Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Tangles on Ordinal Data

Alexander Conzelmann

Datum

Reviewers

Prof. Dr. Ulrike von Luxburg
Theory of Machine Learning
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Prof. Felix Wichmann, DPhil
Neural Information Processing
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Conzelmann, Alexander
Tangles on Ordinal Data
Master Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: 04/2022-10/2022

Abstract

We investigate a new algorithm for clustering triplet data (comparison-based data without absolute distance information). This new approach is based on the tangles framework, a tool previously used to finde dense structures in graphs, which we extend to work on triplet data. Current work focusses on embedding triplet data into a euclidean space, resulting possibly in distortions of our data, and then following up with a classical clustering algorithm. In contrast, our proposed method does not construct an intermediate embedding, potentially introducing less distortions on our data and achieving a higher clustering accuracy. Additionally to being competitive in performance, our approach provides both explainability as well as a cluster hierarchy on top with no added cost. We validate the tangles algorithm both on synthetic data under a diverse range of external influences as well as experimental data from the realm of psychophysics.

Zusammenfassung

Bei einer englischen Masterarbeit muss zusätzlich eine deutsche Zusammenfassung verfasst werden.

Acknowledgements

Write here your acknowledgements.

Contents

Chapter 1

Introduction

Is item i closer to item j or item k? is the central question in a triplet setting. As humans have problems ordering most things on absolute scales (what would be values of the features *pop, rock, metal* of Rihanna?), a research community has formed to investigate how to work with comparison or triplet data. Here, humans are only asked to rate the similarity of objects, instead of their absolute values, which is intuitively easier (Rihanna being more similar to Britney Spears than to Metallica seems logical). Human perception is another interesting venue for triplet questions: purple and red might be very far removed in terms of their wavelength, but we actually perceive them as very similar (Shepard, 1962). Insights like these can be uncovered through triplet data with appropriate data analysis methods.

Most of the research community has dealt with ordinal embeddings (Agarwal et al., 2007, Tamuz et al., 2011, L. van der Maaten and K. Weinberger, 2012, Terada and Luxburg, 2014, Jain et al., 2016, Ghosh et al., 2019, Anderton and Aslam, 2019), which are algorithms that try to embed the original data points into a euclidean space from the given triplet comparisons. An ordinal embedding aims to place the points in such a way that they respect as many original triplet comparisons as possible, with regards to the euclidean distance. However, the euclidean distance is a proper metric, so it is symmetrical and has to obey the triangle inequality, limiting its flexibility. This means that the approach is not always perfect: the objects might have complex similarity values to each other that simply cannot be captured by euclidean distances alone.

With an ordinal embedding, the triplet data is brought into a more common format and accessible for further tasks, such as clustering or classification with classical methods (Kleindessner and von Luxburg, 2017). However, this approach seems a bit lacking: if we already know that an ordinal embedding might introduce distortions in our data, why not try to solve the end tasks on the triplet data directly? This approach has shown promising results in

(Kleindessner and von Luxburg, 2017, Kleindessner and von Luxburg, 2017, Ghoshdastidar et al., 2019) for clustering, classification and hierarchy reconstruction.

In this thesis, we want to use the tangles framework on clustering by **?** and extend it to an algorithm that is capable of clustering and reconstructing hierarchies on triplet data. First, in chapter 2, we will give an introduction to tangles and ordinal data aimed at the unfamiliar reader, which lays the necessary foundations for the rest of the thesis. In chapter 3, we will present our two extensions of the tangles framework, which we have named *landmark tangles* and *majority tangles*. We will use simulated data in chapter 4 to show the performance of our algorithms under different environmental factors, such as the noise of the triplets on simulations or how many triplets are available. In chapter 5 we will showcase an example evaluation with real data from the domain of psychophysics, to establish tangles as a practical tool on triplet data. There we will also highlight the inherent explainability of the tangles framework.

In the end, we will have shown that tangles can make a viable alternative for clustering triplet data and will have identified use cases where it performs especially well.

Chapter 2

Theoretical Background

In this chapter we will give a brief introduction about the theoretical concepts that are used in this work. In particular, we will give an in-depth explanation about tangles, triplet data, and current methods of clustering triplet data.

2.1 Tangles

Tangles have been a tool in mathematical graph theory, introduced originally by Robertson and Seymour (1991). They are quite flexible, with possible areas of application in sociology, psychology, economics, political science and others (Diestel, 2019). In recent times, through the work of **?**, they have been successfully applied to solve problems of clustering. The mentioned work delivers an algorithmic framework and theoretical guarantees for basic problem settings. Additionally, it delivers simplified notations, adapted to the domain of computer science. When talking about tangles, we will exclusively use the definitions introduced there, not those that might be common in mathematics.

In this section, we will now deliver a very brief recap of the basic notions, theory and applications of tangles in a clustering context. For more in-depth explanations of the algorithms and exact procedures, refer to **?**.

2.1.1 What is a Tangle?

The central object in tangles theory is a *bipartition* (which we also refer to as a cut). A bipartition is simply a way of dividing a set of elements $V = \{v_1, v_2, \ldots\}$ into two distinct subsets $A, B \subset V$, such that $A \cap B = \emptyset$ and $A \cup B = V$. We can also write a bipartition as $P = \{A, \overline{A}\}$, with $A \subset V$ and $\overline{A}$ being the complement of A with respect to V. For such a bipartition to be useful in clustering, we expect it to hold some degree of information about the cluster structure of our data. This means that a good bipartition should

not separate groups of data that are tightly coupled. On a graph, a good bipartition $P = \{A, \overline{A}\}$ might separate the set of nodes V such that there are only a few edges between A and $\overline{A}$. How useful a bipartition is will be quantified by a *cost function* $c : \mathcal{P}(V) \to \mathbb{R}$, with $\mathcal{P}(V)$ denoting the power set of V. One is free to choose this cost function and the performance of tangles will also depend on how well the cost function and the problem setting fit together.

