned}
\mathbf{u}^T \mathbf{L}\mathbf{u} = \mathbf{u}^T (\mathbf{D} - \mathbf{W})\mathbf{u} &= \mathbf{u}^T \mathbf{D}\mathbf{u} - \mathbf{u}^T \mathbf{W}\mathbf{u} \\
&= \sum_{i,j=1}^{n} u_i u_j (\mathbf{D})_{ij} - \sum_{i,j=1}^{n} u_i u_j (\mathbf{W})_{ij} && \text{(see appendix A.1.1)} \\
&= \sum_{i=1}^{n} u_i^2 (\mathbf{D})_{ii} - \sum_{i,j=1}^{n} u_i u_j (\mathbf{W})_{ij} && ((\mathbf{D})_{ij} = 0 \text{ if } i \neq j) \\
&= \frac{1}{2} (2 \sum_{i=1}^{n} u_i^2 (\mathbf{D})_{ii} - 2 \sum_{i,j=1}^{n} u_i u_j (\mathbf{W})_{ij}) \\
&= \frac{1}{2} (\sum_{i=1}^{n} u_i^2 (\mathbf{D})_{ii} + \sum_{j=1}^{n} u_j^2 (\mathbf{D})_{jj} - 2 \sum_{i,j=1}^{n} u_i u_j (\mathbf{W})_{ij}) \\
&= \frac{1}{2} (\sum_{i,j=1}^{n} u_i^2 (\mathbf{W})_{ij} + \sum_{i,j=1}^{n} u_j^2 (\mathbf{W})_{ij} - 2 \sum_{i,j=1}^{n} u_i u_j (\mathbf{W})_{ij}) && ((\mathbf{D})_{ii} = W(\{i\}, V)) \\
&= \frac{1}{2} \sum_{i,j=1}^{n} (\mathbf{W})_{ij} (u_i^2 - 2u_i u_j + u_j^2) \\
&= \frac{1}{2} \sum_{i,j=1}^{n} (\mathbf{W})_{ij} (u_i - u_j)^2.
\end{aligned}$$

□

Corollary 3.2. $\mathbf{L}$ *is positive semi-definite, i.e.* $\mathbf{u}^T \mathbf{L}\mathbf{u} \geq 0$ *for every* $\mathbf{u} \in \mathbb{R}^n$.

Proof. Follows from theorem 3.1 since $(\mathbf{W})_{ij} \geq 0$. □

Theorem 3.3. *L has real valued eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Proof. To show 0 is an eigenvalue of $\mathbf{L}$, observe that

$$\mathbf{L}\mathbf{1}^n = (\mathbf{D} - \mathbf{W})\mathbf{1}^n = \mathbf{0}^n = 0 \cdot \mathbf{1}^n.$$

The eigenvector corresponding to λ_0 is $\mathbf{1}^n$. From lemma A.8, we know $\mathbf{L}$ can not have negative eigenvalues, whence $\lambda_0 = 0$ is the smallest eigenvalue of $\mathbf{L}$. □

Theorem 3.4. *Let* $A := \{A_1, \ldots, A_k\}$ *be a k-cut of G. Then,* $dim(E_{\mathbf{M}}(0)) = k$ *(see also appendix A.1.2). Furthermore, the indicator vectors* $\mathbf{1}^V(A_1), \ldots, \mathbf{1}^V(A_k)$ *span the eigenspace.*

Proof. If the graph is fully connected, we have $k = 1$. Let $\mathbf{v} := [v_1, \cdots, v_n]^T$ be some eigenvector of $\mathbf{L}$ with eigenvalue 0. From theorem 3.1, we know that

$$0 = \mathbf{v}^T \mathbf{L}\mathbf{v} = \sum_{i,j=1}^{n} (\mathbf{W})_{ij} (v_i - v_j)^2.$$

Since both $(\mathbf{W})_{ij}$ and $(v_i - v_j)^2$ are non-negative, we have $(\mathbf{W})_{ij} = 0$ or $(v_i - v_j)^2 = 0$ for all $1 \leq i, j \leq n$. If vertices i and j are connected, we thus have $v_i = v_j$. This means that for all vertices that are connected by a path in the graph, their entry in $\mathbf{v}$ should be the same. Therefore, all vectors in the eigenspace of 0 can be written as $\mathbf{1}^n$ multiplied by a scalar. Since G only has one connected component - itself -, $\mathbf{1}^n = \mathbf{1}^V(A_1)$ is also its indicator vector.

Now let $k > 1$. Let $\mathbf{W}_l, \mathbf{D}_l$ be the adjacency and degree submatrix of $\mathbf{W}$ and $\mathbf{D}$ for the l-th connected component consisting of only the rows and columns corresponding to vertices in this l-th component. We can then rewrite $\mathbf{L}$ as the block diagonal matrix

$$\begin{bmatrix} \mathbf{L}_1 & 0 & \dots & 0 \\ 0 & \mathbf{L}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mathbf{L}_k \end{bmatrix},$$

where $\mathbf{L}_l = \mathbf{D}_l - \mathbf{W}_l$ is the unnormalized graph Laplacian of the l-th connected component. That eigenvectors of $\mathbf{L}$ are just the eigenvectors of all $\mathbf{L}_l$, filled with 0 at all other positions [42]. Since $\mathbf{L}_l$ is a graph Laplacian of a graph with one connected component (by construction), we can use our insight from $k = 1$ and see that every $\mathbf{L}_l$ has eigenvalue 0, with the only eigenvector needed to span the eigenspace being the vector with $\mathbf{1}$ at the place of $\mathbf{L}_l$, and 0 at the other positions, i.e. $\mathbf{1}^V(A_l)$. Since the connected components are mutually disjoint, their indicator vectors are linearly independent, hence, the dimension of the eigenspace of eigenvalue 0 is equal to the number of connected components k. □

3.2.3 Reformulating Ncut

We will now reformulate the Ncut objective using $\mathbf{L}$. Let A again be some k-cut of G. We define indicator vectors $\mathbf{1}^V_*(A_j) := \left[a_{1j}, \cdots, a_{nj}\right]^T$ for $1 \le j \le k$ such that

$$a_{ij} = \begin{cases} \frac{1}{\sqrt{\text{assoc}(A_j)}} & \text{if } i \in A_j \\ 0 & \text{if } i \notin A_j \end{cases}.$$

We also define the corresponding indicator matrix $\mathbf{M} \in \mathbb{R}^{n \times k}$ having vectors $\mathbf{1}^V_*(A_1), \dots, \mathbf{1}^V_*(A_k)$ as its columns.

Theorem 3.5. $\mathbf{M}^T\mathbf{M} = \mathbf{1}^{k\times k}$.

Proof. Since A is a partition, we have $A_j \cap A_{j'} = \emptyset$ if $j \neq j'$, meaning $(\mathbf{1}^V_*(A_j))^T \mathbf{1}^V_*(A_{j'}) = 0$. Furthermore, all indicator vectors are unit vectors by definition. Hence, the vectors $\mathbf{1}^V_*(A_1), \dots, \mathbf{1}^V_*(A_k)$ form an orthonormal set of vectors (see appendix A.1.1). In turn, this means that $\mathbf{M}$ is an orthogonal matrix, i.e.

$$\mathbf{M}^T\mathbf{M} = \mathbf{1}^{k\times k}.$$

□

Theorem 3.6. $\mathbf{M}^T\mathbf{D}\mathbf{M} = \mathbf{1}^{k\times k}$.

Proof. Let $\mathbf{d} = [(\mathbf{D})_{11}, \cdots, (\mathbf{D})_{nn}]^T$ be the degree vector containing only the diagonal entries of $\mathbf{D}$. Since all non-diagonal entries of $\mathbf{D}$ are zero, the entry in the m-th row and n-th column of $\mathbf{M}^T\mathbf{D}\mathbf{M}$, denoted $(\mathbf{M}^T\mathbf{D}\mathbf{M})_{mn}$ can be given by

$$\begin{aligned}(\mathbf{M}^T\mathbf{D}\mathbf{M})_{mn} &= (\mathbf{1}^V_*(A_m))^T \cdot \mathbf{d} \cdot \mathbf{1}^V_*(A_n) \\ &= \sum_{i=1}^{n} a_{im} a_{in} (\mathbf{D})_{ii}.\end{aligned}$$

If $m \neq n$, at least one of a_{im} and a_{in} is zero. Therefore, if $m \neq n$, we have

$$\begin{aligned}(\mathbf{M}^T\mathbf{D}\mathbf{M})_{mn} &= \sum_{i=1}^{n} a_{im} a_{in} (\mathbf{D})_{ii} \\ &= \sum_{i=1}^{n} 0 \cdot (\mathbf{D})_{ii} \\ &= 0\end{aligned}$$

If $m = n$, however, we have $a_{im} = a_{in} = 1/\sqrt{\text{assoc}(A_m)}$ if $i \in A_m$, whence

$$\begin{aligned}(\mathbf{M}^T\mathbf{DM})_{mn} &= \sum_{i=1}^{n} a_{im}^2(\mathbf{D})_{ii} \\ &= \sum_{i\in A_m}\left(\frac{1}{\sqrt{\text{assoc}(A_m)}}\right)(\mathbf{D})_{ii} \\ &= \frac{1}{\text{assoc}(A_m)}\sum_{i\in A_m}(\mathbf{D})_{ii} \\ &= \frac{1}{\text{assoc}(A_m)}\cdot\text{assoc}(A_m) \\ &= 1.\end{aligned}$$

Hence, $\mathbf{M}^T\mathbf{DM} = \mathbf{1}_{k\times k}$. □

Theorem 3.7. $\text{Ncut}(A) = \text{Tr}(\mathbf{M}^T\mathbf{LM})$.

Proof. From theorem 3.1, we know that

$$(\mathbf{1}_*^V(A_j))^T\cdot\mathbf{L}\cdot\mathbf{1}_*^V(A_j) = \frac{1}{2}\sum_{i,i'=1}^{n}(\mathbf{W})_{ii'}(a_{ij}-a_{i'j})^2.$$

Furthermore, if $i, i' \in A_j$ or $i, i' \notin A_j$, we have $a_{ij} - a_{i'j} = 0$ Thus, the only non-zero summands are those associated with pairs (i, i') where either i or i' lies in A_j. We can therefore rewrite our sum to get

$$\begin{aligned}\frac{1}{2}\sum_{i,i'=1}^{n}(\mathbf{W})_{ii'}(a_{ij}-a_{i'j})^2 &= \frac{1}{2}\Big(\sum_{i\in A_j, i'\notin A_j}(\mathbf{W})_{ii'}(a_{ij}-a_{i'j})^2 + \sum_{i\notin A_j, i'\in A_j}(\mathbf{W})_{ii'}(a_{ij}-a_{i'j})^2\Big) \\ &= \frac{1}{2}\Big(\sum_{i\in A_j, i'\notin A_j}(\mathbf{W})_{ii'}\left(\frac{1}{\sqrt{\text{assoc}(A_j)}}\right)^2 + \sum_{i\notin A_j, i'\in A_j}(\mathbf{W})_{ii'}\left(-\frac{1}{\sqrt{\text{assoc}(A_j)}}\right)^2\Big) \\ &= \sum_{i\in A_j, i'\notin A_j}(\mathbf{W})_{ii'}\left(\frac{1}{\sqrt{\text{assoc}(A_j)}}\right)^2 \\ &= \frac{\sum_{i\in A_j, i'\in V-A_j}(\mathbf{W})_{ii'}}{\text{assoc}(A_j)} \\ &= \frac{\text{cut}(A_j, V-A_j)}{\text{assoc}(A_j)}.\end{aligned}$$

The trace of $\mathbf{M}^T\mathbf{LM}$ is the sum of its diagonal elements, and since the vectors $\mathbf{1}_*^V(A_1), \ldots, \mathbf{1}_*^V(A_k)$ make up the columns of $\mathbf{M}$, it can be easily checked that the j-th diagonal element of $\mathbf{M}^T\mathbf{LM}$ can be written exactly as $(\mathbf{1}_*^V(A_j))^T\cdot\mathbf{L}\cdot\mathbf{1}_*^V(A_j)$. From the argument above, it follows that

$$\text{Tr}(\mathbf{M}^T\mathbf{LM}) = \sum_{j=1}^{k}(\mathbf{1}_*^V(A_j))^T\cdot\mathbf{L}\cdot\mathbf{1}_*^V(A_j) = \frac{\text{cut}(A_j, V-A_j)}{\text{assoc}(A_j)} = \text{Ncut}(A).$$

□

Based on theorem 3.6 and 3.7, we can reformulate minimizing Ncut towards finding indicator vectors that minimize

$$\begin{aligned}\underset{A}{\text{minimize}} \quad & f_S := \text{Tr}(\mathbf{M}^T\mathbf{LM}), \\ \text{subject to} \quad & \mathbf{M}^T\mathbf{DM} = \mathbf{1}^{k\times k} \text{ and } \mathbf{M} \text{ such that } m_{ij} \in \{0, 1\}.\end{aligned}$$

3.2.4 Relaxing our objective

The way we constructed M based on the k-cut A turns the objective into a discrete minimization problem, since all entries in M are allowed to take on a discrete number of (two) values. Since we have only reformulated our problem, minimizing this objective is still NP-hard. To allow for approximation, we can relax our objective by letting the entries of M take arbitrary real values. Our relaxed objective then becomes

$$\begin{aligned} \underset{M}{\text{minimize}} \quad & f_S' := \text{Tr}(\mathbf{M}^T\mathbf{L}\mathbf{M}), \\ \text{subject to} \quad & \mathbf{M}^T\mathbf{D}\mathbf{M} = \mathbf{1}^{k\times k} \text{ and } \mathbf{M} \in \mathbb{R}^{n\times k}. \end{aligned}$$

The optimal solution to f_S' is more tractable to compute.

Theorem 3.8. *The optimal solution to f_S' is the matrix $\mathbf{U}$ with the first k eigenvectors[9] of the generalized eigenvalue problem $\mathbf{L}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$ as its columns.*

3.3 Proof of theorem 3.8

Originally stated without proof in [68], we construct a full proof of the theorem here. It requires finding a lower bound for $\text{Tr}(\mathbf{M}^T\mathbf{L}\mathbf{M})$, and showing that $\mathbf{M} := \mathbf{U}$ achieves this bound. For a brief mathematical background, we again refer to appendix A.

Lemma 3.9 (Courant-Fischer Min-Max Theorem)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be some symmetric matrix with eigenvalues*

$$\lambda_1 \leq \cdots \leq \lambda_n.$$

Then,

$$\lambda_d = \min_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=d}} \max_{\substack{\mathbf{x}\in S \\ \mathbf{x}\neq\mathbf{0}_n}} R_{\mathbf{A}}(\mathbf{x}),$$

where $R_{\mathbf{A}}(\mathbf{x})$ denotes the Rayleigh-Ritz quotient [42], defined as

$$\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}},$$

for any $1 \leq d \leq n$.

Proof. Let $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^n$ be an orthonormal set of eigenvectors of $\mathbf{A}$ corresponding to eigenvalues $\lambda_1, \ldots, \lambda_n$, implying $\|\mathbf{u}_d\|^2 = 1$ for $1 \leq d \leq n$. Since $\mathbf{A}$ is symmetric, this set exists (see appendix A.1.2).

Now let $S_d := \text{span}(\mathbf{u}_d, \ldots, \mathbf{u}_n)$ and let $S \subseteq \mathbb{R}^n$ be an arbitrary set with $\dim(S) = d$. If $S_d \cap S = \{\mathbf{0}^n\}$, then by lemma A.4, $S_d + S$ would be a subspace of $\mathbb{R}^n$ of dimension $d + (n - d + 1) = n + 1$, which is impossible by lemma A.5. Therefore, $S_d \cap S \neq \{\mathbf{0}^n\}$. Now let $\mathbf{s} \in S_d \cap S$ such that $\mathbf{s} \neq \mathbf{0}^n$. Since $\mathbf{s} \in S_d$, we can write $\mathbf{s}$ as a linear combination $\mathbf{s} = \sum_{j=d}^n \alpha_j \mathbf{u}_j$, for scalars $\alpha_j \in \mathbb{R}$. Then, we see

$$R_{\mathbf{A}}(\mathbf{s}) = \frac{\mathbf{s}^T\mathbf{A}\mathbf{s}}{\mathbf{s}^T\mathbf{s}} = \frac{\sum_{j=d}^n \alpha_j^2\lambda_j}{\sum_{j=d}^n \alpha_j^2} \geq \frac{\sum_{j=d}^n \alpha_j^2\lambda_d}{\sum_{j=d}^n \alpha_j^2} = \lambda_d,$$

where the inequality follows from $\lambda_d \leq \cdots \leq \lambda_n$. Therefore,

$$\max_{\substack{\mathbf{x}\in S \\ x\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x}) \geq \lambda_d.$$

Since S was chosen as an arbitrary subspace of $\mathbb{R}^n$ with dimension d, this statement holds for all such sets.

In particular, it holds for $S_1 := \text{span}(\mathbf{u}_1, \ldots, \mathbf{u}_d)$. Again, using lemma A.4, we can pick $\mathbf{0}^n \neq \mathbf{s} \in S_1 \cap S_d$, and write $\mathbf{s} = \sum_{j=1}^d \beta_j \mathbf{u}_j$, for scalars $\beta_j \in \mathbb{R}$. Analogously, we get

$$R_{\mathbf{A}}(\mathbf{s}) = \frac{\mathbf{s}^T\mathbf{A}\mathbf{s}}{\mathbf{s}^T\mathbf{s}} = \frac{\sum_{j=1}^n \beta_j^2\lambda_j}{\sum_{j=1}^n \beta_j^2} \leq \frac{\sum_{j=1}^n \beta_j^2\lambda_d}{\sum_{j=1}^n \beta_j^2} = \lambda_d,$$

[9] By 'the first k eigenvectors' we mean eigenvectors corresponding to the k smallest eigenvalues.

where the inequality follows from $\lambda_1 \leq \cdots \leq \lambda_d$. Combining the two inequalities, we get

$$\min_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=d}} \max_{\substack{\mathbf{x}\in S \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x}) = \lambda_d,$$

which is what we wanted to show. □

Corollary 3.10. *We have*

$$\lambda_n = \max_{\substack{\mathbf{x}\in\mathbb{R}^n \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x})$$

Proof. Any subspace S of $\mathbb{R}^n$ of dimension n is equal to $\mathbb{R}^n$ itself. The statement now follows from lemma 3.9. □

Corollary 3.11. *Let* $\mathbf{A}$ *as in lemma 3.9. Then*

$$\lambda_d = \max_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=n-d+1}} \min_{\substack{\mathbf{x}\in S \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x})$$

Proof. Analogous as in lemma 3.9, by swapping S_d and S_1. □

Lemma 3.12 (Part of Cauchy's Interlacing Theorem)**.** *Suppose that* $1 \leq k \leq n$. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be symmetric, partitioned as*

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{D} \end{bmatrix}, \text{ with } \mathbf{B} \in \mathbb{R}^{k\times k}, \mathbf{D} \in \mathbb{R}^{n-k\times n-k}, \mathbf{C} \in \mathbb{R}^{k\times n-k},$$

and let the $\lambda_1^{\mathbf{A}} \leq \cdots \leq \lambda_n^{\mathbf{A}}$ *and* $\lambda_1^{\mathbf{B}} \leq \cdots \leq \lambda_k^{\mathbf{B}}$ *be the eigenvalues of* $\mathbf{A}$ *and* $\mathbf{B}$. *Then,* $\lambda_d^{\mathbf{A}} \leq \lambda_d^{\mathbf{B}}$ *for* $1 \leq d \leq k$.