Assume that for a set of elements V we have a set of bipartitions $\mathcal{B} = \{\{A_1, \overline{A_1}\}, \ldots, \{A_n, \overline{A_n}\}\}$ on V. This set of bipartitions, together with the cost function, should tell us a lot about the cluster structure of the data: for all bipartitions, we know how much they do or don't separate dense regions in V. This information in the bipartitions is aggregated and brought into a useable form by the tangles framework. For this, we process $\mathcal{B}$ to a set of so-called *tangles*. These correspond to specific ways of orienting the cuts such that they point to cohesive structures in the data. Orienting here means that we pick one specific side of a bipartition. An *Orientation* of $\mathcal{B}$ is then a set $O = \{o_1, o_2, \ldots o_n\}$, where o_i corresponds to either the partition A_i (oriented *left*) or $\overline{A_i}$ (oriented *right*). A consistent orientation (which we then call a *tangle*) is an orientation for which:

$$\forall A, B, C \in O : |A \cap B \cap C| \geq a \tag{1}$$

for some fixed parameter $a \in \mathbb{N}$, which we refer to as *agreement* parameter.

As is, a lot of sets of bipartitions wouldn't allow for any tangles, as there are too many cuts to consistently orient them. Imagine that our set of bipartitions would contain a few random cuts. On average, each of these halve our set of points, so we can at most orient on the order of $O(\log(n))$ many random bipartitions consistently. This can be resolved using the cost function: we can simply restrict our tangles to a set of low-cost (and thus very insightful) cuts P_ψ, using a threshhold $\psi \in \mathbb{R}$ such that $P_\psi = \{c(P) \leq \psi\}$. A tangle on P_ψ is then said to have order ψ. We have included a schematic drawing of a tangle on a simple data set in Figure 2.1.

2.1.2 Processing tangles to a clustering

As we have seen in Figure 2.1, a tangle might correspond directly to a cluster. But, a set of bipartitions usually allows for a wide variety of possible tangles, some of them pointing to different or overlapping clusters. We now have to process this set of tangles into a useable set of clusters. This step is a bit involved and we aim to only give a rough, intuitional overview here.

Given a set of bipartitions $\mathcal{B} = \{b_1, b_2, \ldots b_n\}$, we first want to determine for all possible orders ψ the sets of all tangles of order ψ on $\mathcal{B}$ according to a given cost function c. Intuitively, this order on the set of tangles determines

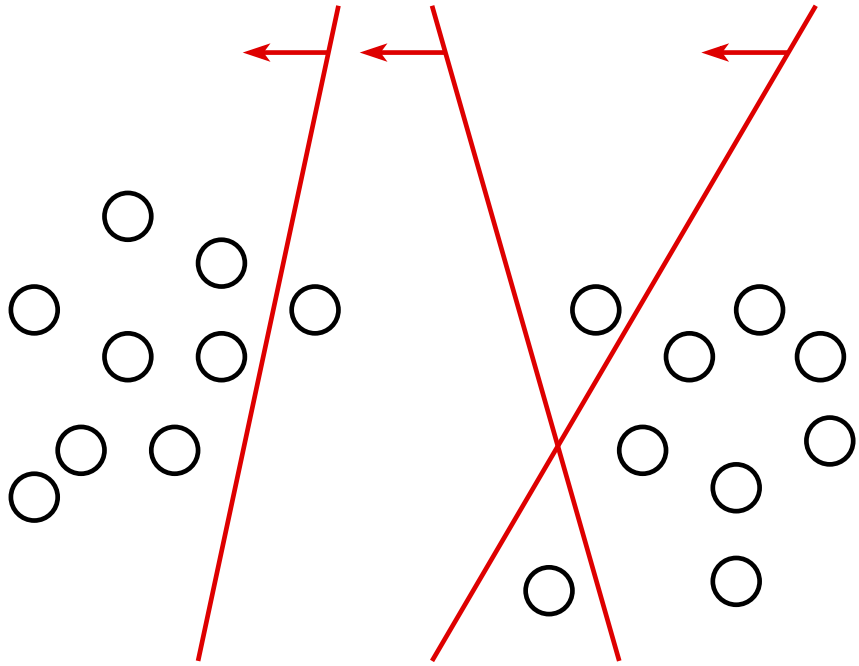


Figure 2.1: An example of how a simple tangle might look like if we assume a reasonably sized agreement (say $a = 3$). The red lines represent cuts which divide the sets of points into a bipartition. The tangle that is depicted here orients all bipartitions to the left side (indicated by the arrows), so that they point to the dense structure on the left. Another possible tangle might orient all cuts to the right. Notice that a tangle on this set of bipartitions can only either orient all bipartitions to the left or to the right, else the consistency criterion is violated. This might already give a good intuition on why tangles are able to find dense structures in data.

how coarse the clustering is they define. If we only use bipartitions with a low cost (so in the case of small ψ), then the bipartitions cut only through very loosely connected structures. If the cost is higher, the bipartitions are allowed to cut through more dense regions. This directly induces a sort of hierarchy, where we go from coarse cluster structures to fine ones with increasing ψ.

To aggregate the tangles of varying costs, we build a tree structure on the set of tangles, called the *tangle search tree*. In the tangle search tree, each node represents a tangle. Every level of the tree contains all possible tangles of a certain threshhold ψ_i which directly corresponds to the cost of bipartition b_i. The exact contents of the tangle corresponding to a particular node is determined by walking the path from the root to the node, adding bipartition b_i to the tangle in a left-oriented way, if it is a left child and in a right-oriented way if it is a right child. An exemplary tangle search tree is illustrated in Figure 2.2.