Proof. From corollary 3.11, we have

$$\lambda_d^{\mathbf{A}} = \max_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=n-d+1}} \min_{\substack{\mathbf{x}\in S \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x}).$$

Similarly, we get

$$\lambda_d^{\mathbf{B}} = \max_{\substack{S\subseteq\mathbb{R}^k \\ \dim(S)=(n-k)-d+1}} \min_{\substack{\mathbf{y}\in S \\ \mathbf{y}\neq\mathbf{0}^n}} R_{\mathbf{B}}(\mathbf{y}).$$

Now for any $\mathbf{y} \in \mathbb{R}^k$ define $\bar{\mathbf{y}} \in \mathbb{R}^n$ which has the same elements as $\mathbf{y}$ for the first k dimensions, and zero for the last $n-k$ dimensions. Then, $\mathbf{y}^T\mathbf{B}\mathbf{y} = \bar{\mathbf{y}}^T\mathbf{A}\bar{\mathbf{y}}$ and also $\mathbf{y}^T\mathbf{y} = \bar{\mathbf{y}}^T\bar{\mathbf{y}}$, whence $R_{\mathbf{B}}(\mathbf{y}) = R_{\mathbf{A}}(\bar{\mathbf{y}})$. We can now rewrite $\lambda_d^{\mathbf{B}}$ as

$$\lambda_d^{\mathbf{B}} = \max_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=n-d+1}} \min_{\substack{\bar{\mathbf{y}}\in S,\dagger \\ \mathbf{y}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\bar{\mathbf{y}}),$$

where † refers to the condition on $\bar{\mathbf{y}}$ that its last $n-k$ elements have to equal zero. Clearly, this is the same statement as $\lambda_d^{\mathbf{A}}$, but with the extra condition †. If we substitute $\mathbf{x} := \bar{\mathbf{y}}$, we thus get

$$\lambda_d^{\mathbf{A}} = \max_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=n-d+1}} \min_{\substack{\mathbf{x}\in S \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x}) \leq \max_{\substack{S\subseteq\mathbb{R}^n \\ \dim(S)=n-d+1}} \min_{\substack{\mathbf{x}\in S,\dagger \\ \mathbf{x}\neq\mathbf{0}^n}} R_{\mathbf{A}}(\mathbf{x}) = \lambda_d^{\mathbf{B}}.$$

□

Corollary 3.13 (part of the Poincaré Separation Theorem)**.** *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be symmetric and suppose that* $1 \leq k \leq n$. *Let* $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^n$ *be an orthonormal set of vectors and let* $\mathbf{V} \in \mathbb{R}^{n\times k}$ *be the (orthogonal) matrix with* $\mathbf{u}_1, \ldots, \mathbf{u}_k$ *as its columns. Set* $\mathbf{B} := \mathbf{V}^T\mathbf{A}\mathbf{V} \in \mathbb{R}^{k\times k}$ *and arrange the eigenvalues of* $\mathbf{A}$ *and* $\mathbf{B}$ *as above. Then,* $\lambda_d^{\mathbf{A}} \leq \lambda_d^{\mathbf{B}}$ *for* $1 \leq d \leq k$.

Proof. If $k < n$, pick $n-k$ additional orthonormal vectors $\mathbf{u}_{k+1}, \ldots, \mathbf{u}_n \in \mathbb{R}^n$, and let $\mathbf{U} \in \mathbb{R}^{n\times n}$ be an extension of $\mathbf{V}$ by adding these last orthonormal vectors as columns. Then, $\mathbf{U}$ is an orthogonal matrix, i.e. $\mathbf{U}^T\mathbf{U} = 1_n$. By lemma A.10, we see that $\mathbf{A}$ and $\mathbf{U}^T\mathbf{A}\mathbf{U}$ share the same eigenvalues.

To finish our argument, see that removing the last $n-k$ rows and columns of $\mathbf{U}^T\mathbf{A}\mathbf{U}$ gives us back $\mathbf{B}$. Hence, lemma 3.12 gives us

$$\lambda_d^{\mathbf{A}} = \lambda_d^{\mathbf{U}^T\mathbf{A}\mathbf{U}} \leq \lambda_d^{\mathbf{B}},$$

which is what we wanted to show. □

Corollary 3.14. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *as above. Then,* $\mathrm{Tr}(\mathbf{B}) \geq \sum_{d=1}^{k} \lambda_d^{\mathbf{A}}$.

Proof. Combining corollary 3.13 and the definition of Tr yields

$$\mathrm{Tr}(\mathbf{B}) = \sum_{d=1}^{k} \lambda_d^{\mathbf{B}} \geq \sum_{d=1}^{k} \lambda_d^{\mathbf{A}}.$$

□

For our proof of theorem 3.8, we will rewrite our original objective

$$\begin{aligned} \underset{\mathbf{M}}{\text{minimize}} \quad & \mathrm{Tr}(\mathbf{M}^T \mathbf{L} \mathbf{M}), \\ \text{subject to} \quad & \mathbf{M}^T \mathbf{D} \mathbf{M} = \mathbf{1}^{k \times k} \text{ and } \mathbf{M} \in \mathbb{R}^{n \times k}. \end{aligned}$$

to suit the form of corollary 3.13. To do so, let $\mathbf{L_N} := \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{n \times n}$ and $\mathbf{H} := \mathbf{D}^{\frac{1}{2}} \mathbf{M}$. The objective then becomes

$$\begin{aligned} \underset{\mathbf{H}}{\text{minimize}} \quad & \mathrm{Tr}(\mathbf{H}^T \mathbf{L_N} \mathbf{H}), \\ \text{subject to} \quad & \mathbf{H}^T \mathbf{H} = \mathbf{1}^{k \times k} \text{ and } \mathbf{H} \in \mathbb{R}^{n \times k}. \end{aligned}$$

To enhance the structure of the proof, we will show one property of $\mathbf{L_N}$ that we will use.

Lemma 3.15. *If* $\mathbf{v}$ *is an eigenvector of* $\mathbf{L_N}$ *with eigenvalue* λ*, then* $\mathbf{u} := \mathbf{D}^{-\frac{1}{2}} \mathbf{v}$ *is a solution to the generalized eigenvalue problem* $\mathbf{L}\mathbf{u} = \lambda \mathbf{D}\mathbf{u}$.

Proof.

$$\begin{aligned} \mathbf{L_N}\mathbf{v} = \lambda \mathbf{v} &\iff \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \mathbf{v} = \lambda \mathbf{v} \\ &\iff \mathbf{L}\mathbf{D}^{-\frac{1}{2}} \mathbf{v} = \lambda \mathbf{D}^{\frac{1}{2}} \mathbf{v} && \text{(multiplying by } \mathbf{D}^{\frac{1}{2}} \text{ from the left)} \\ &\iff \mathbf{L}(\mathbf{D}^{-\frac{1}{2}} \mathbf{v}) = \lambda \mathbf{D}(\mathbf{D}^{-\frac{1}{2}} \mathbf{v}) && \text{(by rearranging)} \\ &\iff \mathbf{L}\mathbf{u} = \lambda \mathbf{D}\mathbf{u}. \end{aligned}$$

□

Proof of theorem 3.8. By definition, $\mathbf{L_N} \in \mathbb{R}^{n \times n}$ is a symmetric matrix. Hence, corollary 3.14 applies, so

$$\mathrm{Tr}(\mathbf{H}^T \mathbf{L_N} \mathbf{H}) \geq \sum_{d=1}^{k} \lambda_d^{\mathbf{L_N}},$$

for all $\mathbf{H} \in \mathbb{R}^{n \times k}$ with $\mathbf{H}^T \mathbf{H} = \mathbf{1}^{k \times k}$.

Now set $\mathbf{H}$ to be the matrix having the first k eigenvectors of $\mathbf{L_N}$, $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$, as its columns. It then follows that

$$\begin{aligned} \mathrm{Tr}(\mathbf{H}^T \mathbf{L_N} \mathbf{H}) &= \sum_{d=1}^{k} \mathbf{v}_d^T \mathbf{L_N} \mathbf{v}_d \\ &= \sum_{d=1}^{k} \mathbf{v}_d^T \lambda_d^{\mathbf{L_N}} \mathbf{v}_d \\ &= \sum_{d=1}^{k} (\mathbf{v}_d^T \mathbf{v}_d) \lambda_d^{\mathbf{L_N}} \\ &= \sum_{d=1}^{k} \lambda_d^{\mathbf{L_N}} && \text{(since } \mathbf{v}_d^T \mathbf{v}_d = 1) \end{aligned}$$

Combining everything, we get

$$\min_{\mathbf{H} \in \mathbb{R}^{n \times k}} \mathrm{Tr}(\mathbf{H}^T \mathbf{L_N} \mathbf{H}) = \sum_{d=1}^{k} \lambda_d^{\mathbf{L_N}},$$

and this minimum is reached by the matrix $\mathbf{H}$ having the first k eigenvectors of $\mathbf{L_N}$ as its columns. If we substitute back $\mathbf{M} = \mathbf{D}^{-\frac{1}{2}}\mathbf{H}$, we see by lemma 3.15 that the minimum for the original objective is reached by the matrix $\mathbf{M}$ having the first k eigenvectors of the generalized eigenvalue problem $\mathbf{Lu} = \lambda\mathbf{Du}$, which is what we wanted to show. □

3.4 The clustering algorithm

3.4.1 From continuous to discrete

In f'_S, the only constraints on solutions $\mathbf{M}$ are that $\mathbf{M}^T\mathbf{DM} = \mathbf{1}^{k\times k}$ and $\mathbf{M} \in \mathbb{R}^{n\times k}$. In particular, the eigenvectors of the generalized eigenvalue problem $\mathbf{u}_1, \ldots, \mathbf{u}_k$ (which are the columns of the optimal solution $\mathbf{U}$ according to theorem 3.8) are contained in $\mathbb{R}^k$, and are in generally not trivial to interpret as indicator vectors for partitions of A. We thus need some way to transform our continuous solution $\mathbf{U}$ back into a discrete k-cut A.

For $k = 2$, hard boundaries can be used. For example, we could define indicator vector $\mathbf{1}^V_*(A)$ as

$$\mathbf{1}^V_*(A) = \begin{cases} 1 \text{ if } m_{i1} - p > 0 \\ 0 \text{ if } m_{i1} - p \leq 0 \end{cases}$$

for some $p \in \mathbb{R}$. It is shown in [81] using the Cheeger inequality that the best solution to f_S can be found by iterating over different values for p.

For $k > 2$, let $\mathbf{r}_i \in \mathbb{R}^k$ correspond the i-th row of $\mathbf{U}$, $1 \leq i \leq n$. In the discrete version of $\mathbf{U}$, $\mathbf{r}_i$ would equal 1 in the column j if $i \in A_j$. In the general continuous $\mathbf{U}$, these r_i can be seen as k-dimensional representations – 'embeddings' – of the vertices i. It is shown in [13] that spectral embedding of vertices onto the space $\mathbb{R}^k$ maintains its distance and similarity relations, and enhances its cluster properties.

3.4.2 Retrieving a clustering

As a final step, the vectors $\mathbf{r}_1, \ldots, \mathbf{r}_n$ corresponding to points $\mathbf{s}_1, \ldots, \mathbf{s}_n$ are clustered using k-means (e.g. algorithm 2). This may seem arbitrary at first glance, but k-means seems to have few problems detecting desired clusters from these r_i. More importantly, a mountain of empircal evidence advocates the use of this algorithm [22] . A recent mathematical argument in favour of k-means is shown in [63], which uses an assumption on the eigenvalue gaps $\lambda_{k+1} - \lambda_k$ to show that the clustering yielded by the spectral embedding combined with k-means can give a good approximation of the optimal solution to f_S.

3.4.3 The algorithm

After discussing all steps necessary in spectral clustering, we yield the following spectral clustering algorithm.

Algorithm 3 (Normalized) spectral clustering (adapted from [73])

1: **input:** $S \subset \mathbb{R}^n$, k.
2: construct a similarity graph G as in section 3.1.
3: compute adjacency matrix W, and the corresponding graph Laplacian L of G.
4: compute the first k eigenvectors $u_1, \ldots, u_k$ of the generalized eigenvalue problem $Lu = \lambda Du$.
5: construct the matrix U containing these eigenvectors as columns.
6: let r_i correspond to the i-th row of U, $1 \leq i \leq n$.
7: apply algorithm 2 with parameters $R := \{r_1, \ldots, r_n\}$ and amount of clusters k, to get clustering $R_1, \ldots, R_k$.
8: **return** a clustering $C_1, \ldots, C_k$ with $C_i = \{s_i \mid r_i \in R_i.\}$

Step 2 and 3 correspond to the construction of graph G, step 4,5 and 6 to the minimization of f'_S following theorem 3.8, and clustering takes place in step 7.

4 (H)DBSCAN

4.1 Challenging assumptions

4.1.1 The k-means assumptions

The assumption behind the k-means objective is that a good clustering minimizes the sum of pairwise squared euclidean distances of points within a cluster. Implicitly, this assumption alone entails that k-means assumes that clusters are of globular shape, well-separated and of about the same size [79]. If the clusters in the data are indeed of said arrangement, k-means has few problems detecting sensible clusterings.

Since k-means is not the only domain where squared Euclidean distance is used as a metric, let us consider an analogy. In regression problems, for example, minimizing the sum of squared Euclidean distances gives a reliable, unbiased predictor, but only if some very specific assumptions are fulfilled. If these so-called Gauss-Markov assumptions are not met, regression may still work but there are few guarantees on its functionality. A humorous example of such situation was given by Anscombe [4], in which 4 data sets were constructed to have equivalent descriptive statistics - mean, (sample) variance, correlation - but only one of them fulfilled the Gauss-Markov assumptions. Based on their statistics, the resulting regression lines were exactly the same, but for 3 out of 4 sets made only little sense when plotted.

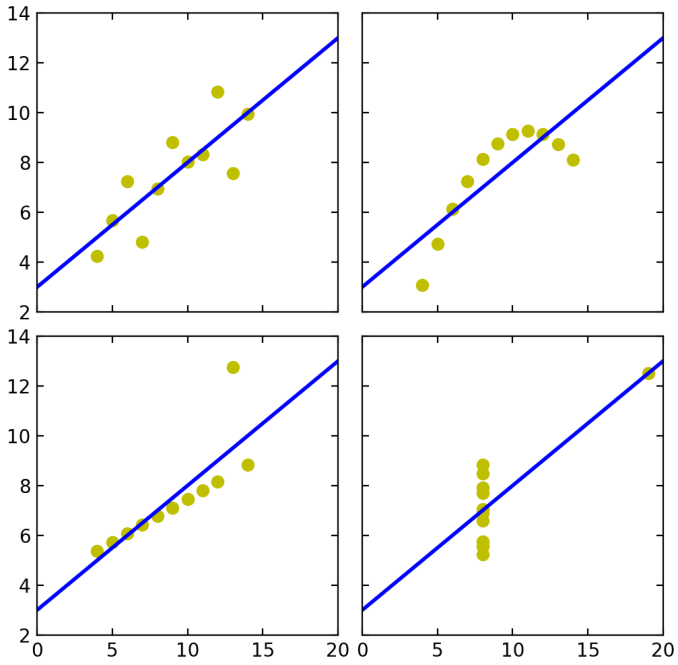


Figure 4.1: The four datasets constructed by Anscombe (see [4] for the details), with their corresponding regression lines drawn in red. All four datasets have similar descriptive statistics, while only one suits the assumptions necessary for squared Euclidean distance to give a reliable, unbiased predictor (seen in the top left).

Equivalently, datasets containing clusters that do not fulfill the k-means assumptions, may be clustered by k-means, but the results may be far from satisfactory.

Another assumption – or rather, implementation choice – of k-means and spectral clustering we have fended off up until now, concernsoutlier points. From the start, we have assumed that all points in our input set S should belong to any one of the k clusters. In practice however, this might be undesirable: data may

be contaminated with (measurement) noise: points which can not easily be grouped together or obviously belong to some structural trend that may constitute a cluster. This is a serious issue to consider, and it might be compelling to exclude some data points as noise (based on some predefined threshold) if the clusters that do persist become stronger from it.

In an attempt to address these issues, the density-based clustering approach was created.

4.1.2 A density-based approach

In density-based clustering, clusters are assumed to be highly dense regions of points, separated by less dense (sparse) regions of points. Since no assumptions are made about cluster shapes and sizes, a well-implemented density-based clustering algorithm may detect clusters no matter their said forms [31]. Moreover, there is a natural extension to detect outliers, by looking in the sparse regions surrounding the denser clusters. We will now shortly formalize this intuition.

For starters, it is assumed that S is not merely a set, but a sample - hence its name - drawn from a probability distribution of a continuous random variable X that takes on values in $\mathbb{R}^d$, with probability density function p_X. We expect areas of high probability according to p_X to contain a lot of points in our sample S, and thus to be 'denser' than areas of low probability.

Given p_X and some $\lambda \in [0,1]$, we define the λ-level set of p_X as the set

$$\{\mathbf{s} \in \mathbb{R}^d \mid p_X(\mathbf{s}) \geq \lambda\}.$$

The level set contains all values X may take on with probability at least λ. We expect areas of high probability to be related with highly dense regions in our sample, and as such, we call λ a density threshold, and all $\mathbf{s} \in S$ that are contained in the λ-level set are considered dense points.

Next, we define a homogeneous 'connectivity' relation σ on S, and say $\mathbf{s}, \mathbf{s}'$ are connected if $(\mathbf{s}, \mathbf{s}') \in \sigma$ (see appendix A.2). The only constraint on σ is that it is an equivalence relation, such that it partitions S into its connected components (see [17] and lemma A.11).

Now, we define the cluster tree of p_X as a function $C_{p_X} : [0,1] \to \Pi(S)$, such that

$$C_{p_X}(\lambda) := \{\text{connected components of } \{\mathbf{s} \in \mathbb{R}^d \mid p_X(\mathbf{s}) \geq \lambda\}\}.$$

Given λ, all elements of $C_{p_X}(\lambda)$ are clusters of S.[10]

Varying λ results in various cluster trees: higher λ means a higher density threshold, which in turn means less elements contained in the λ-level set and thus, smaller clusters. Analogously, a lower λ means larger level sets and thus larger clusters. As such, the different cluster trees form a hierarchy, which we will formalize in the next theorem.

Lemma 4.1 (see [17]). *Pick any $\lambda' \leq \lambda$, and let σ be some equivalence relation. Then:*

1. *For any $C \in C_{p_X}(\lambda)$, there exists a $C' \in C_{p_X}(\lambda')$ such that $C \subseteq C'$.*
2. *For any $C \in C_{p_X}(\lambda)$ and $C' \in C_{p_X}(\lambda')$, either $C \subseteq C'$ or $C \cap C' = \emptyset$.*

Proof.

1. Let $\mathbf{s} \in C$. Since σ is reflexive, we know $(\mathbf{s}, \mathbf{s}) \in \sigma$, and we know that $\mathbf{s} \in \{\mathbf{s} \in \mathbb{R}^d \mid p_X(\mathbf{s}) \geq \lambda\}$ since $C \in C_{p_X}(\lambda)$. Furthermore, since $\lambda' \leq \lambda$, it holds that $\{\mathbf{s} \in \mathbb{R}^d \mid p_X(\mathbf{s}) \geq \lambda\} \subseteq \{\mathbf{s} \in \mathbb{R}^d \mid p_X(\mathbf{s}) \geq \lambda'\}$. Thus, $\mathbf{s}$ is connected according to σ and in the λ'-level set of p_X, whence it is contained in some cluster $C' \in C_{p_X}(\lambda')$. Moreover, since $\mathbf{s}$ is connected to all other elements in C – by definition –, we know that all other $\mathbf{s}' \in C$ are also in C'. Since $\mathbf{s}$ was chosen arbitrarily, we know $C \subseteq C'$.
2. Let $C \in C_{p_X}(\lambda)$ and $C' \in C_{p_X}(\lambda')$, and suppose $C \not\subseteq C'$. Then there exists some $\mathbf{s} \in C$ such that $\mathbf{s} \notin C'$. By the same argument as above, we know that $\mathbf{s}$ is connected via σ to all other elements in C, hence they all belong to the same connected component as $\mathbf{s}$, which is not C'. Therefore, no element of C is contained in C', whence $C \cap C' = \emptyset$.

□

[10] Actually, clusters adhering this definition called 'density-contour' clusters in literature [39], but since they will be the only type of density-based clusters, we will see, we skip that fuss.

See [39, 54] for simplified illustrations of cluster trees based on a probability density function.

4.1.3 An estimated density-based approach

Now that we have outlined the essence of a density-based clustering approach, we may be tempted to start writing a fairly simple algorithm that takes a λ-level set, determines connections according to some equivalence relation σ and returns the resulting connected components as clusters.

However, this all lies on the assumption that we have access to p_X, so that we can reliably estimate the level sets and determine which points of S belong to them. In practice, however, we only have access to input data S, with no further information or guarantees about its underlying probability distribution. Thus, we need a way of estimating p_X, or in other words, estimate the density around all points in S.

A density estimator proposed in the early days of density-based clustering [31] is based around the m-nearest neighbor of all points in S.[11] Intuitively, points that lie in dense regions have many points around them, hence their m-nn is closer than those of points in sparser regions. To formalize this intuition, we follow [15] and define the core distance $d_{\text{core},m}$ of some $\mathbf{s} \in S$ wrt. m as the distance from $\mathbf{s}$ to its m-nn (thereby including $\mathbf{s}$ itself), i.e.

$$d_{\text{core},m}(\mathbf{s}) := \min\{e \mid |N(\mathbf{s}, e)| \geq m\},$$

where $N(\mathbf{s}, e)$ is again defined as the set of points within e distance of $\mathbf{s}$ (the e-neighborhood of $\mathbf{s}$). If we pick some $\varepsilon \geq 1$, we say $\mathbf{s}$ is an ε-core point for every ε greater than or equal than the core distance of $\mathbf{s}$, i.e. $d_{\text{core},m}(\mathbf{s}) \leq \varepsilon$. Here, ε functions as a sparsity threshold, since higher ε means more points are considered a core point. The corresponding density threshold can be found by setting $\lambda := 1/\varepsilon$. For measuring distance, any δ as described in section 3.1 holds, although Euclidean distance was used in the original experiments.

To connect points based on said sparsity estimates, we define $\mathbf{s}, \mathbf{s}' \in S$ to be reachable (wrt. ε) if and only if both $\mathbf{s}$ and $\mathbf{s}'$ are ε-core points, and $\delta(\mathbf{s}, \mathbf{s}') \leq \varepsilon$. We define the smallest ε for which this happens as the mutual reachability distance, i.e.

$$d_{\text{mreach}}(\mathbf{s}, \mathbf{s}') := \begin{cases} \max(d_{\text{core},m}(\mathbf{s}), d_{\text{core},m}(\mathbf{s}'), \delta(\mathbf{s}, \mathbf{s}')) & \text{if } \mathbf{s} \neq \mathbf{s}' \\ 0 & \text{if } \mathbf{s} = \mathbf{s}' \end{cases}.$$

Moreover, we define an equivalence[12] relation σ such that $(\mathbf{s}, \mathbf{s}') \in \sigma$ if and only if $\mathbf{s}$ and $\mathbf{s}'$ are directly or transitively reachable wrt. ε. Lastly, we construct a – complete – mutual reachability graph $G_{reach,m} := (V, E, \mathbf{W})$ (see appendix A.3), such that the vertices represent all points in S and the $w_{ij} := d_{\text{mreach}}(\mathbf{s}, \mathbf{s}')$. The proofs of the following statements follow directly from its construction:

Corollary 4.2. *For some $\varepsilon \geq 1$, let $G_{reach,m,\varepsilon} \subseteq G_{reach,m}$ be the subgraph obtained by removing all edges with weight higher than ε from $G_{reach,m}$. The connected components of $G_{reach,m,\varepsilon}$ are the connected points of S wrt. ε and σ.*

Corollary 4.3. *For some $\varepsilon \geq 1$, let O be the set of non-ε-core points. Then, O consists of all singleton connected components of $G_{reach,m,\varepsilon}$.*

Since O contains all points that are unconnected to all other points in S, we call these the outlier points.

4.2 Algorithms

4.2.1 DBSCAN

To wrap up the last section, we can revise our previous definition of density-based clusters. With knowledge of p_X, we defined clusters to be the connected components of a λ-level set of p_X. Using density estimates, we define clusters to be the connected components of $G_{reach,m,\varepsilon}$ wrt. ε and σ. If we translate this into an algorithm, we yield DBSCAN.

[11] Again, we use m instead of k to prevent confusion with the number of clusters k.
[12] It can be easily checked that it is an equivalence relation.

Algorithm 4 DBSCAN

1: **input:** $S \subset \mathbb{R}^n$, m, ε
2: compute $d_{\text{core},m}(s)$ for all $s \in S$.
3: construct $G_{reach,m,\varepsilon}$.
4: compute the non-singleton connected components $C_1, \ldots, C_n$ of $G_{reach,m,\varepsilon}$, and let O be the set of singleton components.
5: **return** a clustering $C_1, \ldots, C_n$ and noise points O.

Note that this is the first clustering algorithm[13] we have seen that does not require a parameter k for the number of clusters beforehand, as this number depends entirely on the input data and the choices for m and ε.

4.2.2 Varying ε

Varying ε as the input to algorithm 4 changes the resulting clusterings: a lower ε means clusters get smaller, until disappearing (and becoming noise) or breaking into smaller clusters if the last edge connecting them is removed. In particular, there exist $\varepsilon_{\max}$ and $\varepsilon_{\min}$, such that $G_{reach,m,\varepsilon}$ is fully connected if $\varepsilon \geq \varepsilon_{\max}$ and fully disconnected if $\varepsilon \leq \varepsilon_{\min}$. The smallest such $\varepsilon_{\max}$ equals the largest edge weight in $G_{reach,m}$, while the largest such $\varepsilon_{\min}$ equals the smallest weight. Using $\varepsilon = \varepsilon_{\max}$ as input, DBSCAN returns one cluster containing all elements. Likewise, setting $\varepsilon := \varepsilon_{\min}$ only returns singleton components. In this sense, the interval between $\varepsilon_{\min}$ and $\varepsilon_{\max}$ must contain all possible clustering outputs produced by DBSCAN.

Furthermore, for all vertices in the input[14], there exists a 'tipping point' for ε^* such that for all $\varepsilon \geq \varepsilon^*$ it belongs to some cluster, and $\varepsilon < \varepsilon^*$ it is considered noise. It can be easily seen that this tipping point lies exactly at the lowest mutual reachability distance with any other vertex in the input, since this is the last edge connecting it to its cluster. Therefore, we can again implement a hierarchy of connected components, by varying ε from $\varepsilon_{\max}$ down to $\varepsilon_{\min}$. This hierarchy resembles the result from lemma 4.1.

Lemma 4.4. *Let us define a cluster tree* $C_G : [1, \infty) \to \Pi(S)$ *such that*

$$C_G(\varepsilon) := \{\text{connected components of } G_{reach,m,\varepsilon}\}.$$

Pick any $\varepsilon \leq \varepsilon'$. *Then:*

1. *For any* $C \in C_G(\varepsilon)$, *there exists a* $C' \in C_G(\varepsilon')$ *such that* $C \subseteq C'$.
2. *For any* $C \in C_G(\varepsilon)$ *and* $C' \in C_G(\varepsilon')$, *either* $C \subseteq C'$ *or* $C \cap C' = \emptyset$.

Proof. 1. Let w be the highest edge weight of an edge connecting two vertices in C. Since $C \in C_G(\varepsilon)$, we know $w \leq \varepsilon$, and thus $w \leq \varepsilon'$. Therefore, the edge with weight w will not be removed when constructing $G_{reach,m,\varepsilon'}$. Since w was the largest edge weight, none of the edges in C will be removed, hence all vertices are still contained in some connected component of $G_{reach,m,\varepsilon'}$. If we call this component C', it holds that $C \subseteq C'$.

2. In 1. we established that all vertices in some $C \in C_G(\varepsilon)$ are still all connected to each other in some cluster in $C_G(\varepsilon')$. The other clusters in $C_G(\varepsilon')$ therefore contain none of the vertices of C. Hence, $C \subseteq C'$ or $C \cap C' = \emptyset$.

□

Although C_G is defined here as a function from values of ε to a partition of S, we can draw an analogy to the concept of a tree known from graph theory. In particular, the hierarchy level $\varepsilon_{\max}$ corresponding to the root of the tree contains only node (corresponding to a cluster), while the singleton components at level $\varepsilon_{\min}$ correspond to its leaves. Descendants and ascendants are structured according to lemma 4.4, i.e. a node at hierarchy ε that is a child of a node at hierarchy ε', $\varepsilon \leq \varepsilon'$, corresponds to a cluster $C \in C_G(\varepsilon)$ such that it is contained in the cluster at the parent's node, $C' \in C_G(\varepsilon')$.

[13] This version of DBSCAN is known in literature as DBSCAN* [15].
[14] Assuming this input contains more than one point.

4.2.3 HDBSCAN

Constructing a cluster tree as per lemma 4.4 is precisely the goal of the Hierarchical DBSCAN algorithm, called HDBSCAN. Starting from $\varepsilon_{\max}$, we keep lowering ε until we end up at $\varepsilon_{\min}$.

Algorithm 5 HDBSCAN using connected components

1: **input:** $S \subset \mathbb{R}^n$, m
2: compute $d_{\text{core},m}(s)$ for all $s \in S$.
3: compute the mutual reachability graph $G_{reach,m}$.
4: compute $\varepsilon_{\max}$, $\varepsilon_{\min}$, and set $\varepsilon := \varepsilon_{\max}$.
5: **while** $\varepsilon \geq \varepsilon_{\min}$:
6: compute $C_G(\varepsilon)$.
7: let x be the highest edge weight in $G_{reach,m}$, and remove this edge.
8: set $\varepsilon := x$.
9: **return** C_G.

At any iteration, the edge with the highest weight gets removed from $G_{reach,m}$, until no edges are left. The returned cluster tree C_G then contains all results computed in step 6. This way of iterating, while correct, is computationally expensive, since it requires to remove all edges one at a time, resulting in $|S| \cdot (|S| - 1)/2$ edge removals since $G_{reach,m}$ is fully connected.

To make a simplification, note that the only edge removals that change a clustering those after which some vertex is no longer connected to any other vertex. Specifically, this happens at the tipping point ε^* discussed before. In other words, we only need to keep track of the edge with the lowest weight that is not a self-edge. After removing all other edges, the resulting graph is a minimum spanning tree of $G_{reach,m}$ (see appendix A.3). Building a MST of a graph can be done easily and efficiently, e.g. using Borůvka's [12] or Prim's [65] algorithm. This yields the following improved version of HDBSCAN:

Algorithm 6 HDBSCAN

1: **input:** $S \subset \mathbb{R}^n$, m
2: compute $d_{\text{core},m}(s)$ for all $s \in S$.
3: compute the mutual reachability graph $G_{reach,m}$.
4: compute the MST of $G_{reach,m}$.
5: **while** MST is not empty:
6: let y be the highest weight in the MST, and remove this edge (in case of ties, remove both edges).
7: compute $C_G(y) :=$ {connected components of the MST}.
8: **return** C_G.

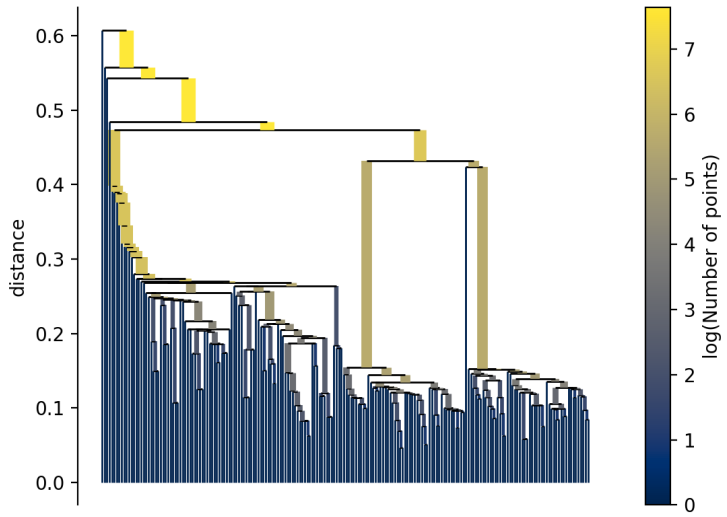


Figure 4.2: Cluster tree constructed by HDBSCAN of the toy data set introduced in section 1. The root of the tree contains all points, while the leaves are all singleton components. All clusters are contained in the cluster of their parent nodes.

As a MST contains $|S|-1$ edges, algorithm 6 will remove exactly as many (cf. $|S|\cdot(|S|-1)/2$ edge removals in algorithm 5).

4.2.4 Minimum cluster size

As seen in figure 4.2, HDBSCAN yields a vast cluster tree, in which all possible DBSCAN clusterings are contained. However, one property of DBSCAN that is now undealt with, is its capability of handling noise. In particular, all singleton components are included as leaves in the cluster tree, and they form singleton clusters after reaching their respective tipping points. Moreover, the resulting tree is highly complex, given it includes $|S|$ leaf nodes. To simplify the cluster tree, we adopt a tree pruning approach proposed in [77]. It starts off by introducing of a minimum cluster size $c_{\text{min}} \geq 1$, and forcing HDBSCAN to only include clusters larger than c_{min} in the cluster tree. The other points get added to the set of noise O.

Algorithm 7 HDBSCAN, lines 5-8, with (optional) parameter $c_{\text{min}} \geq 1$

5: **while** MST is not empty:
6: let y be the highest weight of all edges in the MST, and remove this edge (in case of ties, remove both edges).
7: create a new noise set O.
8: compute $C_G(y) :=$ {connected components of the MST}.
9: let $O_y \subseteq C_G(y)$ be the set of components C s.t. $|C| \leq c_{\text{min}}$.
10: set $C_G(y) := C_G(y) - O_y$.
11: add the edges of all components in O_y to O, and remove them from the MST.
12: **return** C_G.

Even though this addition adds a new parameter to the algorithm, tuning the minimum cluster size turns out to be quite intuitive in practice [54]. Moreover, the resulting cluster tree has significantly less branches and leaves, while still containing valuable information about the cluster structures of the input [77].

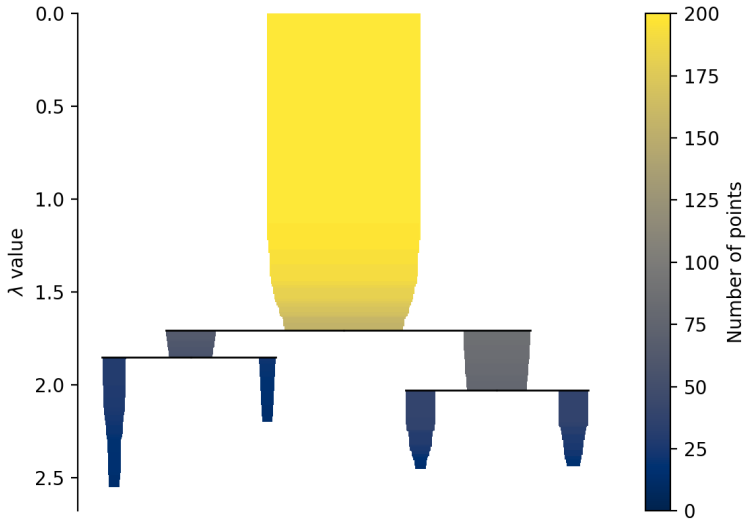


Figure 4.3: Simplified cluster tree constructed by HDBSCAN of the toy data set in section 1, using optional parameter $c_{\min} = 10$. On the y-axis, we see the hierarchy levels in the form of λ ($= 1/\varepsilon$) values.

4.3 Optimizing HDBSCAN

After computing a cluster tree such as figure 4.3, we are still left with clusterings at different density levels and sizes. In most cases however, we are mainly interested in extracting a 'flat', non-overlapping partition from our dataset, adhering to the definition given in section 2.1. In other words, we are presented with the challenge of picking clusters freely from the cluster tree, bearing in mind that no point is picked twice.

4.3.1 Excess of Mass

To recap, HDBSCAN produces all possible solutions produced by DBSCAN wrt. a value for m and all thresholds $\varepsilon \geq 1$. The local density around a point is estimated by its core distance $d_{\text{core},m}$. By definition, the core distance gets lower as points get closer together, whence assigning 'denser' points a lower value for $d_{\text{core},m}$. For this discussion we let $\lambda := 1/\varepsilon$ denote our density threshold and describe our density for a point s as $1/d_{\text{core},m}(s)$. Analogous to our earlier discussion using ε, increasing λ means clusters get smaller until they disappear or split into (multiple) subclusters. A first approach might be to pick the clusters of one specific hierarchy level and return this clustering as a solution. However, this would totally defeat the purpose of including clusters of different density levels. Therefore, we adopt a different approach.

For any cluster C, let $\lambda_{\min}(C)$ denote the smallest λ for which C appears in the cluster tree. The intuition behind the excess of mass metric is that clusters that persist over a large part of the cluster tree should be favored over short-lived clusters. For example, in figure 4.3, the large yellow-colored cluster (the root) and the blue cluster in the bottom left are more persistent than the 'trapezoidal' blue/yellow cluster in between them, since they exist over a larger interval of λ/hierarchy levels.

Formally, we define $\Gamma := \{\gamma_1, \ldots, \gamma_\kappa\}$ to be the set of all clusters on all hierarchy levels in C_G, i.e. $\Gamma = \bigcup_{y \in Y} C_G(y)$, where Y is the set of weights in the MST constructed in algorithm 7. Then, the excess of mass of a cluster γ_i wrt. p_X is defined as

$$E(\gamma_i) := \int_{\mathbf{s} \in \gamma_i} \Big(p_X(\mathbf{s}) - \lambda_{\min}(\gamma_i) \Big) \, d\mathbf{s}.$$

Theorem 4.5. *Along any branch of the cluster tree, the excess of mass does not decrease as λ increases (i.e. going in downward direction if the tree is depicted as in figure 4.3).*

Proof. Let $\lambda \leq \lambda'$, and let γ and γ' be clusters on the same branch of the cluster tree, at density levels λ and λ' respectively. We then have

$$\begin{aligned} E(\gamma) &= \int_{\mathbf{s}\in\gamma} \Big(p_X(\mathbf{s}) - \lambda_{\min}(\gamma)\Big)\, d\mathbf{s} \\ &\leq \int_{\mathbf{s}\in\gamma} \Big(p_X(\mathbf{s}) - \lambda_{\min}(\gamma')\Big)\, d\mathbf{s} \\ &\leq \int_{\mathbf{s}\in\gamma'} \Big(p_X(\mathbf{s}) - \lambda_{\min}(\gamma')\Big)\, d\mathbf{s} \\ &= E(\gamma'), \end{aligned}$$

where the first inequality follows since $\lambda_{\min}(\gamma) \leq \lambda_{\min}(\gamma')$ and the second since $\gamma' \subseteq \gamma$ (lemma 4.4). □

4.3.2 Formulating an optimization objective

Because of the monotonic behaviour of E shown by theorem 4.5, it does not make much sense to compare clusters within the same branch based on its excess of mass, since this would favor either highly dense or sparse clusters – depending on whether we choose to minimize or maximize E. In particular, by theorem 4.5, the excess of mass of clusters closer to the root encompasses those of its descendants. Furthermore, the current definition of E requires access to $p_X(\mathbf{s})$ instead of our estimate $1/d_{\text{core},m}(\mathbf{s})$.

To compare clusters (on the same branch), let $\lambda_{\max}(\gamma)$ denote the value for λ at which γ disappears or is split up into subclusters, and let $\lambda_{\max}(\mathbf{s},\gamma) := \min\big(d_{\text{core},m}(\mathbf{s}), \lambda_{\max}(\gamma)\big)$. The relative excess of mass of a cluster γ is then defined as

$$E_R(\gamma) := \int_{\mathbf{s}\in\gamma} \Big(\lambda_{\max}(\mathbf{s},\gamma) - \lambda_{\min}(\gamma)\Big)\, d\mathbf{s}.$$

If cluster γ_1 gets split up into clusters $\gamma_2, \ldots, \gamma_p$ at density level λ_y, then $\lambda_y = \lambda_{\max}(\gamma_1) = \lambda_{\min}(\gamma_2) = \cdots = \lambda_{\min}(\gamma_p)$.

In reality, we have a finite sample S, such that we can discretize E_R to get

$$P(\gamma) := \sum_{\mathbf{s}\in\gamma} \Big(\lambda_{\max}(\mathbf{s},\gamma) - \lambda_{\min}(\gamma)\Big).$$

Here, $P(\gamma)$ is called the persistence of γ, $\lambda_{\min}(\gamma)$ is the lowest level of density at which γ exists and $\lambda_{\max}(\mathbf{s},\gamma)$ is the density level at which $\mathbf{s}$ leaves γ – either because $\mathbf{s}$ is considered noise or because γ is split up [15].

Our to goal is to extract a non-overlapping clustering of S, containing the most persistent clusters in the cluster tree, including some possible noise. This can be formulated as the optimization objective

$$\begin{aligned} &\underset{\delta_2,\ldots,\delta_\kappa}{\text{maximize}} \quad f_D := \sum_{i=2}^{\kappa} \delta_i P(\gamma_i), \\ &\text{subject to} \quad \begin{cases} \delta_i \in \{0,1\} \text{ for } i \in \{2,\ldots,\kappa\} \\ \sum_{z\in W_b} \delta_z = 1 \text{ for all } b \in B, \end{cases} \end{aligned}$$

where $\delta_i = 1$ if $\mathbf{C}_i$ is included in the partition and $\delta_i = 0$ if not,

$$B := \{b \mid \mathbf{C}_b \text{ is a leaf cluster}\}$$

is the set of indices of all leaf clusters, and

$$W_b := \{z \mid z \neq 1 \text{ and } \mathbf{C}_z \text{ is an ascendant of } \mathbf{C}_b \text{ (including } b)\}$$

contains the indices of all clusters on the path from leaf node b to the root (excluding the root).[15] The constraint '$\sum_{z\in W_b} \delta_z = 1$ for all $b \in B$' assures that all paths from leaf to the root contain exactly one cluster, i.e. it makes sure that the resulting partition does not overlap.