From this tangle search tree, we can now obtain a soft, hierarchical clus-

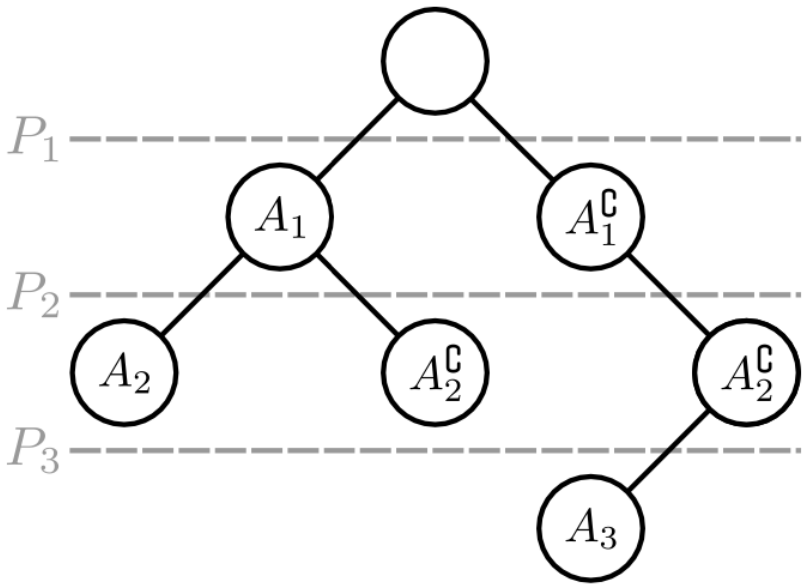


Figure 2.2: An example of a possible tangles search tree for a set of bipartitions $\mathcal{B} = \{\{A_1, \overline{A_1}\}, \{A_2, \overline{A_2}\}, \{A_3, \overline{A_3}\}\}$. Each level corresponds to the tangles of the order given by the bipartition P_i that is written on the dashed line to the left of it. As an example, if we would now want to determine the content of the node in level 3, we just walk from the root to it and add the bipartitions in the direction indicated by the tree. We end up with $T = \{\overline{A_1}, \overline{A_2}, A_3\}$. Figure taken with permission from **?**.

tering. *Soft* means that every data point is assigned a probability of belonging to each cluster. *Hierarchical* means that we have a hierarchy on the resulting clusters, which arises naturally from the form of the tangles search tree. For the clustering, the interesting nodes are those where the tree splits up (as this represents a new clustering) and the leaves (which correspond to the clusters). For each of these *splitting nodes*, we determine the set of *characterizing cuts*. A cut belongs to this set, if it is both oriented the same way inside the subtrees, and oriented in a different way between the two subtrees. To illustrate this, we take a look at the exemplary tree in Figure 2.2. Here, for the root node, P_1 is a characterizing cut (pretty trivially), while P_2 is not: below the node A_1, the bipartition is both oriented to the left and to to the right, violating the requirement that the cuts are always oriented the same way inside the subtrees.

To get a soft clustering, we can leverage the characterising cuts, which express some kind of consensus on how to align the bipartitions in the subtrees of the node. For a soft clustering, we have to determine with what probability a point a belongs to a cluster C. To do this, we start from the root. At every splitting node of the tree (which includes the root), we now examine

the set of characterising cuts. We then orient them in the same way they are oriented in the left subtree and count how many of these so oriented cuts contain a. Divided by the total amount of characterising cuts at the splitting node, we receive the probability p_L that a belongs to the left cluster complex. As the characterising cuts are always oriented differently in the two subtrees, the probability of a belonging to the right cluster complex is then given by $1 - p_L$. We can include these probabilites on the edges of our tree. To find out with what probability a belongs to C, we walk down the from the root to the leaf node that corresponds to C and simply take the product of all edge probabilites that we encountered along the way. By assigning each node to the cluster it belongs to with the highest probability, we obtain the corresponding *hard* clustering.

2.2 Ordinal Constraints

A dataset of ordinal constraints consists of object for which we don't know their exact attributes or features, but only how they relate to other objects in the form of comparisons such as *item i is closer to item j than to item k*. Formally, we assume that i, j, k are from a set D where we can define a dissimilarity function $d : D \times D \to \mathbb{R}$. Note that d can be a metric on D, but does not have to be. Using d, we can express our above constraint as $d(i, j) < d(i, k)$. Such data often appears when humans are asked to judge objects, as they often naturally perform better on comparing objects than on accurately placing them on an abstract scale (Stewart et al., 2005). Applications consist of estimating perceptual scales in psychophysics (Haghiri et al., 2020) or crowd-sourcing clustering algorithms (Ukkonen, 2017). We want to keep our focus here mainly the realm of psychophysics.

2.2.1 Forms of Ordinal Constraints

Ordinal constraints take one of two possible forms: quadruplets (used for example in Ghoshdastidar et al. (2019)) and triplets (used for example in **?**Haghiri et al. (2019)). A quadruplet is a quadruple (a,b,c,d) and expresses the following constraint on our data points: $\mathrm{d}(a, b) < \mathrm{d}(c, d)$. Analogously, a triplet is a triple (a,b,c), which expresses $\mathrm{d}(a, b) < \mathrm{d}(a, c)$. A triplet (a,b,c) can also by expressed by the quadruplet (a,b,a,c), making quadruplets strictly more general.

Datasets of triplet comparisons (which we also simply call triplet data) are almost always obtained by asking humans participants. For example, we might collect triplet data on images by presenting human participants with three images a, b, c, with image a as anchor point and ask them, *is b or c closer to a?*. The way that participants are questioned varies on the context

of the experiment (and might also influence their answers). An example is presenting the participant with three images, and asking *which is the most central image?*, or *which is the odd one out?*, but the results can then always transformed back to triplet format for further prcoessing. For example, if a is the *odd-one-out* of the three elements a, b, c, then we know that $d(b, c) < d(b, a)$ and $d(c, b) < d(c, a)$.

2.2.2 Ordinal embeddings

One of the most central problems when dealing with triplet data consists of finding a so called *ordinal embedding* of the data. If we have a set of triplet comparisons $T = \{t_1, t_2, \dots t_n\}$, of the form $t_i = (a, b, c)$, encoding that $d(a, b) < d(a, c)$, we want to find a set of points $y_1, y_2, \dots y_n \in \mathbb{R}^m$, such that they uphold most of the original triplet constraints in $\mathbb{R}^m$ with the euclidean distance as metric. More formally, we want to minimize (**?**)

$$\min_{y_1,\dots y_n \in \mathbb{R}^m} \sum_{t=(i,j,k)\in T} \mathbb{1}_{\|y_i - y_j\|_2 < \|y_i - y_k\|_2}.$$

This problem is difficult to optimize and thus various algorithms have been proposed that solve a relaxed or modified version of this objective function. These include Soft Ordinal Embedding (SOE, Terada and Luxburg (2014)), Generalized Non-Metric Multidimensional Scaling (GNDMS, Agarwal et al. (2007)), Crowd Kernel Learning (CKL, Tamuz et al. (2011)), Fast Ordinal Triplets Embedding (FORTE, Jain et al. (2016)), T-Stochatic Triplet Embedding (T-STE, L. van der Maaten and K. Weinberger (2012)) and various others.