[15] From the German words *'Blatt'*, *'Wurzel'* and *'Zweig'*.

It is common to exclude the root cluster containing all clustered points from the optimization function (i.e. skipping δ_1), since clustering the complete input into one cluster is most often an undesirable outcome. However, nothing technical stops us from including the root, and it is provided as an optional parameter in all HDBSCAN implementations.

4.3.3 Optimizing our objective

Now that we have defined f_D we proceed to optimize it. To do so, we can greedily process all nodes in the cluster tree (except for the root in our case), starting from the leaves, and decide at any cluster γ_i whether the γ_i itself or the best selection of clusters in the substrees of γ_i yields the highest possible persistence. If we let $K_i := \{k \mid \gamma_k \text{ is a child node of } \gamma_i\}$, and let $U_i := \{u \mid \gamma_u \text{ is a descendant of } \gamma_i\}$ denote all clusters in the subtrees of γ_i, then the highest possible persistence of the subtree with γ_i as its root, denoted by $\bar{P}(\gamma_i)$, can be determined in a recursive fashion as

$$\bar{P}(\gamma_i) = \begin{cases} P(\gamma_i) & \text{if } i \in B \\ \max\left(P(\gamma_i), \sum_{k \in K_i} \bar{P}(\gamma_k)\right) & \text{if } i \notin B. \end{cases}$$

This yields the following algorithm, which can be optimized to have a running time (asymptotic) complexity of $O(\kappa)$:

Algorithm 8 Optimizing HDBSCAN

input: set of clusters Γ in the cluster tree (based on output of algorithm 7).
initialize: $\delta_2 = \cdots = \delta_\kappa = 1$
set $\bar{P}(\gamma_b) := P(\gamma_b)$ for all $b \in B$.
for all $b \in B$:
    for all $z \in W_b$:
        if $P(\gamma_i) < \sum_{k \in K_i} \bar{P}(\gamma_k)$:
            set $\bar{P}(\gamma_i) := \sum_{k \in K_i} \bar{P}(\gamma_k)$ and set $\delta_i := 0$.
        else:
            set $\bar{P}(\gamma_i) := P(\gamma_i)$ and set $\delta_u := 0$ for all $u \in U_i$.
return $\delta_2, \ldots, \delta_\kappa$.

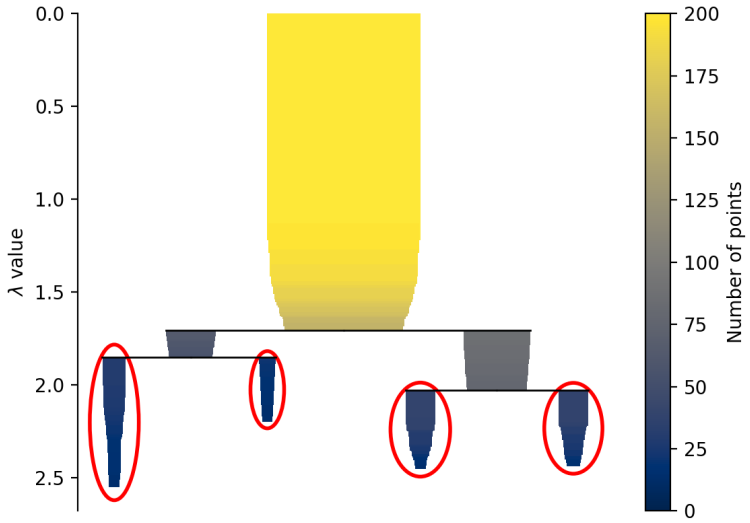


Figure 4.4: The cluster tree from figure 4.3, with the chosen clusters based on f_D circled red. The large root cluster is skipped in the optimization analysis, although it has the largest persistence of all the clusters in the tree.

5 Experiment

In this section, we will outline all the steps of our experiment. It is written such that it can be read independently from the previous and following sections. Therefore, whenever we refer to concepts already introduced elsewhere, we will link to its corresponding section in the thesis. An overview of the experiment can be found in figure 5.1. In short, we will start off by discussing the data set we will use from the ENIGMA-ASD workgroup. Then, we will preprocess the data, including excluding some subjects with missing data, resulting in a neurotypical (D_{NT}) and ASD cohort (D_{ASD}). Then, we will use the two cohorts to build a normative model that functions as the input to our two clustering algorithms. We will discuss the (clinical) properties of the found clusters in the section 6.

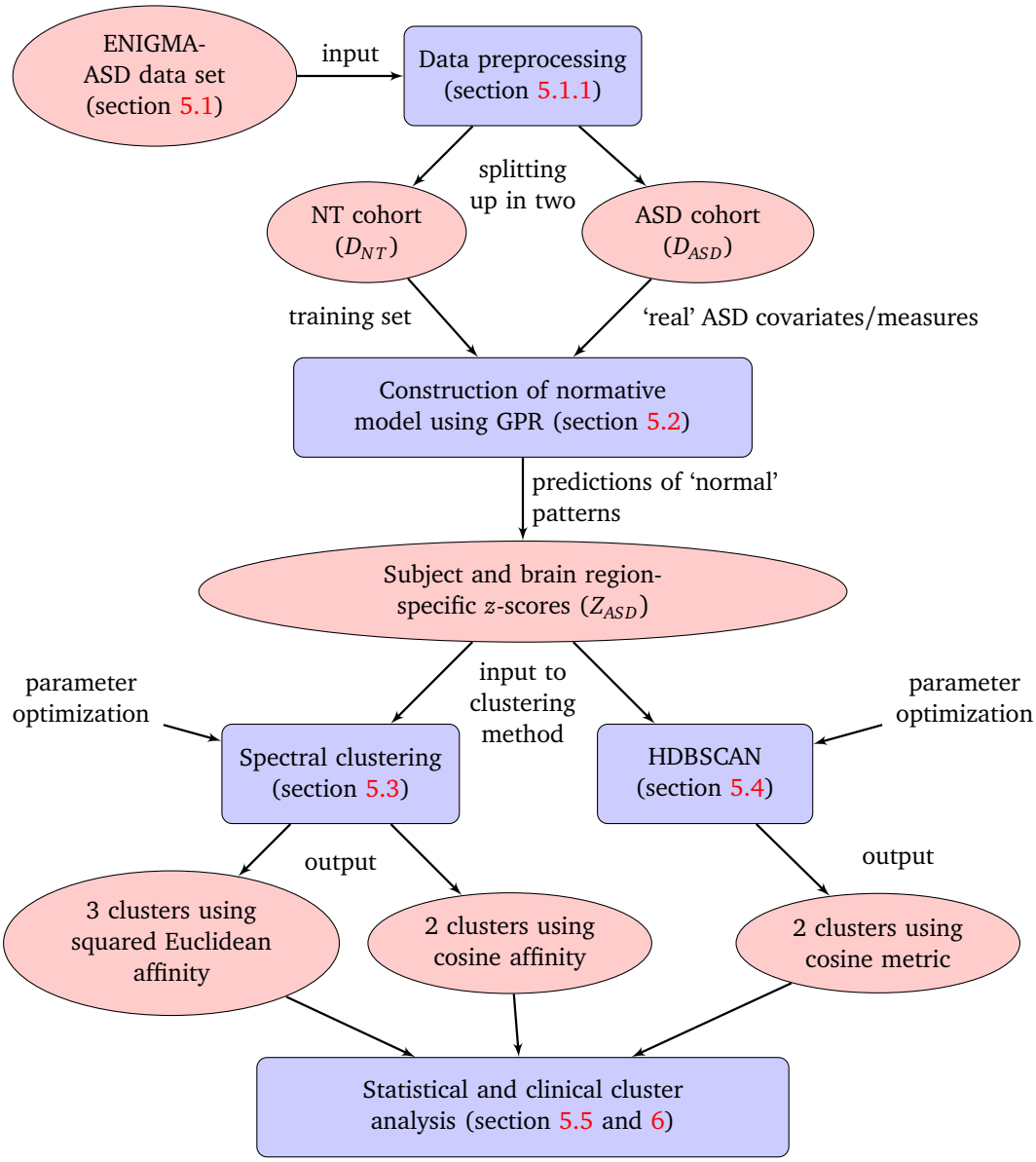


Figure 5.1: Schematic, simplified overview of our experiment, presented as a flow diagram following the structure of this section. (Red) rounds indicate datasets/clusterings, while (blue) blocks indicate the different steps in the experiment.

5.1 The ENIGMA-ASD dataset

Although neuroimaging techniques have been used extensively to analyse brain structures and functions of individuals with neurodevelopmental disorders like ASD, two difficulties have persisted over the years. Firstly, results have been attenuated by small sample sizes, making powerful statements about the obtained results difficult. Furthermore, the methods used during preprocessing and analysis have differed between research groups, making comparison and replication studies harder [41, 92]

The ENIGMA-ASD working group, founded in 2014, aims to overcome these problems. It combines structural T1-weighted MRI data [78] from over 50 research groups worldwide (as of September 2021), bringing the total number of included individuals to 3671 (1833 of which are people with ASD). To address heterogeneity between the contributing sites (e.g. due to different equipment), all data processing within ENIGMA is subject to a standardized quality protocol (http://enigma.ini.usc.edu/protocols/imaging-protocols/). Ultimately, the MRI scans are segmented into different brain regions of interest according to the Desikan-Killiany atlas [25], using the open source FreeSurfer software (http://surfer.nmr.mgh.harvard.edu/). The data that is then extracted consists of 8 subcortical volume measures and 34 cortical thickness and surface area measures for each participant. Logically, cortical volume is a product of both thickness and surface area, which means changes in volume must be do the either one or both of these cortical measures [55]. The quality control also includes, among other things, exclusion of measuring malfunction or anomalies in the segmentation process.

Besides the clinical measures, covariates such as age and IQ scores are included as well; some sites also include ADOS scores for the ASD cohort. ADOS – the Autism Diagnostic Observation Schedule – is widely used as part of diagnostic procedures, and the outcomes of the test are used as an indication of ASD severity [16, 24].

Further details on the specifics of the data set can be found in [83]. The most important demographic data from the data set can be found in figure 5.2.

5.1.1 Data preprocessing

For this research, we have excluded subjects missing one or more clinical and/or covariate measures. The only exception to this were subjects only missing ADOS scores. Subsequent analyses using ADOS (in section 5.5) therefore only consider subjects for whom this score is available.

	ASD ($N = 1138$)		NT ($N = 1169$)	
	Mean	SD	Mean	SD
Age	16.4	8.9	16.2	8.5
IQ	104.5	19.2	111.7	14.9
ADOS	13.7	8.7	-	-
	N	%	N	%
Female	173	15.2	274	23.4

Figure 5.2: Demographic data from the ENIGMA-ASD data set. ADOS score data is based on the subjects for which this score was available (around 62% of the ASD cohort).

Clinical measures are extracted for both left and the right hemisphere, and we have used the mean of the two in our analysis.

5.2 Normative modelling

In a case-control study, two groups of research participants are compared: those having some (medical) condition, and a very similar, 'healthy', control group not having this condition. Then, researches can determine to what extent certain clinical variables are associated with this condition. While this has been successful in many areas of medicine, it poses problems for studying neurodevelopmental disorders like ASD which is known to be highly heterogeneous in terms of symptom expression, behavior and underlying biology. In the

case-control approach however, it is assumed that the two groups are well-defined, clearly distinguishable and internally homogenous [98, 97, 58, 52].

To chart this heterogeneity, we will use a technique called normative modelling. Instead of comparing complete cohorts (as in the case-control approach), normative modelling assesses the deviation of the 'real' clinical measures of someone with ASD from a prediction of 'normal' measures for this individual based on the NT cohort. In other words, we like to predict what the clinical measures of the subject would be if they were in the non-ASD group, and calculate how much the actual measured values differ from this 'normal pattern'. We will make this prediction based on the covariates age, sex, and IQ, as well as (dummy coded) site variables, which are readily available across the data set. In some research using the ENGIMA-ASD dataset [47], only males are considered because of a lack of females with ASD in the data set. However, we choose to correct for sex (among the other covariates) by adding it as a coviariate in our normative model. We have also tried including (combinations of) intercranial volume, total cortical thickness and cortical surface area as covariates, but this did not significantly alter results.

5.2.1 Predicting the normal patterns

Formally, let us define $X = \mathbb{R}^m$ to be the space of input variables (in our case, covariate measures), and $Y = \mathbb{R}^{m'}$ as the space of predictor variables (in our case, clinical measures). Then, we denote the set of NT subjects as

$$D_{NT} := \{(\mathbf{x_i}, \mathbf{y_i}) : 1 \leq i \leq n\},$$

comprising of covariate/clinical measure pairs $\mathbf{x_i}$ and $\mathbf{y_i}$ for every subject. Analogously, we define

$$D_{ASD} := \{(\mathbf{x'_j}, \mathbf{y'_j}) : 1 \leq j \leq n'\}$$

representing the ASD cohort.

Our goal now is to learn a function $f : X \to Y$, using D_{NT} as our training data, such that f returns a prediction $f(\mathbf{x}'_j)$ of the clinical measures given unseen covariates $\mathbf{x}'_j \in X$, that will stem from D_{ASD}. Since f was trained on D_{NT}, we interpret these predictions as an estimation of 'normal' clinical measures for the individual j.

To learn this function f, we will follow recent research [97] and use Gaussian Process Regression (GPR), which is briefly described in appendix B. Ultimately, the GPR algorithm returns given input $\mathbf{X}' := \left[\mathbf{x'_1}, \ldots, \mathbf{x'_{n'}}\right]^T$, not just single prediction values, but mean predictions $m(\mathbf{x'_j})$ with prediction variance $v(\mathbf{x'_j})$ for all $1 \leq j \leq n'$.

5.2.2 Choosing a kernel

The most important parameter to tweak when using GPR is the choice of the kernel that determines the pairwise covariances of the input data. (again see appendix B). In our implementation, we have chosen the RBF kernel, which is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / (2l^2)\right),$$

where l is a parameter called the length scale, which can be constant for all m dimensions of X or be modelled as a vector $\boldsymbol{\lambda} := [l_1, \cdots, l_m]^T$ with different length scales for each dimension. RBF has over the years become the de-facto standard kernel used in GPR. It has the property that it is infinitely differentiable, which means the functions sampled from the Gaussian Process prior turn out very smooth [28]. Moreover, it is stationary, which means the covariance between two points calculated by the kernel only depends on the difference between those points. Lastly, length scale optimization is built in by the `sklearn` implementation of the kernel.

Sometimes, two kernels are combined by either addition or multiplication, which yields a new kernel that inherits and combines features from both originals. Inspired by [98], we tested combinations of RBF with the (non-stationary) dot kernel, defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j.$$

This yielded similar predictions compared to using 'only' RBF.

5.2.3 Quantifying deviation

Having obtained this prediction, for every individual j in the ASD cohort ($1 \leq j \leq n'$), and every clinical measure r ($1 \leq r \leq m'$), we calculate our definitive measure of deviation as a single z-score z_{jr} as follows:

$$z_{jr} = \frac{\mathbf{y}'_j - m(\mathbf{x}'_j)}{\sqrt{v(\mathbf{x}'_j) + v_r}},$$

where v_r denotes the variance in clinical measure r determined from D_{NT} as follows:

$$v_r = \frac{1}{n}\sum_{i=1}^{n}(y_i^r - m(\mathbf{y}^r))^2,$$

where y_i^r denotes the r-th entry of $\mathbf{y}_i$ and $m(\mathbf{y}^r)$ denotes the mean of the vector $\mathbf{y}^r$ comprised of all such y_i^r.

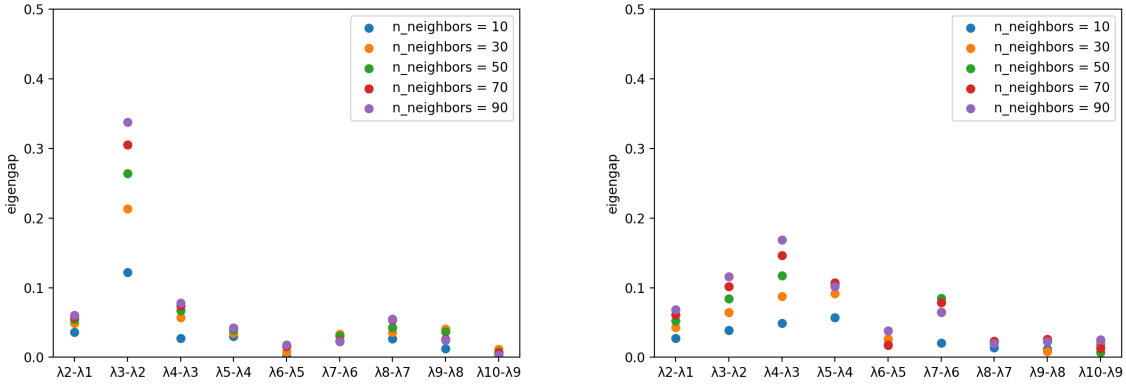
Each z_{jr} reflects how much their measure deviates from the normative model in terms of units of standard deviation. v_r is added to reflect true underlying variability between clinical measures, which cannot be reduced by adding more data, while $v(\mathbf{x}'_j)$ reflects uncertainty about our GPR prediction that may be minimized by using a larger training set. Methods omitting GPR variance have been proposed, but have been prone to yield (overly) optimistic results [51]

The resulting data set, denoted Z_{ASD}, comprised of all z_{jr} is the one we will use as input to the clustering algorithms.

5.3 Spectral clustering

We implement spectral clustering with a nearest neighbor affinity matrix $\mathbf{W}$ using both cosine and squared Euclidean affinity. For the nearest neighbor graph, we will need to choose the number of neighbors m.

Furthermore, a major assumption when applying k-means clustering is that the amount of clusters k is known a priori. Since spectral clustering internally also uses k-means (see algorithm 3), we need to establish which value for k best fits our data. For this, an often-used method that stems from matrix pertubation theory is called the 'eigengap' heuristic, which involves the eigenvalues $\lambda_1, \ldots, \lambda_n$ of the Laplacian $\mathbf{L}$ (discussed in section 3.2.2). In an ideal case where there exist k completely disconnected clusters, the eigenvalue 0 should have multiplicity k, i.e. $\lambda_1 = \cdots = \lambda_k = 0$. Hence, there exists an 'eigengap' $|\lambda_{k+1} - \lambda_k| > 0$. In the general case, we are looking for the number k such that the first k eigenvalues are relatively small, and λ_{k+1} relatively big, i.e. the largest eigengap [84, 14].



(a) Cosine affinity

(b) Squared euclidean affinity

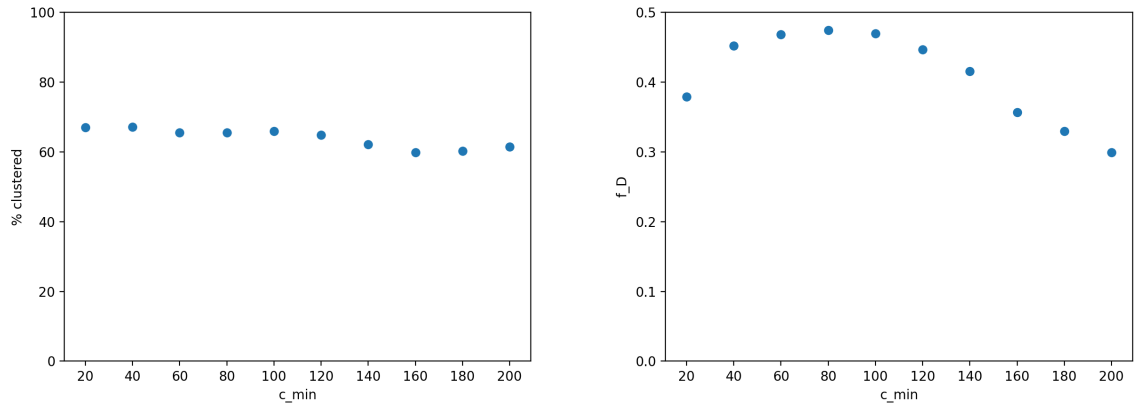
Figure 5.3: Estimated eigengaps for the first 10 eigenvalues of $\mathbf{L}$. The labels on the x-axis indicate eigengaps, e.g. $\lambda 3 - \lambda 2$ refers to the gap between the second and third eigenvalue. `n_neighbors` refers to the parameter m that determines the number of neighbors used to construct G.

Eigengaps for both affinities can be found in figure 5.3. We see that for cosine and squared euclidean affinity, the largest eigengap is achieved for $k = 2$ and $k = 3$ respectively.

We also esimate of the effect of the parameter `n_neighbors` using the mean silhouette score (discussed in section 2.4.2). Silhouette scores are stable overall, and the resulting plots can be found in C. We ended up picking `n_neighbors` $= 50$ for both affinities.

5.4 HDBSCAN

We have implemented HDBSCAN using cosine similarity to compute core distances (see section 4.1.3). The main parameter to tweak is $c_{\min}$, that determines the minimum size clusters can have before they are considered noise (see section 4.2.4). The results can be found in figure 5.4. Interestingly, the percentage of points clustered – i.e. not declared as noise – was stable given different values for $c_{\min}$ (see figure 5.4a). The highest cluster prominence is found for $c_{\min} = 80$, whence it is the value we proceed with (see figure 5.4b).



(a) Percentage of points

(b) HDBSCAN optimization objective

Figure 5.4: (a) Effects of $c_{\min}$ on the percentage of points clustered by the HDBSCAN algorithm. (b) Effects of $c_{\min}$ on the HDBSCAN optimization objective f_D, which is the sum of cluster prominences described in section 4.3.2.

5.5 (Statistical) cluster analysis

At a subject level, we assessed deviations of clinical measures from the normative model in three ways. Firstly, we computed for each subject the number of significant absolute deviations (NSD), where a deviation for subject j and clinical measure r was deemed significant if $|z_{jr}| > 1.96$, i.e. outside a 95% confidence interval.[16] Secondly, we took a trimmed mean [98] of 1% of top absolute deviations (MTD). Lastly, we compiled a ranking of the top 15 absolute deviations to get a ranking of deviating clinical measures (TOP).

For every cluster, we computed the mean and standard deviation of the covariates introduced in figure 5.2. Then, we computed Pearson correlation coefficients r_{ADOS} between ADOS scores – when available – and both NSD and MTD outcomes. Furthermore, we added the TOP rankings of all subjects together to yield a clusterwide TOP ranking, to analyse which clinical measures manifest themselves most per cluster.

5.6 Python

The experiment is performed entirely in Python, aided by standard algebra packages such as `numpy` and `scipy`. For GPR and spectral clustering, `scikit-learn` was used, while HDBSCAN uses an accelarated im-

[16] since we are talking about z-scores, the null hypothesis is set at $z_{jr} = 0$ for all $1 \leq j \leq n'$, $1 \leq r \leq m'$, indicating no difference between the subjects in D_{ASD} and D_{NT}.

plementation based on [54]. The entire implementation can be found at https://github.com/dnndbrkt/stratifyingautism.

6 Results

6.1 Spectral clustering

6.1.1 Cosine affinity

Performing spectral clustering with $k = 2$ and cosine affinity resulted in two clusters that split Z_{ASD} almost exactly in halves figure 6.1. When inspecting TOP rankings (figure 6.9 and appendix D), we see that both clusters are oriented mostly towards regions in the frontal and parietal/occipital lobes, and the striatum. Cluster 1 seems to be represented more by lateral components (e.g. middle and inferior temporal gyri and the banks of the superior temporal sulcus), while cluster 2 contains more medial components (e.g. the isthmus of the cingulate cortex and posterior cingulate). Cluster 2 showed significant correlations with both NSD and MTD outcomes ($p \ll 0.0001$), while cluster 1 did not (see figure 6.2).

	Cluster 1 ($N = 548, 48.2\%$)		Cluster 2 ($N = 590, 51.8\%$)	
	Mean	SD	Mean	SD
Age	16.5	10.2	16.2	7.6
IQ	103.5	20.1	105.4	18.3
ADOS	12.7	7.1	14.9	10.0
	N	%	N	%
Female	118	21.5	55	9.3

Figure 6.1: Demographic data on spectral clustering results using cosine affinity.

	Cluster 1 ($N = 548, 48.2\%$)			Cluster 2 ($N = 590, 51.8\%$)		
	Mean	r_{ADOS}	p	Mean	r_{ADOS}	p
NSD	1.93	0.02	0.76	2.72	0.23	$3.29 \cdot 10^{-5}$
MTD	2.04	0.11	0.04	2.39	0.25	$3.72 \cdot 10^{-6}$

Figure 6.2: NSD/MTD of spectral clustering using cosine affinity.

6.1.2 Squared Euclidean affinity

Using squared Euclidean affinity and $k = 3$ resulted in clusters containing 13.4%, 32.2%, and 54.4% of Z_{ASD} subjects. Of this, cluster 1 was characterized by lower age, IQ and ADOS scores (on average), while cluster 2 and 3 did not differ significantly from the overall Z_{ASD} demographic (see figure 6.3). While cluster 1 showed no significant correlation with NSD or MTD outcomes, cluster 2 and 3 did (figure 6.4). Cluster 1 is represented mostly by frontal and parietal regions, and showed no significant correlations with NSD or MTD outcomes. Cluster 2 showed more deviations in the frontal, parietal and occipital regions, while cluster 3 showed more frontal, striatal and temporal components (see figure 6.7 and appendix D). Since the parietal lobe is associated mostly with sensory processing and the occipital lobe deals with all aspects of vision, cluster 2 could be related to the sensory abnormalities that prevail significantly in people with ASD [64]. Moreover, connections between frontal and temporal lobe have been associated with processing social information [3], whence we may relate cluster 3 to the social impairments often seen in people with ASD.

	Cluster 1 ($N = 152, 13.4\%$)		Cluster 2 ($N = 366, 32.2\%$)		Cluster 3 ($N = 620, 54.4\%$)	
	Mean	SD	Mean	SD	Mean	SD
Age	13.1	9.0	16.7	7.7	17.0	9.4
IQ	96.5	22.9	106	17.9	105.6	18.4
ADOS	11.9	4.3	15.2	10.4	13.5	8.5
	N	%	N	%	N	%
Female	49	32.2	34	9.3	90	14.5

Figure 6.3: Demographic data on spectral clustering results using squared Euclidean affinity.

	Cluster 1 ($N = 152, 13.4\%$)			Cluster 2 ($N = 366, 32.2\%$)			Cluster 3 ($N = 620, 54.4\%$)		
	Mean	r_{ADOS}	p	Mean	r_{ADOS}	p	Mean	r_{ADOS}	p
NSD	5.46	0.04	0.67	3.99	0.26	$2.56 \cdot 10^{-4}$	0.64	0.23	$4.36 \cdot 10^{-6}$
MTD	2.43	0.03	0.80	2.62	0.26	$2.06 \cdot 10^{-4}$	1.24	0.24	$3.38 \cdot 10^{-6}$

Figure 6.4: NSD/MTD of spectral clustering using squared Euclidean affinity.

6.2 HDBSCAN

With HDBSCAN, two clusters were found consisting of 34.8% and 30.8% of points. The rest (34.4%) was classified as noise (see figure 6.5). The TOP ranking (figure 6.8 and appendix D) of cluster 1 showed regions in all different brain lobes, and ADOS scores correlated significantly with NSD and MTD (figure 6.6). Cluster 2 showed no significant correlations and contained mostly frontal regions. We determined TOP rankings of the 'noise' cluster as well, which featured mostly subcortical volumes and correlated significantly with ADOS scores. Many studies have reported anomalies in subcortical volumes among people with ASD [99], which might give an explanation for this phenomenon.

	Cluster 1 ($N = 396, 34.8\%$)		Cluster 2 ($N = 350, 30.8\%$)	
	Mean	SD	Mean	SD
Age	16.7	7.9	15.8	10.1
IQ	106.2	17.7	101.3	20.6
ADOS	14.9	10.1	12.5	6.6
	N	%	N	%
Female	36	9.3	87	24.9

Figure 6.5: Demographic data on HDBSCAN results using cosine metric. Of the 1138 subjects in Z_{ASD}, 392 were classified as noise (34.4%).

	Cluster 1 ($N = 396, 34.8\%$)			Cluster 2 ($N = 350, 30.8\%$)		
	Mean	r_{ADOS}	p	Mean	r_{ADOS}	p
NSD	3.71	0.27	$6.23 \cdot 10^{-5}$	2.77	0.02	0.75
MTD	2.56	0.27	$6.87 \cdot 10^{-5}$	2.14	0.07	0.32

Figure 6.6: NSD/MTD of HDBSCAN using cosine metric.

6.3 General observations

Over all methods, we have found that cortical thickness measures did not appear in any of the TOP rankings. This suggests that deviations of cortical thickness from normative models are less apparent than deviations from subcortical volume cortical surface area. This is in line with some current research suggesting that cortical thickness is comparable between ASD and NT subjects, but people with ASD see significantly greater surface area in some cortical regions [60, 40] . In [55], researchers found indications that cortical thickness decreases over time in NT subjects but remains stable for subjects with ASD. However, other research has found subtle yet significant deviations in cortical thickness, mainly in the frontal and temporal lobe - one even using the same ENIGMA-ASD data set, see [83]. Furthermore, a normative modelling approach using a different dataset revealed small subgroups of people with ASD with highly deviating cortical thickness, but this was only observed in children and young adults [11].

Furthermore, it seems like NSD and MTD do an equally good job at determining correlations with ADOS scores, which is what we would expect since the two measures also show a natural correlation: subjects with higher amounts of significant deviations are likely to have a higher mean of top deviations as well. The strength of the correlation is comparable to other research [98, 37].

In both the cosine (cluster 1) and squared Euclidean (cluster 1) spectral clusterings, as well as the HDBSCAN clustering (cluster 2), we have found a cluster that is uncorrelated to ADOS scores for both NSD and MTD outcomes. In both cases, this cluster is associated with regions mostly in the frontal lobe. A possible explanation might be that frontal lobe surface area seems to decrease with age [49] especially in people with ASD, which means it is not corrected for in the constructing of the normative model since it is based on neurotypical subjects where this developmental phenomenon is less pronounced [38]. At the same time, a study concerning symptom severity and and cortical surface area found no significant correlations between frontal lobe surface area and ADOS scores [27], which agrees with our findings.

Another possible explanation for the found disconnection between frontal lobe surface area and ADOS could be the potential comorbidity with other disorders or covariates that are missing in the used data set. For example, (small-scale) studies have shown alterations in frontal lobe surface area based on sexual orientation [89] and higher genetic heritability compared to the surface area of other lobes [32]. This can indicate that this region may vary within the general population, but might be less important in the analysis of ASD.

Clusters with significant correlations between NSD/MTD and ADOS scores do exhibit combinations of frontal-parietal, frontal-temporal or frontal-occipital regions in their TOP rankings. This aligns with studies using task-based functional MRI scans, that have also found alterations in this functional connectivity between lobes in people with ASD, such as using an executive function task (frontal-parietal, e.g. in [75] and [43]) during language comprehension (frontal-parietal, e.g. in [44]), visuospatial processing (frontal-parietal/occiptal, e.g. in [23]), visual change detection (frontal-occipital, e.g. in [20]) and social processing (frontal-parietal/temporal/occiptal, e.g. in [45]). It might thus be these functional brain connectivity structures the clustering algorithms pick up.

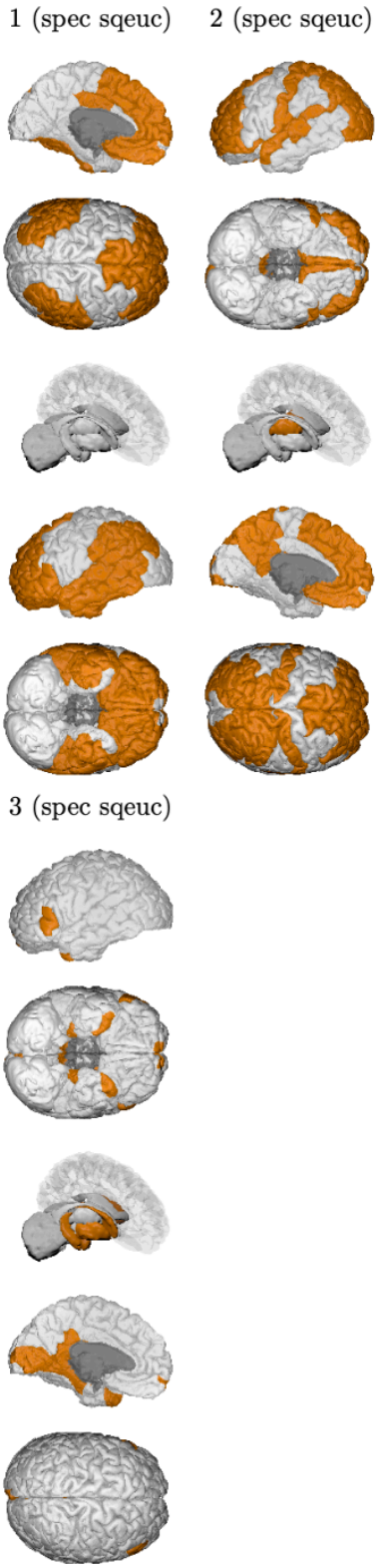


Figure 6.7: Visualized TOP ranking of spectral clustering using squared Euclidean affinity, based on figure D.1, D.2 and D.3. Image generated using BrainPainter [50]

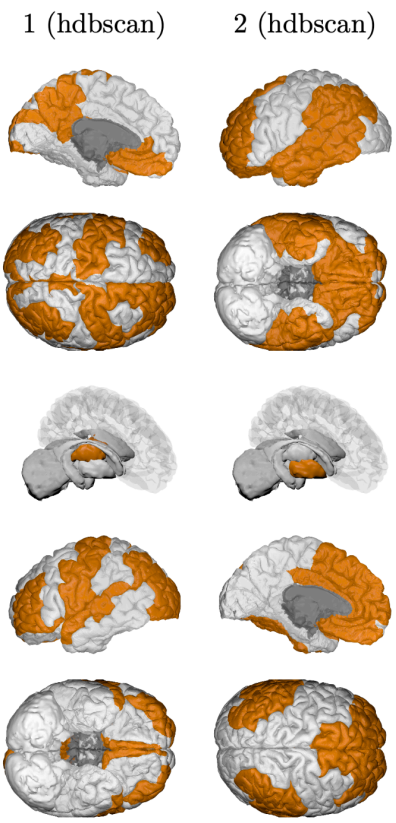


Figure 6.8: Visualized TOP ranking of HDBSCAN using cosine metric, based on figure D.6 and D.7. Image generated using BrainPainter [50]

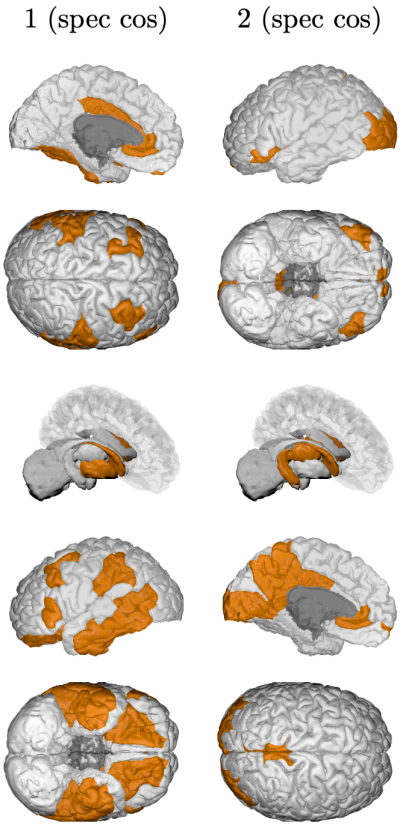


Figure 6.9: Visualized TOP ranking of spectral clustering using cosine affinity, based on figure D.4 and D.5. Image generated using BrainPainter [50]

7 Conclusion

In this thesis, we aimed to stratify structural MRI data from people with ASD using cluster analysis, and identify brain regions of interest associated with the resulting clusters. The data, stemming from the ENIGMA-ASD workgroup, contained subcortical volume as well as cortical surface area and thickness measures. As a theoretical basis, we explored three clustering methods (in section 2,3 and 4), each having their own approach towards clustering, and showed their corresponding optimization objectives and algorithmic implementations. For spectral clustering, we proved that a relaxed objective can be minimized by solving a generalized eigenvalue problem, while the discussion of the (density-based) HDBSCAN algorithm has a minimization objective that can be optimized to run in linear time in terms of the number of clusters.

Next, we constructed a normative model for the ASD cohort using Gaussian Process Regression (section 5.2), which was used to quantify deviation of the clinical measures from a normal pattern. This model was used as an input for the clustering methods. Finally, we analysed the output both statistically and clinically (section 6) by computing correlations between available ADOS scores and two measures based on (significant) deviation of ASD clinical measures from the normative model, and by analysing brain regions of interest that deviated the most often per cluster.

The analyses showed multiple clusters across the different algorithms having significant correlations with ADOS scores, while other clusters did not. Clusters that did show correlation were mostly associated with interaction between the frontal lobe and either the parietal, temporal or occipital lobe, while uncorrelated clusters were associated with regions only in the frontal lobe. Note that these analyses are based only on the cortical surface area and subcortical volume measures since, interestingly, cortical thickness measures did not show up in the most deviating clinical measures in any of the resulting clusters.

When comparing HDBSCAN and Spectral clustering, interpretability and results did not differ significantly between the two methods. In particular, HDBSCAN did not find stronger correlations with ADOS scores or clusters that were easier to clearly interpret, despite having the ability to exclude outlier points from the discussion. This finding is at variance with our hypothesis that HDBSCAN would perform better in this regard. As briefly alluded to in the introduction, this does not mean that the discussion concerning the toy data set - upon which we based our hypothesis - is false; what it does show, however, is that such simple, two-dimensional situations do not necessarily translate well to a real-life setting. All algorithms are based on assumptions that simplify reality, and may just as well fail if these assumptions turn out not to hold.[17] While choosing the right set of assumptions given a data set may be hard in its own right – and requires a priori knowledge on the (latent) structure of the data at hand –, it poses an even bigger issue in the unsupervised setting, where discovering said structure is precisely the goal of the learning task itself [1, 85, 9]. As for the choice of clustering methods, a lot of effort is therefore put into algorithmic details such as (computational) complexity or minor implementation differences compared to other algorithms, offering mostly general analysis applicable to all kinds of data [9]. However, approaches such as interactive semi-supervised learning that try to break this boundary and incorporate interaction between the algorithm (designer) and the end user have also been proposed [88, 7] . Until the right tools for this approach have been developed, however, finding the right clustering method may continue to be rather ad hoc than a rigorous endeavor.

This leaves us with the question: where do we go from here? Based on the found clusters, we suggest further research into the relation between the severity of ASD symptoms and interaction between the frontal and other three major lobes of the brain. Furthermore, since large deviations within (only) the frontal lobe showed little correlation with ADOS scores, we suggest research into whether these deviations may be due to variations of frontal lobe size within the complete population, unrelated to ASD. The question of whether deviations in cortical thickness are related to ASD symptoms has been the subject of a number of studies already, with conclusions jumping back and forth on both the existence and the strongness of such relation. While the results found here argue against its existence, follow-up studies are needed to assess the robustness of that claim. For example, research using the (same) ENIGMA dataset [46] has suggested an association of ASD with lower hemispheric asymmetry of cortical thickness. Since we consistently took the mean value over both hemispheres (see section 5.1.1), possible inter-hemispheric differences may have been leveled out. It may therefore be interesting to repeat this study with separate measures for both hemispheres.

Furthermore, research concerning high-dimensional data usually includes some dimensionality reduction

[17] Actually, this is a paraphrase of a rather charming construction in mathematics called the 'No Free Lunch' theorem, first introduced by David Wolpert and William Macready [94]). It states that if some algorithm performs well given a certain set of problems, it must necessarily pay for that with worse performance on all other problems.

such as (kernel) principal/independent component analysis [74, 61], as the 'raw' data may be difficult to analyse due to its sparsity, which is the consequence of a phenomenon known as the curse of dimensionality [8]. As for this thesis, spectral embedding – as applied in algorithm 3 (p. 23) – can also be seen as a form of (non-linear) dimensionality reduction [57]. Since this step is contained within the spectral clustering paradigm itself, we have chosen not to perform a similar manipulation to the data for HDBSCAN, to maintain a level playing field between the two methods. Adding said reduction may improve its performance though, and might give HDBSCAN a slight advantage over spectral clustering. Not to mention, the brain regions that make up the dimensions of the data are profoundly interconnected and -correlated, which gives a clinical argument in favor of dimensinality reduction too.