It has been proven that if the original points come from the space $\mathbb{R}^m$, one can recover the points (up to scaling and orthogonal transformations) with $O(mn \log(n))$ many triplet comparisons (Jain et al., 2016). On this basis, an ordinal embedding can be used together with more classical machine learning algorithms, such as support vector machines or k-Means, for other machine learning tasks. Examples tasks are classification (Tamuz et al., 2011, Kleindessner and von Luxburg, 2017) or clustering (Kleindessner and von Luxburg, 2017).

Chapter 3

Applying Tangles to Triplet Data

As we have made out in section 2.1, the tangles algorithm operates on bipartitions of data that contain some information about the cluster structure. If we have obtained a set of triplet data, we are now faced with the task of processing this data to appropriate bipartitions. In this work, we have developed two methods for this which we call *landmark cuts* and *majority cuts*. We will elaborate on the methods and intuitions in the following sections.

3.1 Landmark cuts

In recent years, there have been algorithms that hope to speed up ordinal embedding by focussing on so-called *landmarks* (Ghosh et al., 2019, Anderton and Aslam, 2019), which are objects in the dataset for which we know all (or all relevant) triplet comparisons. The definitions of what constitutes landmarks varies a bit, but we will use the one that is used in Haghiri et al. (2019). Assume we have a set of objects D, as well as a set of T triplet constraints of the form (a, b, c), indicating that $d(a, b) < d(a, c)$. In a landmark setting, we have a set of m objects $L \subset D$, for which

$$\forall l_1, l_2 \in L \, \forall x \in D : (x, l_1, l_2) \in T \vee (x, l_2, l_1) \in T.$$

Landmarks make it very easy to define a set of bipartitions on triplet data: for each combination of landmarks l_1, l_2, we can make a bipartition $P = \{A, \overline{A}\}$ by assigning all points closer to l_1 to A and all those closer to l_2 to $\overline{A}$. Thus we can define the bipartition between landmark points l_1, l_2 as $A_{12} = \{x \in D \mid (x, l_1, l_2) \in T\}$, denoting as $P_L = \{A_{ij} \mid i, j \in \{1 \dots m\}, \, i < j\}$ all such possible bipartitions on L. These bipartitions are then called *landmark cuts*. Later in the simulations, as tangles is not reliant on having all triplet

constraints for all landmark objects, we will simply sample a subset $P'_L \subset P_L$ of bipartitions, which corresponds to repeatedly picking some objects a, b and sampling all triplet comparisons to all other objects $x \in D$.

Landmark cuts intuitively capture a cluster structure: the more close x is to l_1 according to d, the more likely that A_{12} will contain x. In the end, A_{12} will consist of the points that are in some sense close to l_1 (how strong this closeness is depends on how close l_1 and l_2 are). In the euclidean space, this notion is very easily captured: A landmark cut between l_1, l_2 is simply a linear cut between the two points, as illustrated in Figure 3.1.

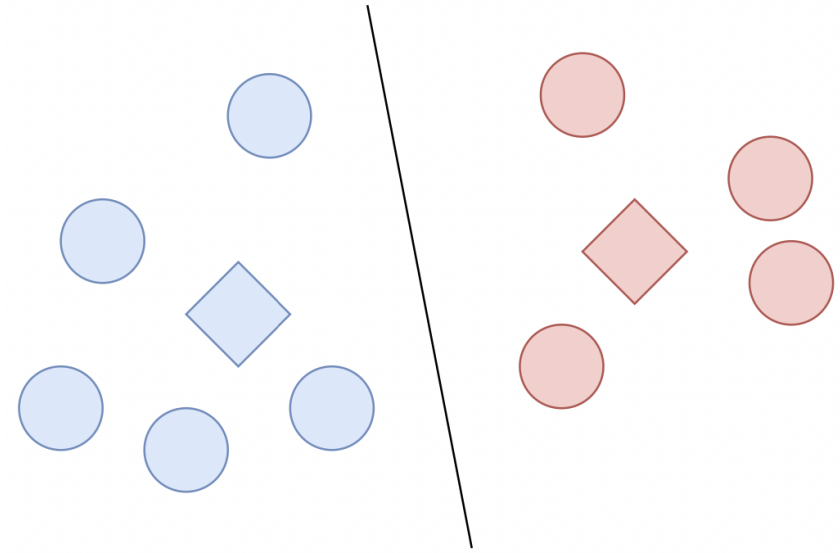


Figure 3.1: Example of a landmark cut on a euclidean data set, with the two triangles being the landmarks l_1, l_2, where all blue items (left of the line) would be assigned to A, all red items to $\overline{A}$ (or equivalently the other way around).

The landmark approach is a quite unusual way of sampling triplet questions. In most triplet experiments, the triplets to sample are chosen randomly (Kleindessner and von Luxburg, 2017, Haghiri et al., 2020) or according to some metric, for example maximizing some measure of gained information (Roads and Love, 2021). There was no experimental dataset available which is both sampled according to a landmark approach and exhibits a cluster structure, thus we rely on simulations for testing landmark cuts.