Finally, what does all this discussion entail for the diagnosis of ASD? Up until now, structural MRI studies to find biomarkers for ASD have brought many conflicting findings regarding brain volume and surface area abnormalities, though many of these studies have been uncomparable due to differences in sample size, processing of data or participant demographics. As databases - such as ENIGMA's - grow and start adding more relevant clinical variables and descriptions for its participants, it is a matter of time before we will see which of these results will hold up. By incorporating HDBSCAN into the debate, a relatively new approach to clustering that combines traits from previous paradigms, we hope to have added just another small piece to this enormous puzzle.

Bibliography

[1] Karim T Abou-Moustafa and Dale Schuurmans. Generalization in unsupervised learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 300–317. Springer, 2015.

[2] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AML-Book New York, NY, USA:, 2012.

[3] Matthew Ainsworth, Jérôme Sallet, Olivier Joly, Diana Kyriazis, Nikolaus Kriegeskorte, John Duncan, Urs Schüffelgen, Matthew FS Rushworth, and Andrew H Bell. Viewing ambiguous social interactions increases functional connectivity between frontal and temporal nodes of the social brain. *Journal of Neuroscience*, 41(28):6070–6086, 2021.

[4] Francis J Anscombe. Graphs in statistical analysis. *The american statistician*, 27(1):17–21, 1973.

[5] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[6] American Psychiatric Association. *Diagnostic and statistical manual of mental disorders*. American Psychiatric Publishing, 2013.

[7] Eric Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[8] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[9] Shai Ben-David. Clustering-what both theoreticians and practitioners are doing wrong. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] Shai Ben-David, Ulirke von Luxburg, and Dávid Pál. A sober look at clustering stability. *Proceedings of the 19th Annual Conference on Learning Theory*, pages 5–19, 2006.

[11] Richard AI Bethlehem, Jakob Seidlitz, Rafael Romero-Garcia, Stavros Trakoshis, Guillaume Dumas, and Michael V Lombardo. A normative modelling approach reveals age-atypical cortical thickness in a subgroup of males with autism spectrum disorder. *Communications biology*, 3(1):1–10, 2020.

[12] Otakar Borůvka. O jistém problému minimálním. 1926.

[13] Matthew Brand and Kun Huang. A unifying theorem for spectral embedding and clustering. In *International Workshop on Artificial Intelligence and Statistics*, pages 41–48. PMLR, 2003.

[14] Gabriel Budel and Piet Van Mieghem. Detecting the number of clusters in a network. *Journal of Complex Networks*, 8(6):cnaa047, 2020.

[15] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[16] Themba Carr. *Autism Diagnostic Observation Schedule*, pages 349–356. Springer New York, 2013.

[17] Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *NIPS*, pages 343–351. Citeseer, 2010.

[18] Rong Chen, Yun Jiao, and Edward H Herskovits. Structural mri in autism spectrum disorder. *Pediatric research*, 69(8):63–68, 2011.

[19] Karri Chiranjeevi and Uma Ranjan Jena. Image compression based on vector quantization using cuckoo search optimization technique. *Ain Shams Engineering Journal*, 9(4):1417–1431, 2018.

[20] H Clery, F Andersson, F Bonnet-Brilhault, A Philippe, B Wicker, and M Gomot. fmri investigation of visual change detection in adults with autism. *NeuroImage: Clinical*, 2:303–312, 2013.

[21] Edward Thomas Copson. *Metric spaces*. Number 57. CUP Archive, 1988.

[22] B Cung, T Jin, Juan Ramirez, A Thompson, Christos Boutsidis, and Deanna Needell. Spectral clustering: An empirical study of approximation algorithms and its application to the attrition problem. *arXiv preprint arXiv:1211.3444*, 2012.

[23] Saudamini Roy Damarla, Timothy A Keller, Rajesh K Kana, Vladimir L Cherkassky, Diane L Williams, Nancy J Minshew, and Marcel Adam Just. Cortical underconnectivity coupled with preserved visuospatial cognition in autism: Evidence from an fmri study of an embedded figures task. *Autism Research*, 3(5):273–279, 2010.

[24] Annelies de Bildt, Iris J Oosterling, Natasja DJ van Lang, Sjoerd Sytema, Ruud B Minderaa, Herman van Engeland, Sascha Roos, Jan K Buitelaar, Rutger-Jan van der Gaag, and Maretha V de Jonge. Standardized ados scores: Measuring severity of autism spectrum disorders in a dutch sample. *Journal of autism and developmental disorders*, 41(3):311–319, 2011.

[25] Rahul S Desikan, Florent Ségonne, Bruce Fischl, Brian T Quinn, Bradford C Dickerson, Deborah Blacker, Randy L Buckner, Anders M Dale, R Paul Maguire, Bradley T Hyman, et al. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *Neuroimage*, 31(3):968–980, 2006.

[26] Gabriel S Dichter. Functional magnetic resonance imaging of autism spectrum disorders. *Dialogues in clinical neuroscience*, 14(3):319, 2012.

[27] Krissy AR Doyle-Thomas, Azadeh Kushki, Emma G Duerden, Margot J Taylor, Jason P Lerch, Latha V Soorya, A Ting Wang, Jin Fan, and Evdokia Anagnostou. The effect of diagnosis, age, and symptom severity on cortical surface area in the cingulate cortex and insula in autism spectrum disorders. *Journal of child neurology*, 28(6):732–739, 2013.

[28] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[29] Jennifer Harrison Elder, Consuelo Maun Kreider, Susan N. Brasher, and Margaret Ansell. Clinical impact of early diagnosis of autism on the prognosis and parent–child relationships. *Psychol Res Behav Manag.*, (10):283–292, 2017.

[30] Taban Eslami, Fahad Almuqhim, Joseph S. Raiker, and Fahad Saeed. Machine learning methods for diagnosing autism spectrum disorder and attention-deficit/hyperactivity disorder using functional and structural mri: A survey. *Frontiers in Neuroinformatics*, 14:62, 2021.

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[32] Lisa T Eyler, Elizabeth Prom-Wormley, Matthew S Panizzon, Allison R Kaup, Christine Fennema-Notestine, Michael C Neale, Terry L Jernigan, Bruce Fischl, Carol E Franz, Michael J Lyons, et al. Genetic and environmental contributions to regional cortical surface area in humans: a magnetic resonance imaging twin study. *Cerebral cortex*, 21(10):2313–2321, 2011.

[33] Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. On using class-labels in evaluation of clusterings. In *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*, page 1, 2010.

[34] Richard E Frye, Sarah Vassall, Gurjot Kaur, Christina Lewis, Mohammand Karim, and Daniel Rossignol. Emerging biomarkers in autism spectrum disorder: a systematic review. *Annals of translational medicine*, 7(23), 2019.

[35] Daniel H Geschwind and Pat Levitt. Autism spectrum disorders: developmental disconnection syndromes. *Current opinion in neurobiology*, 17(1):103–111, 2007.

[36] Ronald L Graham, Donald E Knuth, Oren Patashnik, and Stanley Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989.

[37] Shlomi Haar, Sigal Berman, Marlene Behrmann, and Ilan Dinstein. Anatomical abnormalities in autism? *Cerebral cortex*, 26(4):1440–1452, 2016.

[38] Antonio Y Hardan, Roger J Jou, Matcheri S Keshavan, Ravi Varma, and Nancy J Minshew. Increased frontal cortical folding in autism: a preliminary mri study. *Psychiatry Research: Neuroimaging*, 131(3):263–268, 2004.

[39] John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

[40] Heather Cody Hazlett, Michele Poe, Guido Gerig, Rachel Gimpel Smith, James Provenzale, Allison Ross, John Gilmore, and Joseph Piven. Magnetic resonance imaging and head circumference study of brain size in autism: birth through age 2 years. *Archives of general psychiatry*, 62(12):1366–1376, 2005.

[41] Martine Hoogman, Daan Van Rooij, Marieke Klein, Premika Boedhoe, Iva Ilioska, Ting Li, Yash Patel, Merel C Postema, Yanli Zhang-James, Evdokia Anagnostou, et al. Consortium neuroscience of attention deficit/hyperactivity disorder and autism spectrum disorder: The enigma adventure. *Human brain mapping*, 43(1):37–55, 2022.

[42] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[43] Marcel Adam Just, Vladimir L Cherkassky, Timothy A Keller, Rajesh K Kana, and Nancy J Minshew. Functional and anatomical cortical underconnectivity in autism: evidence from an fmri study of an executive function task and corpus callosum morphometry. *Cerebral cortex*, 17(4):951–961, 2007.

[44] Rajesh K Kana, Timothy A Keller, Vladimir L Cherkassky, Nancy J Minshew, and Marcel Adam Just. Sentence comprehension in autism: thinking in pictures with decreased functional connectivity. *Brain*, 129(9):2484–2493, 2006.

[45] Rajesh K Kana, Timothy A Keller, Vladimir L Cherkassky, Nancy J Minshew, and Marcel Adam Just. Atypical frontal-posterior synchronization of theory of mind regions in autism during mental state attribution. *Social neuroscience*, 4(2):135–152, 2009.

[46] Xiang-Zhen Kong, Merel C Postema, Tulio Guadalupe, Carolien de Kovel, Premika SW Boedhoe, Martine Hoogman, Samuel R Mathias, Daan Van Rooij, Dick Schijven, David C Glahn, et al. Mapping brain asymmetry in health and disease through the enigma consortium. *Human Brain Mapping*, 2020.

[47] Ting Li, Martine Hoogman, Nina Roth Mota, Jan Buitelaar, Alejandro Arias Vasquez, Barbara Franke, Daan van Rooij, ENIGMA-ASD Working Group, et al. Dissecting the heterogeneous subcortical brain volume of autism spectrum disorder (asd) using community detection. *bioRxiv*, 2020.

[48] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International workshop on algorithms and computation*, pages 274–285. Springer, 2009.

[49] Kathleen M Mak-Fan, Margot J Taylor, Wendy Roberts, and Jason P Lerch. Measures of cortical grey matter structure and development in children with autism spectrum disorder. *Journal of autism and developmental disorders*, 42(3):419–427, 2012.

[50] Razvan Marinescu, Arman Eshaghi, Daniel Alexander, and Polina Golland. Brainpainter: A software for the visualisation of brain structures, biomarkers and associated pathological processes. *arXiv preprint arXiv:1905.08627*, 2019.

[51] Andre F Marquand, Seyed Mostafa Kia, Mariam Zabihi, Thomas Wolfers, Jan K Buitelaar, and Christian F Beckmann. Conceptualizing mental disorders as deviations from normative functioning. *Molecular psychiatry*, 24(10):1415–1424, 2019.

[52] Andre F Marquand, Iead Rezek, Jan Buitelaar, and Christian F Beckmann. Understanding heterogeneity in clinical cohorts using normative models: beyond case-control studies. *Biological psychiatry*, 80(7):552–561, 2016.

[53] Andre F Marquand, Thomas Wolfers, Maarten Mennes, Jan Buitelaar, and Christian F Beckmann. Beyond lumping and splitting: a review of computational approaches for stratifying psychiatric disorders. *Biological psychiatry: cognitive neuroscience and neuroimaging*, 1(5):433–447, 2016.

[54] Leland McInnes and John Healy. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 33–42. IEEE, 2017.

[55] Vincent T Mensen, Lara M Wierenga, Sarai van Dijk, Yvonne Rijks, Bob Oranje, René CW Mandl, and Sarah Durston. Development of cortical thickness and surface area in autism spectrum disorder. *NeuroImage: Clinical*, 13:215–222, 2017.

[56] Antonio Mucherino, Petraq Papajorgji, and Panos M Pardalos. *Data mining in agriculture*, volume 34. Springer Science & Business Media, 2009.

[57] Donglin Niu, Jennifer Dy, and Michael I Jordan. Dimensionality reduction for spectral clustering. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 552–560. JMLR Workshop and Conference Proceedings, 2011.

[58] Abraham Nunes, Thomas Trappenberg, and Martin Alda. Measuring heterogeneity in normative models as the effective number of deviation patterns. *PloS one*, 15(11):e0242320, 2020.

[59] National Institute of Mental Health. Autism spectrum disorder. Retrieved January 03, 2022, from https://www.nimh.nih.gov/health/topics/autism-spectrum-disorders-asd.

[60] Haruhisa Ohta, Christine Wu Nordahl, Ana-Maria Iosif, Aaron Lee, Sally Rogers, and David G Amaral. Increased surface area, but not cortical thickness, in a subset of young boys with autism spectrum disorder. *Autism research*, 9(2):232–248, 2016.

[61] Erkki Oja and A Hyvarinen. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[62] Paul H Patterson. Maternal infection and immune involvement in autism. *Trends in molecular medicine*, 17(7):389–394, 2011.

[63] Richard Peng, He Sun, and Luca Zanetti. Partitioning well-clustered graphs: Spectral clustering works! In *Conference on Learning Theory*, pages 1423–1455. PMLR, 2015.

[64] Annio Posar and Paola Visconti. Sensory abnormalities in children with autism spectrum disorder. *Jornal de pediatria*, 94:342–350, 2018.

[65] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[66] Ramkripa Raghavan, Anne W Riley, Heather Volk, Deanna Caruso, Lynn Hironaka, Laura Sices, Xiumei Hong, Guoying Wang, Yuelong Ji, Martha Brucato, et al. Maternal multivitamin intake, plasma folate and vitamin b12 levels and autism spectrum disorder risk in offspring. *Paediatric and perinatal epidemiology*, 32(1):100–111, 2018.

[67] Ahmed Sameh and Zhanye Tong. The trace minimization method for the symmetric generalized eigenvalue problem. *Journal of Computational and Applied Mathematics*, 123(1-2):155–175, 2000.

[68] Ahmed H Sameh and John A Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 19(6):1243–1259, 1982.

[69] Jorge M Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pages 175–184. Springer, 2009.

[70] Lauren M Schmitt, Edwin H Cook, John A Sweeney, and Matthew W Mosconi. Saccadic eye movement abnormalities in autism spectrum disorder indicate dysfunctions in cerebellum and brainstem. *Molecular autism*, 5(1):1–13, 2014.

[71] Mohammed Yousef Shaheen. Applications of artificial intelligence (ai) in healthcare: A review. *ScienceOpen Preprints*, 2021.

[72] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[73] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[74] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[75] Marjorie Solomon, Sally J Ozonoff, Stefan Ursu, Susan Ravizza, Neil Cummings, Stanford Ly, and Cameron S Carter. The neural substrates of cognitive control deficits in autism spectrum disorders. *Neuropsychologia*, 47(12):2515–2526, 2009.

[76] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[77] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of classification*, 20(1):25–47, 2003.

[78] M Symms, HR Jäger, K Schmierer, and TA Yousry. A review of structural magnetic resonance neuroimaging. *Journal of Neurology, Neurosurgery & Psychiatry*, 75(9):1235–1244, 2004.

[79] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[80] Kshitij Tiwari and Nak Young Chong. *Multi-Robot Exploration for Environmental Monitoring: The Resource Constrained Perspective*. Academic Press, 2019.

[81] Nicolas Tremblay and Andreas Loukas. Approximating spectral clustering via sampling: a review. *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 129–183, 2020.

[82] Toon Van Craenendonck and Hendrik Blockeel. Using internal validity measures to compare clustering algorithms. *Benelearn 2015 Poster presentations (online)*, pages 1–8, 2015.

[83] Daan Van Rooij, Evdokia Anagnostou, Celso Arango, Guillaume Auzias, Marlene Behrmann, Geraldo F Busatto, Sara Calderoni, Eileen Daly, Christine Deruelle, Adriana Di Martino, et al. Cortical and subcortical brain morphometry differences between patients with autism spectrum disorder and healthy individuals across the lifespan: results from the enigma asd working group. *American Journal of Psychiatry*, 175(4):359–369, 2018.

[84] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[85] Ulrike Von Luxburg and Shai Ben-David. Towards a statistical theory of clustering. In *Pascal workshop on statistics and optimization of clustering*, pages 20–26. Citeseer, 2005.

[86] Centraal Bureau voor de Statistiek. Ervaren gezondheid, zorggebruik en leefstijl bij kinderen tot 12 jaar. Retrieved January 03, 2022, from https://opendata.cbs.nl/statline/#/CBS/nl/dataset/83716NED/table?dl=60829.

[87] Dorothea Wagner and Frank Wagner. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer, 1993.

[88] Kiri Wagstaff. Refining inductive bias in unsupervised learning via constraints. In *AAAI/IAAI*, page 1112, 2000.

[89] Dandan Wang, Lu Han, Caixi Xi, Yi Xu, Jianbo Lai, Shaojia Lu, Manli Huang, Jianbo Hu, Ning Wei, Weijuan Xu, et al. Interactive effects of gender and sexual orientation on cortical thickness, surface area and gray matter volume: a structural brain mri study. *Quantitative imaging in medicine and surgery*, 10(4):835, 2020.

[90] Fei Wang, Hector-Hugo Franco-Penya, John D Kelleher, John Pugh, and Robert Ross. An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 291–305. Springer, 2017.

[91] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[92] Thomas Wolfers, Christian F Beckmann, Martine Hoogman, Jan K Buitelaar, Barbara Franke, and Andre F Marquand. Individual differences v. the average patient: mapping the heterogeneity in adhd using normative models. *Psychological Medicine*, 50(2):314–323, 2020.

[93] Thomas Wolfers, Jan K Buitelaar, Christian F Beckmann, Barbara Franke, and Andre F Marquand. From estimating activation locality to predicting disorder: a review of pattern recognition for neuroimaging-based psychiatric diagnostics. *Neuroscience & Biobehavioral Reviews*, 57:328–349, 2015.

[94] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[95] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

[96] Jinglin Xu, Junwei Han, Kai Xiong, and Feiping Nie. Robust and sparse fuzzy k-means clustering. In *IJCAI*, pages 2224–2230, 2016.

[97] Mariam Zabihi, Dorothea L Floris, Seyed Mostafa Kia, Thomas Wolfers, Julian Tillmann, Alberto Llera Arenas, Carolin Moessnang, Tobias Banaschewski, Rosemary Holt, Simon Baron-Cohen, et al. Fractionating autism based on neuroanatomical normative modeling. *Translational psychiatry*, 10(1):1–10, 2020.

[98] Mariam Zabihi, Marianne Oldehinkel, Thomas Wolfers, Vincent Frouin, David Goyard, Eva Loth, Tony Charman, Julian Tillmann, Tobias Banaschewski, Guillaume Dumas, et al. Dissecting the heterogeneous cortical anatomy of autism spectrum disorder using normative models. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, 4(6):567–578, 2019.

[99] Chenyi Zuo, Daoyang Wang, Fuxiang Tao, and Yanpei Wang. Changes in the development of subcortical structures in autism spectrum disorder. *NeuroReport*, 30(16):1062–1067, 2019.

A Mathematical prerequisites

A.1 Linear algebra

A.1.1 Vector space

A vector space over $\mathbb{R}$ is a non-empty set V equipped with an addition and a (scalar) multiplication operation, defined

$$\begin{aligned} + : V \times V &\to V \\ (\mathbf{x}, \mathbf{y}) &\mapsto \mathbf{x} + \mathbf{y} \end{aligned}$$

and

$$\begin{aligned} \cdot : \mathbb{R} \times V &\to V \\ (\lambda, \mathbf{y}) &\mapsto \lambda \cdot \mathbf{y} = \lambda \mathbf{y}, \end{aligned}$$

such that the following conditions hold:

- For all $\mathbf{x}, \mathbf{y} \in V : \mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- For all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V : (\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- For all $\mathbf{x}, \mathbf{y} \in V$ there exists a unique $\mathbf{z} \in V$ such that $\mathbf{x} + \mathbf{z} = \mathbf{y}$
- For all $\mathbf{x} \in V$ and $\lambda, \lambda' \in \mathbb{R}$: $(\lambda\lambda')\mathbf{x} = \lambda(\lambda'\mathbf{x})$
- For all $\mathbf{x} \in V$ and $\lambda, \lambda' \in \mathbb{R}$: $(\lambda + \lambda')\mathbf{x} = \lambda\mathbf{x} + \lambda'\mathbf{x}$
- For all $\mathbf{x}, \mathbf{y} \in V$ and $\lambda \in \mathbb{R}$: $\lambda(\mathbf{x} + \mathbf{y}) = \lambda\mathbf{x} + \lambda\mathbf{y}$
- For all $\mathbf{x} \in V$: $1\mathbf{x} = \mathbf{x}$.

Elements of the vector space V are called vectors. In this thesis, we will only use Euclidean vector spaces $\mathbb{R}^d$ with $d > 1$ with commonly defined dot product and norms. That is, for $\mathbf{x} := [x_1, \cdots, x_d]^T, \mathbf{y} := [y_1, \ldots, y_d]^T \in \mathbb{R}^d$, we denote the dot product between $\mathbf{x}$ and $\mathbf{y}$ by

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_d \end{bmatrix} = \sum_{i=1}^{d} x_i y_i,$$

and the (Euclidean) norm of $\mathbf{x}$ by

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}.$$

Lemma A.1.

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 - 2\mathbf{x}^T\mathbf{y} + \|\mathbf{y}\|^2 .$$

Proof. We have

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|^2 &= (\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y}) \\ &= \sum_{i=1}^{d} (x_i - y_i)^2 \\ &= \sum_{i=1}^{d} (x_i)^2 + \sum_{i=1}^{d} (y_i)^2 - 2\sum_{i=1}^{d} x_i y_i \\ &= \|\mathbf{x}\|^2 - 2\mathbf{x}^T\mathbf{y} + \|\mathbf{y}\|^2 . \end{aligned}$$

□

Two vectors $\mathbf{x}$ and $\mathbf{y}$ are called orthogonal if $\mathbf{x}^T\mathbf{y} = 0$, and orthonormal if they are orthogonal and additionally $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$.

A linear combination of vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ is a vector $\mathbf{x} \in \mathbb{R}^d$ such that

$$\mathbf{x} = \lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n$$

for some $\lambda_1, \dots, \lambda_n \in \mathbb{R}$. Vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ are said to be linearly independent if the linear combination such that

$$\lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n = \mathbf{0}^d$$

is achieved only with $\lambda_1 = \cdots = \lambda_n = 0$. The span of vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$, denoted $\text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ is the set of all linear combinations of $\mathbf{x}_1, \dots, \mathbf{x}_n$.

Lemma A.2. *If $\mathbf{x}_1, \dots, \mathbf{x}_n$ are orthonormal, they are linearly independent.*

Proof. Let $\lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n = \mathbf{0}^d$, and consider that for any $1 \le j \le n$, we have

$$\begin{aligned}\mathbf{x}_j^T \cdot (\lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n) &= (\lambda_1\mathbf{x}_j^T\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_j^T\mathbf{x}_n) \\ &= \lambda_j\mathbf{x}_j^T\mathbf{x}_j.\end{aligned}$$

Since $\mathbf{x}_j^T\mathbf{x}_j = \mathbf{0}^d$ implies $\mathbf{x}_j^T$ (which contradicts orthonormality) it holds that $\lambda_j = 0$. Hence, $\lambda_1 = \cdots = \lambda_n = 0$. □

A subspace $W \subset V$ is a subset of V such that

- $\mathbf{0} \in W$
- For all $\mathbf{x}, \mathbf{y} \in W$: $\mathbf{x} + \mathbf{y} \in W$, and
- For all $\mathbf{x} \in W$, $\lambda \in \mathbb{R}$: $\lambda\mathbf{x} \in W$.

Given two subspaces $W, W' \subset V$, the sets $W \cap W'$ and $W + W' := \{\mathbf{w} + \mathbf{w}' \mid \mathbf{w} \in W, \mathbf{w}' \in W'\}$ are also subspaces of V. The dimension of a subspace W – denoted $\dim(W)$ – is the largest number k such that there exist k linearly independent vectors in W.

Lemma A.3. *Let $\mathbf{x}_1, \dots, \mathbf{x}_n$ be orthonormal. Then it holds that $dim(span(\mathbf{x}_1, \dots, \mathbf{x}_n)) = n$.*

Proof. By lemma A.2 we know $\dim(\text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)) \ge n$. By definition, all elements $\mathbf{x}$ of $\text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ are linear combinations of $\mathbf{x}_1, \dots, \mathbf{x}_n$, and thus any $\mathbf{x} \in \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ can be written as $\mathbf{x} = \lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n$, or $\mathbf{0}^d = \lambda_1\mathbf{x}_1 + \cdots + \lambda_n\mathbf{x}_n - \mathbf{x}$. Since at least the last scalar is non-zero, $\mathbf{x}$ is not linearly independent. Hence, $\dim(\text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)) = n$. □

Lemma A.4. *For two subspaces $W, W' \subseteq V$ of a vector space V, we have $dim(W) + dim(W') = dim(W + W')$ if $W \cap W' = \{\mathbf{0}^d\}$.*

Proof. Let $\dim(W) := r$ and $\dim(W') := s$, and let $\mathbf{u}_1, \dots, \mathbf{u}_r \in W$, $\mathbf{v}_1, \dots, \mathbf{v}_s \in W'$ be two linearly independent sets of vectors. We will show that $\mathbf{u}_1, \dots, \mathbf{u}_r, \mathbf{v}_1, \dots, \mathbf{v}_s$ are all linear independent from each other. To do so, let

$$\mathbf{0}^d = \sum_{i=1}^{r} \lambda_i\mathbf{u}_i + \sum_{j=1}^{s} \lambda'_j\mathbf{v}_j,$$

or

$$\sum_{i=1}^{r} \lambda_i\mathbf{u}_i = -\sum_{j=1}^{s} \lambda'_j\mathbf{v}_j.$$

Let $\mathbf{x} := \sum_{i=1}^{r} \lambda_i\mathbf{u}_i = -\sum_{j=1}^{s} \lambda'_j\mathbf{v}_j$. By definition of subspaces, $\mathbf{x} \in W$ and also $\mathbf{x} \in W'$, so $\mathbf{x} \in W \cap W'$. Since $W \cap W' = \{\mathbf{0}^d\}$ however, we know, $\sum_{i=1}^{r} \lambda_i\mathbf{u}_i = -\sum_{j=1}^{s} \lambda'_j\mathbf{v}_j = \mathbf{0}^d$. Since $\mathbf{u}_1, \dots, \mathbf{u}_r$ and $\mathbf{v}_1, \dots, \mathbf{v}_s$ are linearly independent, this means $\lambda_1 = \cdots = \lambda_r = \lambda'_1 = \cdots = \lambda'_s = 0$, which in turn means $\mathbf{u}_1, \dots, \mathbf{u}_r, \mathbf{v}_1, \dots, \mathbf{v}_s$ are linearly independent.

Now let $\mathbf{x} \in W + W'$, such that it can be written as $\mathbf{x} = \sum_{i=1}^{r} \lambda_i\mathbf{u}_i + \sum_{j=1}^{s} \lambda'_j\mathbf{v}_j$. By the same argument as lemma A.3 we see $\mathbf{x}$ is not linearly independent from $\mathbf{u}_1, \dots, \mathbf{u}_r, \mathbf{v}_1, \dots, \mathbf{v}_s$. Since $\mathbf{x}$ was chosen arbitrarily, there

exist at most $r+s$ linearly independent vectors in $W+W'$, i.e. $\dim(W+W') = r+s = \dim(W)+\dim(W')$, which is what we wanted to show. □

Lemma A.5. *$dim(\mathbb{R}^d) = d$.*

Proof. Clearly, vectors $[1,0\cdots0]^T, [0,1\cdots0]^T, \ldots, [0,0\cdots1]^T \in \mathbb{R}^d$ are linearly independent. Therefore, $\dim(\mathbb{R}^d) \geq d$. Now let $\mathbf{x} = [x_1 \cdots x_d]^T$ be some other element of $\mathbb{R}^d$. Then, $\mathbf{x}$ can be written as

$$\mathbf{x} = x_1 \cdot [10\cdots0]^T + \cdots + x_d \cdot [00\cdots1]^T,$$

which means $\mathbf{x}$ is not linearly independent from $[10\cdots0]^T, [01\cdots0]^T, \ldots, [00\cdots1]^T$. Since $\mathbf{x}$ was chosen arbitrarily, $\dim(\mathbb{R}^d) = d$. □

A.1.2 Matrices

A matrix is a rectangular array of objects – which, in the context of this thesis, are all real numbers – called entries. A matrix with n rows and k columns is a $n \times k$ matrix, and the set of all such matrices is denoted $\mathbb{R}^{n\times k}$ (n and k are also called the dimensions). The entry in the i-th row and j-th column of a matrix $\mathbf{M} \in \mathbb{R}^{n\times k}$ is denoted $(\mathbf{M})_{ij}$, such that the matrix can be portrayed visually as

$$\mathbf{M} = \begin{bmatrix} (\mathbf{M})_{11} & (\mathbf{M})12 & \ldots & (\mathbf{M})1k \\ (\mathbf{M})21 & (\mathbf{M})22 & \ldots & (\mathbf{M})2k \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{M})n1 & (\mathbf{M})n2 & \ldots & (\mathbf{M})nk \end{bmatrix}.$$

A d-dimensional (row) vector is also a $1 \times d$-dimensional matrix. There are a couple basic operations defined on matrices: addition, (scalar) multiplication and transposition. Addition and scalar multiplication function entrywise, i.e. for $\mathbf{M}, \mathbf{M}' \in \mathbb{R}^{n\times k}$, $\lambda \in \mathbb{R}$:

$$(\mathbf{M}+\mathbf{M}')_{ij} = (\mathbf{M})_{ij} + (\mathbf{M}')_{ij},$$
$$(\lambda\mathbf{M})_{ij} = \lambda(\mathbf{M})_{ij}.$$

Switching rows and columns, we get the transpose of $\mathbf{M}$, denoted $\mathbf{M}^T$, defined as

$$(\mathbf{M}^T)_{ij} = (\mathbf{M})_{ji}.$$

Multiplication of two matrices is defined for matrices $\mathbf{M} \in \mathbb{R}^{n\times k}$ and $\mathbf{M}' \in \mathbb{R}^{k\times n'}$ – such that the number of columns in $\mathbf{M}$ equals the number of rows in $\mathbf{M}'$ – and returns a matrix $\mathbf{M}\mathbf{M}' \in \mathbb{R}^{n\times n'}$ such that

$$(\mathbf{M}\mathbf{M}')_{ij} = \sum_{r=1}^{k} (\mathbf{M})_{ir}(\mathbf{M}')_{rj} = \mathbf{m}_i^T \mathbf{m}'_j,$$

where $\mathbf{m}_i$ and $\mathbf{m}'_j$ refer to the i-th row of $\mathbf{M}$ and j-th column of $\mathbf{M}'$ respectively. A few properties using and connecting the operations include:

- $\mathbf{M}(\mathbf{M}'+\mathbf{M}'') = \mathbf{M}\mathbf{M}' + \mathbf{M}\mathbf{M}''$
- $\mathbf{M}(\mathbf{M}'\mathbf{M}'') = (\mathbf{M}\mathbf{M}')\mathbf{M}''$
- $(\mathbf{M}^T)^T = \mathbf{M}$
- $(\mathbf{M}+\mathbf{M}')^T = \mathbf{M}^T + \mathbf{M}'^T$

If $\mathbf{M}^T = \mathbf{M}$ we call $\mathbf{M}$ symmetric. A matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ is called orthogonal if the columns of $\mathbf{M}$ form a set of orthonormal vectors, such that $\mathbf{M}^T\mathbf{M} = \mathbf{1}^{n\times n}$. Here, $\mathbf{1}^{n\times n}$ is the identity matrix defined as $(\mathbf{1})_{ij} = \delta_{ij}$ where δ_{ij} is the Kronecker delta. The $n \times k$ matrix with only zero entries is denoted $\mathbf{0}^{n\times k}$.

Multiplication of a matrix with a vectors induces another important notion we will use: eigenvectors. For any square matrix $\mathbf{M} \in mathbbR^{n\times n}$, $\mathbf{x} \in mathbbR^n$, we call $\mathbf{x}$ an eigenvector of $\mathbf{M}$ if

$$\mathbf{M}\mathbf{x} = \lambda\mathbf{x}$$

for some $\lambda \in \mathbb{R}$, and call λ a eigenvalue of $\mathbf{M}$.

Lemma A.6. *If* $\mathbf{v}$ *is an eigenvector of* $\mathbf{M}$ *with eigenvalue* λ *than* $\alpha\mathbf{v}$ *is also an eigenvector* $\mathbf{M}$ *with eigenvalue* λ *for any* $0 \neq \alpha \in \mathbb{R}$.

Proof. Let λ be the eigenvalue corresponding to $\mathbf{v}$. Then, we have

$$\mathbf{M}(\alpha\mathbf{v} = \alpha(\mathbf{M}\mathbf{v}) = \alpha\lambda\mathbf{v} = \lambda(\alpha\mathbf{v}).$$

□

Lemma A.6 tells us it is silly to speak of *the* eigenvectors of $\mathbf{M}$, when in fact there are an infinite amount of them corresponding to every distinct eigenvalue. In particular, for every λ we can construct an eigenvector of unit length by picking some eigenvector $\mathbf{v}$ and determining

$$\frac{1}{\|\mathbf{v}\|}\mathbf{v}.$$

Lemma A.7. *Let* $\mathbf{m}$ *be symmetric. Then, any two eigenvectors of* $\mathbf{M}$ *with distinct eigenvectors are orthogonal.*

Proof. Let $\mathbf{x}, \mathbf{y}$ be distinct eigenvectors with eigenvalues λ and λ', $\lambda \neq \lambda'$. Then by definition the following holds:

$$\mathbf{M}\mathbf{x} = \lambda\mathbf{x}$$
$$\mathbf{M}\mathbf{y} = \lambda'\mathbf{y}.$$

It now follows that

$$\begin{aligned}\lambda(\mathbf{x}^T\mathbf{y}) &= (\lambda\mathbf{x})^T\mathbf{y} \\ &= (\mathbf{M}\mathbf{x})^T\mathbf{y} \\ &= \mathbf{x}^T\mathbf{M}^T\mathbf{y} \\ &= \mathbf{x}^T\mathbf{M}\mathbf{y} \\ &= \mathbf{x}^T\lambda'\mathbf{y} \\ &= \lambda'(\mathbf{x}^T\mathbf{y}),\end{aligned}$$

meaning $0 = (\lambda - \lambda')(\mathbf{x}^T\mathbf{y})$. Since $\lambda \neq \lambda'$, we have $\mathbf{x}^T\mathbf{y} = 0$, which means $\mathbf{x}$ and $\mathbf{y}$ are orthogonal. □

A $n \times n$ symmetric matrix $\mathbf{M}$ is called positive semi-definite if and only if $\mathbf{x}^T\mathbf{M}\mathbf{x} \geq \mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^n$.

Lemma A.8. *If* $\mathbf{M}$ *is positive semi-definite, it has only non-negative eigenvalues.*

Proof. Let λ be some eigenvalue of $\mathbf{M}$ and let $\mathbf{v}$ be some corresponding eigenvector. Then it holds that $0 \leq \mathbf{v}^T\mathbf{M}\mathbf{v} = \mathbf{v}^T(\lambda\mathbf{v}) = (\mathbf{v}^T\mathbf{v})\lambda$. Since $\mathbf{v}^T\mathbf{v} \geq 0$ (easily checked) it must be the case that $\lambda \geq 0$. □

Combining lemma A.6, A.7 and A.8, we see that for any positive semi-definite matrix $\mathbf{M}$, we know $\mathbf{M}$ has eigenvalues $0 \leq \lambda_1 \leq \cdots \leq \lambda_k$ (for some $k > 0$), such that there exists an orthonormal set of eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ corresponding to those eigenvalues.

For any eigenvalue λ, we define its eigenspace $E_{\mathbf{M}}(\lambda)$ as the set of all eigenvectors corresponding to this eigenvalue, i.e. for some $\mathbf{M} \in mathbbR^{n\times n}$ it is defined as

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{M}\mathbf{x} = \lambda\mathbf{x}\}.$$

Lemma A.9. *For any* $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y} \in \mathbb{R}^d$, *with* $\mathbf{x}_i := \left[x_i^1, \ldots, x_i^d\right]^T$ *for* $1 \leq i \leq n$, *we have*

$$\sum_{i=1}^{n} \mathbf{x}_i^T\mathbf{y} = (\sum_{i=1}^{n} \mathbf{x}_i)^T\mathbf{y}$$

Proof.

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{y} &= (\sum_{i=1}^{n} \mathbf{x}_i^T) \cdot \mathbf{y} \\ &= (\sum_{i=1}^{n} \mathbf{x}_i)^T \cdot \mathbf{y}. && \text{(since } (\mathbf{M}+\mathbf{M}')^T = \mathbf{M}^T + \mathbf{M}'^T\text{)}\end{aligned}$$

□

Lemma A.10. *Let* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$. *Then* $\lambda \neq 0$ *is an eigenvalue of* **AB** *iff* λ *is an eigenvalue of* **BA**.

Proof. Let $\mathbf{u}$ be an eigenvector of **AB** associated with eigenvalue λ, i.e. $\mathbf{ABu} = \lambda\mathbf{u}$. Let $\mathbf{v} := \mathbf{Bu}$. Then,

$$\begin{aligned}\mathbf{BAv} &= \mathbf{BABu} \\ &= \mathbf{B}(\mathbf{ABu}) \\ &= \mathbf{B}\lambda\mathbf{u} \\ &= \lambda(\mathbf{Bu}) \\ &= \lambda\mathbf{v}.\end{aligned}$$

Hence, λ is an eigenvalue of BA too. The other implication follows analogously, mutatis mutandis. □

Lastly, the trace of a square matrix $\mathbf{M}$, denoted $\mathrm{Tr}(\mathbf{M})$ is the sum of its diagonal elements, i.e.

$$\mathrm{Tr}(\mathbf{M}) = \sum_{i=1}^{n} (\mathbf{M})_{ii}.$$

It can be shown that the trace is equal to the sum of the eigenvalues of $\mathbf{M}$.

A.2 Relations

A (binary) relation over sets A and B is a set of ordered pairs $R := \{(a,b) \mid a \in A, b \in B\} \subseteq A \times B$. If $A = B$, we say R is homogeneous. For any homogeneous relation R, we say R is

- reflexive iff $(a,a) \in R$ for all $a \in A$,
- symmetric iff $(a,a') \in R$ entails $(a',a) \in R$ for all $a, a' \in A$,
- transitive iff $\{(a,a'),(a',a'')\} \subseteq R$ entails $(a,a'') \in R$ for all $a, a', a'' \in A$, and
- an equivalence relation iff it is reflexive, symmetric, transitive.

Lemma A.11. *Let* R_A *be the set of all homogeneous relations over a set* A. *There exists a bijection* $f : R_A \to \Pi(A)$.

Proof. Let $R \in R_A$, and for any $a \in A$, let $[a] := \{b \mid (a,b) \in R\}$. We define $f(R) := \{[a] \mid a \in A\}$. To show that $f(R)$ is a partition of A, we will show the conditions given in section 2.1 hold:

1. let $a \in A$. Since R is reflexive, we have $a \in [a]$, whence $a \in \bigcup_{a\in A}[a]$. Since a was chosen arbitrarily, we have $\bigcup_{a\in A}[a] = A$.
2. let $[a],[a'] \in f(R)$ with $[a] \neq [a']$. Suppose now that $[a]\cap[a'] \neq \emptyset$, such that there exists $x \in [a]\cap[a']$. Then for any $y \in [a]$ and $z \in [a']$, we have $(y,x),(x,z) \in R$, whence $(y,z) \in R$ since R is transitive. This however means that $x \in [a']$ and $z \in [a]$, whence $[a] = [a']$, which is a contradiction. Hence, $[a]\cap[a'] = \emptyset$
3. Any $[a] \in f(R)$ contains at least one element, namely a. So $\emptyset \notin f(R)$.

Conversely, we let $P := \{P_1, \ldots, P_n\} \in \Pi(A)$ and define $f^{-1}(P) := \{(a,a') \mid a, a' \in P_i \text{ for some } 1 \leq i \leq n\}$. We will show that $f^{-1}(P)$ is an equivalence relation.

1. Let $a \in A$. If $a \in P_i$, then $a \in P_i$ holds vacuously. So $(a,a) \in f^{-1}(P)$, hence it is reflexive.
2. If $a, a' \in A$ then $a', a \in A$. So if $(a,a') \in f^{-1}(P)$ then $(a',a) \in f^{-1}(P)$.