3.2 Majority cuts

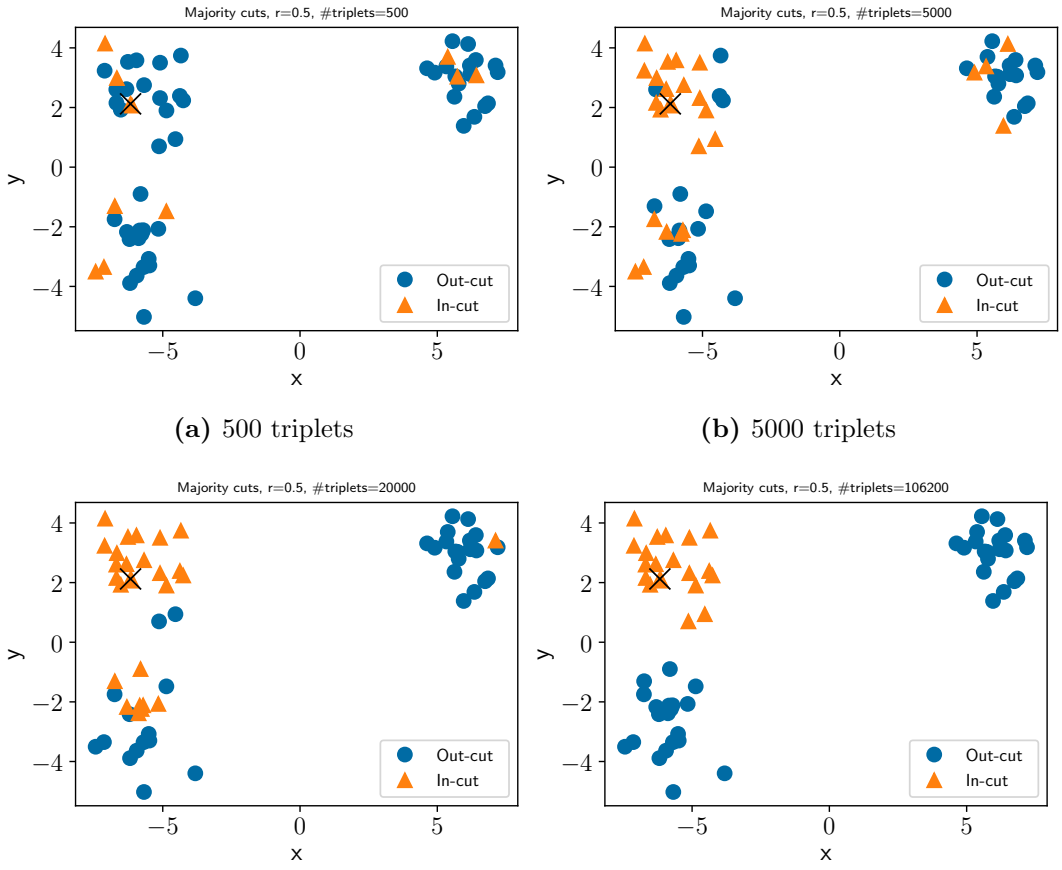
As explained in section 3.1, sampling triplet data in a landmark-fashion is not very widely used in current practice. Due to this, we aimed to also develop

a more general approach to processing tangles to cuts that can be applied to any set of triplet comparisons T regardless of the sampling method. For this, we again use the intuition that triplets tell us something about the closeness of data points, and thus about the cluster structure.

Assume again that we have a set of objects D and a set of triplet comparisons T. We fix two points $a, b \in D$. Assume that a, b are close and we sample a random point x. With high probability, it will be that $d(a, b) < d(a, x)$. The reverse holds if a, b are far away. Now we can take this the other way around: if we observe a triplet $(a, b, c) \in D$, then we can more reasonably assume that a, b are close than a, c being close. This leads us to the following method of defining a set of close points of a: Let $L_x = \{t \in T | t = (x, b, c), b, c \in D\}$ be the set of all available triplets where a is in the left position, and equivalently M_x and R_x the set of triplets where x is in the middle and right position. Then we define: $P_a := \{x \in D \mid |L_a \cap M_x| < |L_a \cap R_x|\}$ which is the set of all points that are more often closer to a than they are farther away. We refer to these bipartitions as *majority cuts*, and to the point a that as the *anchor point* of the cut P_a. These majority cuts capture some cluster structure by assigning points that are close together to the same bipartition. Majority cuts can be made more flexible by including a ratio r that controls the size of the cuts. We then define $P_a(r) := \{x \in D \mid |L_a \cap M_x| < r \cdot |L_a \cap R_x|\}$ and call r the *radius* of the cut. The definition of majority cuts given before is then simply the case $r = 1$.

Next, we want to gain some intuition about the form of the majority cuts. What we would expect, assuming that we had all triplets, is that each cut $P_a(r)$ is a ball around a that contains the $n \cdot \frac{r}{r+1}$ points that are closest to a according to the distance measure d that defined the triplets. In a euclidean setting for $r = 0.5$ we thus expect $P_a(1)$ to be a ball around a that contains the $\frac{n}{3}$ points that are closest to a. To test this, we plot a majority cut with a fixed anchor point on a simple mixture of gaussians in Figure 3.2. We can see that we get closer to the ideal form of a ball around a (the point marked with a cross) with radius $\frac{n}{3}$ as we increase the amount of sampled triplets. When we have fewer than all triplets available, our cut contains some noise in the form of points that in the cut together with a, but are outside of the $\frac{n}{3}$ ball around a (and might in fact be very far away).

This intuition already gives us hints how we might want to choose the radius: if we pick a smaller radius, we will detect smaller clusters versus when we pick a larger radius. In particular, we should not pick a radius smaller than the smallest cluster we want to detect, else the tangles algorithm will quickly have troubles consistently aligning the cuts. On the contrary, we are safe if we pick a radius that is a bit larger, as the tangles algorithm can work well with cuts that contain a cluster together with some additional data points (as long as they are not always the same points on all cuts).



(a) 500 triplets

(b) 5000 triplets

(c) 20000 triplets

(d) All (106200) triplets

Figure 3.2: Depiction of how majority cuts look like on a simple mixture of gaussians, with a varying number of triplets available. We plot just one bipartition P_a with radius $r = 0.5$, which was generated according to the procedure in section 3.2. The big X marks the anchor point a. The bipartition P_a consists of the orange triangles, which are the points that are twice as often closer to a than they are not (according to the drawn triplets). The blue points are those not in P_a.