3. If $a, a' \in A$, $a', a'' \in A$, then $a, a'' \in A$. So by the same argument, $f^{-1}(P)$ is transitive.

So we have constructed a natural bijection $f(f^{-1}(P)) = P$ and $f^{-1}(f(R)) = R$, which can be easily checked. □

A.3 Graphs

A graph G is an ordered triple $(V, E, \mathbf{W})$ comprising of a set V called vertices, E is a set of edges which is a homogeneous binary relation over V, and $\mathbf{W} := (w)_{ij} \in \mathbb{R}^{|V|\times|V|}$ is a weight matrix where for $1 \leq i, j \leq |V|$ such that $(i, j) \in E$, w_{ij} is the called weight of edge (i, j), and $w_{ij} = 0$ if $(i, j) \notin E$. If E is symmetric, we call G undirected.

For $v, v' \in V$, we say v to v' are connected if there exists a finite sequence of distinct edges $(e_1 \dots e_{n-1})$ and a sequence of vertices $(v_1 \dots v_n)$ such that $v_1 = v$, $v_n = v'$, $e_i = (v_i, v_{i+1})$, and $v_i \neq v_j$ and $e_i \neq e_j$ if $i \neq j$. A graph is called connected if all vertices are connected to all other vertices in the graph. If any v is connected to itself, the sequence corresponding to this connection $(v_1 \dots v_n)$ is called a cycle.

A spanning tree of a connected graph $G = (V, E, \mathbf{W})$ is a subset $E' \subseteq E$ with weight matrix $\mathbf{W}'$ such that the graph $G' := (V, E', \mathbf{W}')$ is still connected, and has no cycles. A minimum spanning tree is a spanning tree such that the total sum of all edge weights is minimized.

For $A, B \subseteq V$, any $(a, b) \in E$ with $a \in A, b \in B$ is called a crossing edge. A connected component of a graph is a subset of $A \subseteq V$ such that all vertices in A are connected to each other, and there exist no crossing edges from A to $V - A$. If we define a relation σ over V such that $(v, v') \in \sigma$ are connected, we can easily check that σ is an equivalence relation. Hence, in accordance with lemma A.11, the set of connected components of G form a partition of V.

B Gaussian Process Regression

We'd like to learn a function $f : X \to Y$ given domain $X(= \mathbb{R}^m)$ and range $Y(= \mathbb{R}^{m'})$, that is trained using training set D_{NT} and tested on the set D_{ASD}. The method we will use for this is called Gaussian Process Regression, which we will briefly describe here.

Consider some probability density function p_f of a random variable f over some function space F, where each $f \in F$ is defined such that $f : X \to Y$, for said X and Y. When we generate samples from F, and evaluate every sample f at a fixed point $\mathbf{x} \in X$, we will find that $f(\mathbf{x})$ is a random m'-dimensional vector with some distribution fixed by p_f. Similarly, we can sample from F at n fixed points to yield an $m' \times n$ matrix $\mathbf{F} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]$, of which the joint probability density function $p_{\mathbf{F}}$ could be found. Now, assume that this distribution of $\mathbf{F}$ is a multivariate Gaussian for all finite n and all $\mathbf{x}_i \in X$, $1 \leq i \leq n$. That is, the joint distribution of any finite set of fixed test points is a multivariate Gaussian. Such a probability distribution is called a Gaussian Process. Given our dataset D_{NT} with input $\mathbf{x_i}$, $1 \leq i \leq n$, we model the corresponding response variables $\mathbf{y_i}$ assuming some additive, i.i.d. Gaussian noise, i.e. $\mathbf{y_i} = f(\mathbf{x_i}) + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Thus, we are in fact considering n random vectors $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$, which we collect into a matrix $\mathbf{F} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]$. When performing Gaussian Process Regression, we place a multivariate Gaussian prior distribution over $\mathbf{F}$, so

$$\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(m(\mathbf{X}), \mathbf{K}(\mathbf{X}, \mathbf{X})),$$

where $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, $m(\mathbf{X}) := [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)]$ is the mean matrix with $m(\mathbf{x}_i)$ being the mean of $f(\mathbf{x}_i)$, and $\mathbf{K}(\mathbf{X}, \mathbf{X})$) is a $n \times n$ covariance matrix dependent on $\mathbf{X}$. In particular, for $1 \leq i, j \leq n$, the element in the i-th row and j-th column of $\mathbf{K}$ equals $k(\mathbf{x}_i, \mathbf{x}_j)$, where $k : X \times X \to \mathbb{R}_{>0}$ is a positive definite covariance function, also called a kernel. This kernel function has to be symmetric, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$. Often, it is assumed that the mean matrix for $m(\mathbf{X}) = 0$, such that the distribution can be written as

$$\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(0, \mathbf{K}(\mathbf{X}, \mathbf{X})).$$

The shape of this distribution is decided entirely by the choice for the kernel.[18] Now given this distribution of $\mathbf{F}$ conditioned on $\mathbf{X}$, we would like to make predictions for new points $\mathbf{x}'_j$, $1 \leq j \leq n'$, in our case coming from the dataset D_{ASD}. If we let $\mathbf{X}' := \left[\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'}\right]$, we obtain matrix $\mathbf{F}' := \left[f(\mathbf{x}'_1), \ldots, f(\mathbf{x}'_{n'})\right]$ similarly as before. Now, the joint distribution of $\mathbf{F}$ and $\mathbf{F}'$ is again a multivariate Gaussian distribution, and is found by augmenting our previous distribution with $\mathbf{F}'$ to get

$$\begin{bmatrix} \mathbf{F} \\ \mathbf{F}' \end{bmatrix} \mid \mathbf{X} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}') \\ \mathbf{K}(\mathbf{X}', \mathbf{X}) & \mathbf{K}(\mathbf{X}', \mathbf{X}') \end{bmatrix}\right),$$

where $\mathbf{K}(\mathbf{X}, \mathbf{X}')$, $\mathbf{K}(\mathbf{X}', \mathbf{X})$, and $\mathbf{K}(\mathbf{X}', \mathbf{X}')$ are the $n \times n'$, $n' \times n$ and $n' \times n'$ matrix with covariances between $\mathbf{X}$ and $\mathbf{X}'$,$\mathbf{X}'$ and $\mathbf{X}$ and $\mathbf{X}'$ and $\mathbf{X}'$ respectively.

Let $\mathbf{Y} = [\mathbf{y_1}, \ldots, \mathbf{y_n}]^T$. Then, based on our assumption that $\mathbf{y_i} = f(\mathbf{x_i}) + \varepsilon$, it is shown in [91] that we can rewrite the distribution to get

$$\begin{bmatrix} \mathbf{Y} \\ \mathbf{F}' \end{bmatrix} \mid \mathbf{X} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{1}^{n \times n} & \mathbf{K}(\mathbf{X}, \mathbf{X}') \\ \mathbf{K}(\mathbf{X}', \mathbf{X}) & \mathbf{K}(\mathbf{X}', \mathbf{X}') \end{bmatrix}\right),$$

which is now a joint distribution over the observed target values $\mathbf{Y}$ (in our case, the brain volumes measured in D_{NT}) and the values of $\mathbf{F}'$ that we would like to predict. For brevity, let $\mathbf{K} := \mathbf{K}(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* := \mathbf{K}(\mathbf{X}, \mathbf{X}')$ and $\mathbf{K}_{**} := \mathbf{K}(\mathbf{X}', \mathbf{X}')$. Since k is a symmetric function, we have that $\mathbf{K}_*^T = \mathbf{K}(\mathbf{X}', \mathbf{X})$. Using standard Gaussian equations (see [91]), we can now derive the conditional distribution

$$\mathbf{F}' \mid \mathbf{Y}, \mathbf{X} \sim \mathcal{N}\left(m(\mathbf{X}'), \mathrm{v}(\mathbf{X}')\right),$$

where

$$m(\mathbf{X}') = \mathbf{K}_*^{\mathbf{T}} \left(\mathbf{K} + \sigma^2 \mathbf{1}^{n \times n}\right)^{-1} \mathbf{Y},$$

and

$$\mathrm{v}(\mathbf{X}') = \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}} (\mathbf{K} + \sigma^2 \mathbf{1}^{n \times n})^{-1} \mathbf{K}_*.$$

[18] Which kernels are chosen in our case is discussed in section 5.2.2.

Note that v($\mathbf{X}'$) only depends on $\mathbf{X}$ and $\mathbf{X}'$, and not on $\mathbf{y}$. If we only consider one testpoint $\mathbf{x}'$ in $\mathbf{X}'$, we can simplify these equations to get

$$m(\mathbf{x}') = \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{1}^{n\times n})^{-1}\mathbf{Y},$$
$$v(\mathbf{x}') = k(\mathbf{x}',\mathbf{x}') - \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{1}^{n\times n})^{-1}\mathbf{k}_*,$$

where $\mathbf{k}_*$ refers to the vector with covariance values between $\mathbf{x}'$ and all training points $\mathbf{X}$. From this distribution, we can yield an algorithm to calculate a predictive mean and variance for all elements of $\mathbf{X}'$:

Algorithm 9 Prediction using GPR (adapted from [91])

1: **input:** $\mathbf{X}, \mathbf{Y}$ (training input/output from D_{NT}), $\mathbf{X}'$ (test input from D_{ASD}), k (covariance function), σ^2 (noise level).
2: **for** all $\mathbf{x}' \in \mathbf{X}'$:
3: **calculate** $m(\mathbf{x}')$:
4: $\mathbf{L} := \text{cholesky}(\mathbf{K} + \sigma^2\mathbf{1}^{n\times n})$
5: $\mathbf{A} = \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{Y})$
6: $m(\mathbf{x}') = \mathbf{k}_*^\mathbf{T} \cdot \mathbf{A}$.
7: **calculate** v($\mathbf{x}'$):
8: $\mathbf{b} = \mathbf{L} \setminus \mathbf{k}_*$
9: v($\mathbf{x}'$) $= k(\mathbf{x}',\mathbf{x}') - \mathbf{b}^T\mathbf{b}$.
10: **return** $m(\mathbf{x}')$, v($\mathbf{x}'$) for all $\mathbf{x}' \in \mathbf{X}'$.

Line 3 refers to the Cholesky decomposition of $(\mathbf{K} + \sigma^2\mathbf{1}^{n\times n})$, which takes the form of $\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a real lower triangular matrix with positive entries on the diagonal. This decomposition exists since $(\mathbf{K}+\sigma^2\mathbf{1}^{n\times n})$ is symmetric and positive-definite (which follows from the definition of k) [42]. The matrix $\mathbf{L}$ has the property that in order to find an $\mathbf{A}$ that solves the linear system $(\mathbf{K}+\sigma^2\mathbf{1}^{n\times n})\mathbf{A} = \mathbf{Y}$, we need to calculate $\mathbf{L}^T\setminus(\mathbf{L}\setminus\mathbf{Y})$, where $\setminus$ refers to the left matrix division operation. Once we have found $\mathbf{A}$, we see that $\mathbf{A} = (\mathbf{K}+\sigma^2\mathbf{1}^{n\times n})^{-1}\mathbf{Y}$, whence multiplying with ${\mathbf{K}_*}^T$ yields $m(\mathbf{X}')$ again. Similarly, we can check that line 7 and 8 correctly yield our formula for v($\mathbf{X}'$). Alternatively, standard matrix inversion algorithms can be used, but Cholesky decomposition is often preferred for symmetric positive-definite matrices due to its computational benefits [80].

C Effect of `n_neighbors` on mean silhouette score

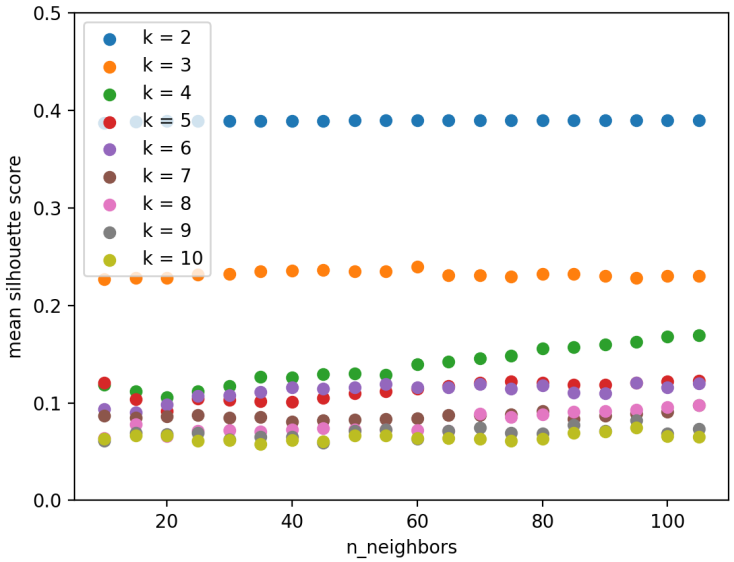


Figure C.1: Effect of the `n_neighbors` parameter on the mean silhouette score for different values of k using cosine affinity matrix.

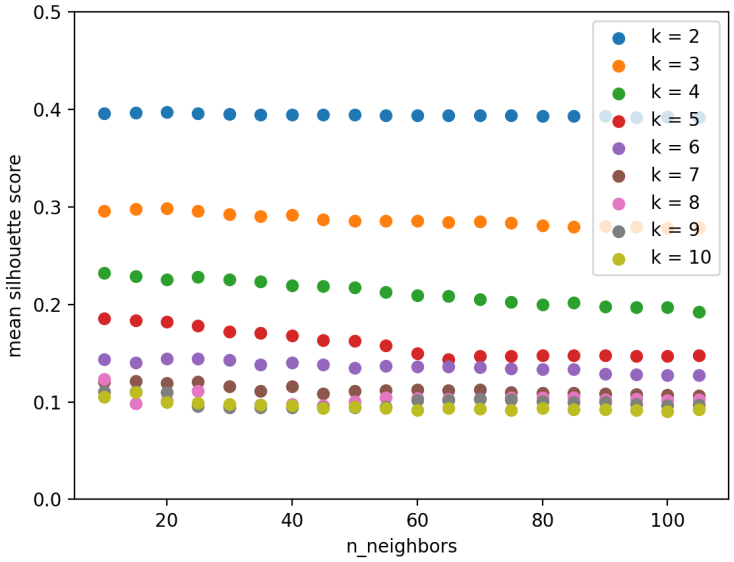


Figure C.2: Effect of the `n_neighbors` parameter on the mean silhouette score for different values of k using squared euclidean affinity matrix.

D TOP rankings for all clusters

These are the TOP rankings of all resulting clusters discussed in 6. The 'Measure' column refers to the brain regions according to the Desikan-Killiany atlas, an overview of which can be found in [25]. The addition '(SA)' means this is a (cortical) measure of surface area. The 'Rank' column refers to the total number of appearances of that measure in the TOP ranking of individual subjects (see section 5.5). The 'Lobe' columns refers to the lobe that measure (approximately) belongs to.

Measure	Rank	Lobe
middletemporal (SA)	92	temporal
lateralorbitofrontal (SA)	84	frontal
inferiortemporal (SA)	80	temporal
bankssts (SA)	79	parietal
fusiform (SA)	77	frontal
superiortemporal (SA)	76	temporal
superiorfrontal (SA)	76	frontal
medialorbitofrontal (SA)	74	frontal
posteriorcingulate (SA)	72	frontal/cingulate
rostralmiddlefrontal (SA)	68	frontal
rostralanteriorcingulate (SA)	67	frontal/cingulate
inferiorparietal (SA)	66	parietal
supramarginal (SA)	63	parietal
parsorbitalis (SA)	63	frontal
parstriangularis (SA)	63	frontal

Figure D.1: TOP ranking of cluster 1 of spectral clustering using squared Euclidean affinity.

Measure	Rank	Lobe
insula (SA)	177	insula
superiorfrontal (SA)	158	frontal
lateraloccipital (SA)	155	occipital
postcentral (SA)	149	parietal/motor
precentral (SA)	148	frontal/motor
precuneus (SA)	147	occipital
rostralmiddlefrontal (SA)	145	frontal
medialorbitofrontal (SA)	144	frontal
rostralanteriorcingulate (SA)	143	frontal/cingulate
parsorbitalis (SA)	142	frontal
thalamus	139	*subcortical*
superiortemporal (SA)	138	temporal
superiorparietal (SA)	138	parietal
isthmuscingulate (SA)	137	frontal/cingulate
inferiorparietal (SA)	137	parietal

Figure D.2: TOP ranking of cluster 2 of spectral clustering using squared Euclidean affinity.

Measure	Rank	Lobe
accumbens	307	*subcortical*
pallidus	293	*subcortical*
putamen	287	*subcortical*
frontalpole (SA)	274	frontal
caudate nucleus	271	*subcortical*
lateral ventricles	248	-
pericalcarine (SA)	247	occipital
hippocampus	246	subcortical/temporal
transversetemporal (SA)	244	temporal
temporalpole (SA)	243	temporal
isthmuscingulate (SA)	238	frontal/cingulate
amygdala	234	*subcortical*
parahippocampal (SA)	233	temporal
lingual (SA)	224	occipital
parstriangularis (SA)	223	frontal

Figure D.3: TOP ranking of cluster 3 of spectral clustering using squared Euclidean affinity.

Measure	Rank	Lobe
middletemporal (SA)	227	temporal
lateralorbitofrontal (SA)	226	frontal
putamen	225	*subcortical*
accumbens	220	*subcortical*
inferiortemporal (SA)	214	temporal
posteriorcingulate (SA)	212	frontal/cingulate
parstriangularis (SA)	210	frontal
caudalanteriorcingulate (SA)	206	frontal/cingulate
caudalmiddlefrontal (SA)	204	frontal
fusiforsurfavg	204	temporal
supramarginal (SA)	204	parietal
rostralanteriorcingulate (SA)	203	frontal/cingulate
bankssts (SA)	202	parietal
accumbens	201	*subcortical*
Mcaud	201	*subcortical*

Figure D.4: TOP ranking of cluster 1 of spectral clustering using cosine affinity.

Measure	Rank	Lobe
accumbens	243	*subcortical*
isthmuscingulate (SA)	239	frontal/cingulate
frontalpole (SA)	237	frontal
insula (SA)	236	insula
Mcaud	233	*subcortical*
thalamus	228	*subcortical*
hippocampus	226	*subcortical*
lateraloccipital (SA)	226	occipital
pericalcarine (SA)	221	occipital
posteriorcingulate (SA)	217	frontal/cingulate
pallidus	217	*subcortical*
lingual (SA)	215	occipital
precuneus (SA)	214	occipital
parsorbitalis (SA)	213	frontal
rostralanteriorcingulate (SA)	213	frontal/cingulate

Figure D.5: TOP ranking of cluster 2 of spectral clustering using cosine affinity.

Measure	Rank	Lobe
insula (SA)	186	insula
superiorfrontal (SA)	164	frontal
lateraloccipital (SA)	162	occipital
precuneus (SA)	158	occipital
precentral (SA)	157	motor
rostralmiddlefrontal (SA)	156	frontal
thalamus	156	*subcortical*
postcentral (SA)	156	motor
isthmuscingulate (SA)	155	frontal/cingulate
inferiorparietal (SA)	153	parietal
parsorbitalis (SA)	152	frontal/cingulate
rostralanteriorcingulate (SA)	150	frontal
medialorbitofrontal (SA)	148	frontal
superiortemporal (SA)	145	temporal
cuneus (SA)	144	occipital

Figure D.6: TOP ranking of cluster 1 of HDBSCAN using cosine affinity.

Measure	Rank	Lobe
middletemporal (SA)	179	frontal
inferiortemporal (SA)	152	frontal
lateralorbitofrontal (SA)	151	frontal
posteriorcingulate (SA)	149	frontal/cingulate
fusiform (SA)	145	temporal
bankssts (SA)	145	parietal
superiorfrontal (SA)	144	frontal
rostralanteriorcingulate (SA)	142	frontal/cingulate
superiortemporal (SA)	141	temporal
medialorbitofrontal (SA)	139	frontal
inferiorparietal (SA)	138	parietal
caudalanteriorcingulate (SA)	136	frontal/cingulate
supramarginal (SA)	135	parietal
rostralmiddlefrontal (SA)	134	frontal
putamen	133	*subcortical*

Figure D.7: TOP ranking of cluster 2 of HDBSCAN using cosine affinity.

E Creation of toy dataset

```
import numpy as np
import sklearn.datasets as sets

moons, _ = sets.make_moons(n_samples=50, noise=0.01)
moons2, _ = sets.make_moons(n_samples=50, noise=0.03)
blobs, _ = sets.make_blobs(n_samples=50, centers=[(-0.75,2.25),
(1.25, 2.0)], cluster_std=0.5)
blobs2, _ = sets.make_blobs(n_samples=50, centers=[(-0.75,2.25),
(1.25, 2.0)], cluster_std=0.5)
data = np.vstack([moons,blobs, moons2, blobs2])
```