Chapter 4

Simulations

In this chapter and the following one, we want to explore how tangles perform on triplet data with the methods we proposed in chapter 3. At first, we focus on simulations, as this allows us to control our data precisely. We will show in which cases tangles perform well, in which ones they don't and what to keep in mind when applying the algorithm.

As datasets, we have decided on two different datasets: a mixture of gaussians and a hierarchical block matrix matrix. In both cases, we will then generate triplets from the data points. The gaussian setup serves as first baseline: it is a de-facto standard in clustering and a lot of real-life data is approximately gaussian. We have decided against using more complex data sets such as two-moons, as the focus lies on how the algorithms interact with the generated triplets under various conditions (triplets being corrupted, triplets missing, et cetera...). The noisy hierarchical block matrix was introduced by Balakrishnan et al. (2011). We use it to illustrate a second property of tangles: in addition to pure clustering, we also produce a hierarchical tree, which can be used for hierarchical clustering.

4.1 Terms and methods used

To evaluate the performance of the tangles algorithm, we will need some metrics and other methods to use as a baseline. To compare a clustering against a ground truth, we have to use a scoring function that is independent of the cluster labels. One such function is the *normalized mutual information score* (NMI). For the NMI, 1.0 indicates the same clustering (up to label permutations), 0.0 indicates absolutely no mutual information between the clusterings (such as when our prediction puts all data points in a single cluster). To compare the quality of different hierarchies, we make use of the *average adjusted rand index* (AARI), introduced by Ghoshdastidar et al. (2019). The

AARI extends the *adjusted rand index* (ARI), which is a clustering performance measurement similar to the NMI, so that is capable of working with hierarchies. In the AARI, we calculate the ARI over all levels of the hierarchies we want to compare and average over all the obtained scores.

To generate the triplets, we first take the data and calculate a (dis)similarity on it, for example the euclidean distance between two data points. This allows us to determine whether (a, b, c) or (a, c, b). Then, we can use two approaches of drawing triplets. The first one is sampling triplets randomly and uniformly from the set of all triplets. The second one is a landmark approach: we fix two randomly sampled points a, b, with $a \neq b$ and sample all triplets that have the form (x, a, b) or (x, b, a). For the landmark tangles, only the second kind of triplets are useable, as discussed in chapter 3, thus we will mostly stick to this format.

We now have two possible ways of altering the triplets. First, we can add *noise* to the triplets. If we have a noise level of p, then every triplet is flipped with probability p, meaning that (a, b, c) would be turned into (a, c, b). We can also reduce the *density*, meaning that instead of sampling all triplets, we sample only a fraction d of all triplets. From now on, when we use the term density, we will refer to triplets sampled in a landmark approach, while when explicitly using the number of drawn triplets, we refer to a uniform sampling.

As a baseline, we will be using a combination of ordinal embedding together with a clustering algorithm. We first use an ordinal embedding algorithm to get an embedding of the triplet data. This approach has also been used in Kleindessner and von Luxburg (2017). It has the advantage of also working with triplet data as opposed to clustering on the original data directly, giving a more fair baseline. Afterwards, we use a standard clustering algorithm for euclidean data on the obtained embedding. There exist numerous algorithms for ordinal embeddings (see **?** for an overview) and for clustering. We have decided to use Soft Ordinal Embedding (SOE Terada and Luxburg (2014)), as this has been identified by **?** as one of the top-performing ordinal embedding algorithms for a variety of use cases, which was also the case for us when comparing different baseline algorithms. Additionally, we have included T-Stochatic Triplet Embedding (T-STE L. van der Maaten and K. Weinberger (2012)) to have a second baseline. For the clustering algorithm, have decided to go with k-Means (Lloyd, 1982), as one of the most basic clustering algorithms with usually good performance.

As another baseline we have included ComparisonHC (Ghoshdastidar et al., 2019). This, like tangles, is also an algorithm that does not construct an embedding before clustering, and thus might allow for a more fair comparison. Especially in the setting of a mixture of gaussians, the fact that ordinal embedding algorithms aim to reconstruct a euclidean embedding seems to already introduce some bias that might benefit the ordinal embedding algorithms. Nor-

mally, ComparisonHC is a hierachical clustering algorithm, which produces a dendrogram as its output. To use it as a baseline for clustering, we simply cut the dendrogram off such that it produces the desired amount of clusters.

In the experiment figures and discussions, we will use the abbreviations L-Tangles for landmark tangles, and M-Tangles for majority tangles.

4.2 Gaussian data

4.2.1 Experimental setup

We generate a mixture of gaussians as follows: We draw a number of points n each from k different gaussian distributions with means $\mu_1, \mu_2, \dots \mu_k$ and variances $\nu_1, \nu_2, \dots \nu_k$. Each point gets assigned a label y_j that corresponds to the number $i \in \{1, \dots k\}$ of the gaussian distribution it was drawn from. For all of the experiments, unless mentioned otherwise, we use $n = 20$, $k = 3$, fixed means $\mu_1 = \begin{bmatrix} -6.0 & 3.0 \end{bmatrix}, \mu_2 = \begin{bmatrix} -6.0 & -3.0 \end{bmatrix}, \mu_3 = \begin{bmatrix} 6.0 & 3.0 \end{bmatrix}$ and a constant variance for all distributions of $\nu = I$, with I as the identity matrix. A plot of the data set can be seen in Figure 4.1. We generate the triplets via the euclidean distance between the points, so that the triplet (a, b, c) implies that $\|a - b\| < \|a - c\|$. For the Tangles algorithms, we use an agreement of $a = 7$ (around 1/3 the size of the smallest clusters we want to detect, in accordance with **?**), and a radius of $r = \frac{1}{3}$, for the majority tangles, such that the cuts roughly have the diameter of the clusters.

All results are the average of 50 runs, where we re-draw both the data points from the gaussian as well as as triplets randomly.

4.2.2 Lowering density

First, we want to observe how our methods behave under different numbers of triplets present. As the number of triplet grows on the order of $O(N^3)$ with the total number of points N in the data set, it is only feasible to obtain all triplets for very small datasets. Even with small $N = n \cdot k = 60$, as in our case, there are already 106200 triplets possible. Those triplets have to be obtained through experiments with real people in most settings. If an algorithm performs better with a lower amount of triplets, this can quickly translate into really large time, labor and money savings.

To test our algorithm, we have drawn an increasing amount of triplets in a landmark format from our dataset. We have plotted the results for a small (60 points) and a large (600 points) dataset in Figure 4.2. Usually, the larger the dataset, the smaller the percentage of all triplets we use, as the number of triplets grows with $O(N^3)$. However, ordinal embedding algorithms

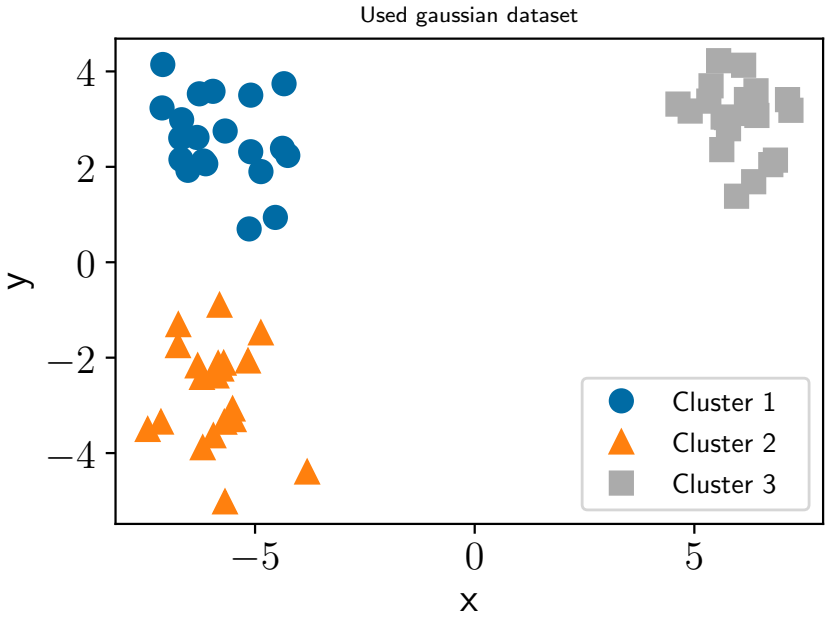
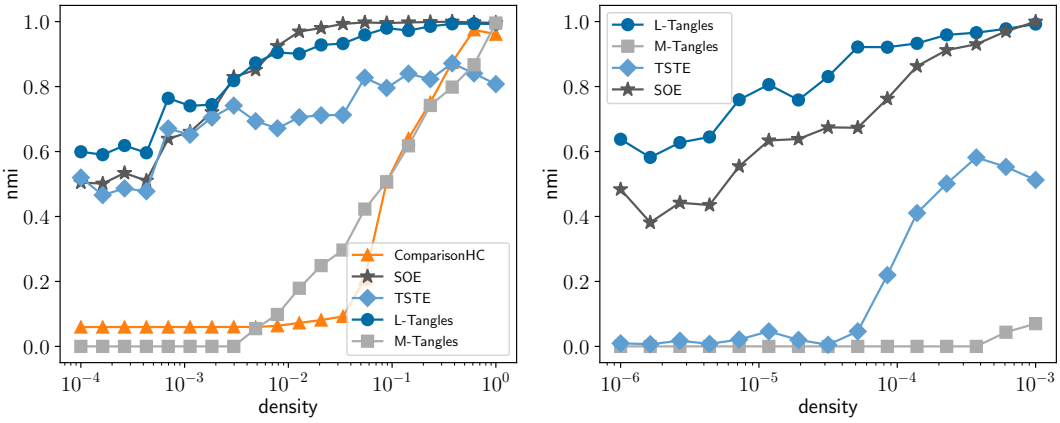


Figure 4.1: An example draw of the gaussian mixture used for our experiments.

empirically perform well with a lot less triplets (for example requiring on the order of $O(nd \log(n))$ for euclidean data (Jain et al., 2016)). Ideally, we would like for the Tangles algorithm to behave similarly.

When looking at the plots, we can see that L-Tangles is performing at least as well as SOE, and even significantly better for a wide range of densities in the case of $n = 200$. Tangles thus shows the desired behaviour of requiring a smaller percentage of total triplets with a higher number of data points. As we would have expected, T-STE performs slightly worse than SOE on the small dataset, but interestingly a lot worse for the larger dataset. ComparisonHC and M-Tangles perform about the same level, but both stay far behind L-Tangles and SOE. With the larger dataset, we could not test ComparisonHC, as our obtained implementation requires constructing a n^4 matrix during the training step (this could possibly be remedied using a different implementation).



(a) 20 points per cluster **(b)** 200 points per cluster

Figure 4.2: We plot the NMI of different clustering method against the percentage of the triplets generated from a draw of a gaussian mixture with 3 clusters. The triplets are generated in a landmark format. We draw 20 data points from each cluster for the left plot, and 200 data points for the right plot. On the x-axis we have the density, where a density of 0.1 means that we only use 10% of the total number of triplets. The embedding methods (SOE, T-STE) are followed by k-Means. The Tangles methods (L-Tangles, M-Tangles), are applied with $a = 7$ for $n = 20$ and $a = 70$ for $n = 200$. ComparisonHC was left out of the right plot due to computational issues.

4.2.3 Adding noise

Next, we want to observe how our algorithm behaves under added noise. This is an important model: most applications of triplet data use triplets that are generated from real humans. They might disagree on which objects are closer and which are not, which can be modelled as noise on the responses of our triplets. The higher the noise, the more disagreement there is about the similarities of objects, so it would matter more about which person you ask than which objects you present them.

We have plotted the performance of our algorithms over increasing noise in Figure 4.3, with all triplets sampled (density 1.0). We observe that both landmark and majority tangles off more quickly in performance with increasing noise than the other algorithms, with L-Tangles still being better than M-Tangles. We observe however, that until noise levels of 0.1, all algorithms performs the same, meaning that L-Tangles can still perform well with low to medium levels of noise. Interestingly, ComparisonHC, which required a lot of triplets to achieve acceptable performance, seems very noise resistant, performing about as well as SOE until noise levels of 0.3.

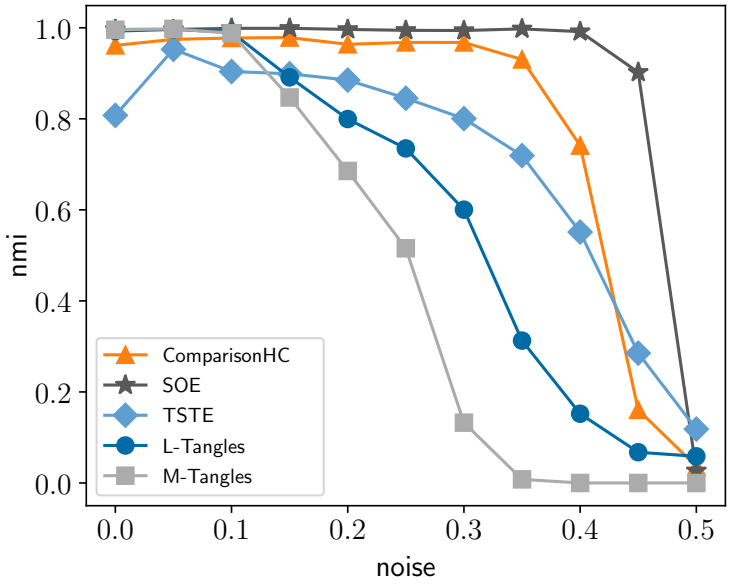
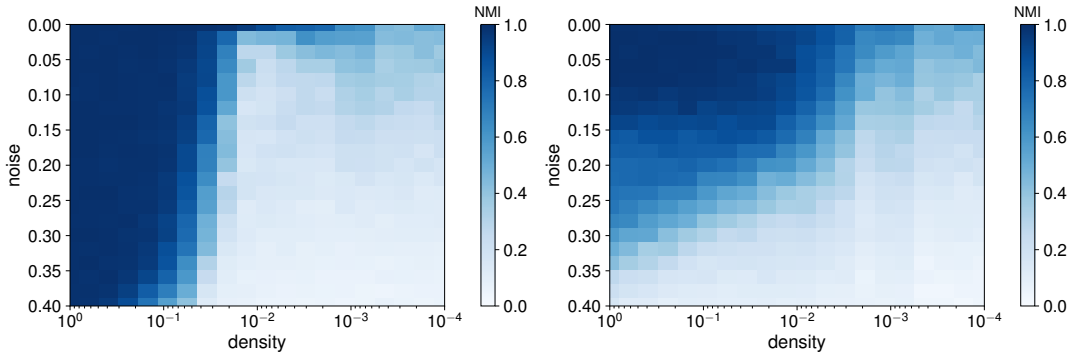


Figure 4.3: We plot the NMI of our chosen clustering methods against the noise that we introduce on the triplets. We use 3 clusters and 20 data points per cluster and sample all possible triplets. A noise level of 0.1 means that we flip 10% of the triplets (turning for example (a, b, c) to (a, c, b)).

4.2.4 Adding noise and lowering density

In this experiment, we will merge subsection 4.2.2 and subsection 4.2.3. We have seen that L-Tangles perform well with a low amount of triplets (a bit better than SOE) and has reasonable performance for noisy triplets (albeit worse than SOE for high noise). We are now interested to see how large the area is where L-Tangles can still outperform SOE when we consider both a low density and noisy triplets, as these two factors are probably the most interesting variables when choosing an algorithm to evaluate triplet data.

We have generated two heat maps, which can be found in Figure 4.4. There we can see that L-Tangles outperforms SOE in quite a large region in the low-noise , low-density regime. We would imagine this effect to be even larger for a $n = 200$ setup, but found this computationally too intensive. On the contrary, SOE performs better in the high-density, high-noise regions, with about similar performance for the cases of low-noise high-density (perfect clustering) and high-noise low-density (random clustering).



(a) SOE **(b)** L-Tangles

Figure 4.4: We plot a heatmap of the NMI of SOE and L-Tangles over noise (y-axis) and density (x-axis) of our the triplets, generated from a mixture of gaussians with 3 clusters and 20 data points per cluster. The triplets are drawn in a landmark format. The regions with a darker shade of blue indicate better performance of the algorithm. Note that the noise increases as we move down, and the density decreases as we move right. This means that clustering gets harder when moving right and/or down and easier when moving left and/or up.

4.2.5 Missing triplets

In this experiment, we will use the small gaussian data set and gradually sample an increasing number of triplets for it. This is very similar to the setup in subsection 4.2.2, but this time we sample triplets uniformly at random without replacement from the set of all triplets. In this setup, we will not be able to use landmark tangles as is, as there will be missing values in the cuts.

An idea would be to impute the missing triplets for the cuts. For, this we just build up our landmark cuts as we did before, but we mark entries for which we don't have triplet information. These missing values can then be imputed via different methods. We have used *random*, *k-nearest neighbour* (k-NN) and *mean* imputation. Random simply sets all missing values to 0 or 1 with equal probability, k-NN imputes a missing entry in a landmark cut with the value that the most similar cut has in that position (closeness being calculated via the manhattan distance), and mean imputes the value with the mean of all other cuts in that position. We have shown the results for our imputation method in Figure 4.5. One can see that k-NN performs quite reasonably, and the simplest choice of $k = 1$ even achieves the best performance.

We have compared the performance of our different algorithms versus the number of triplets drawn in Figure 4.6, where L-Tangles was imputed with 1-NN. Here, L-Tangles loses out against SOE, which achieves an acceptable level of performance at much lower amounts of triplets. However, we can see that

it is competitive against M-Tangles and ComparisonHC. T-STE also reaches an acceptable performance earlier, but saturates quicker at an NMI of 0.8, compared to the other methods, which achieve an NMI of > 0.9 with > 10000 triplets

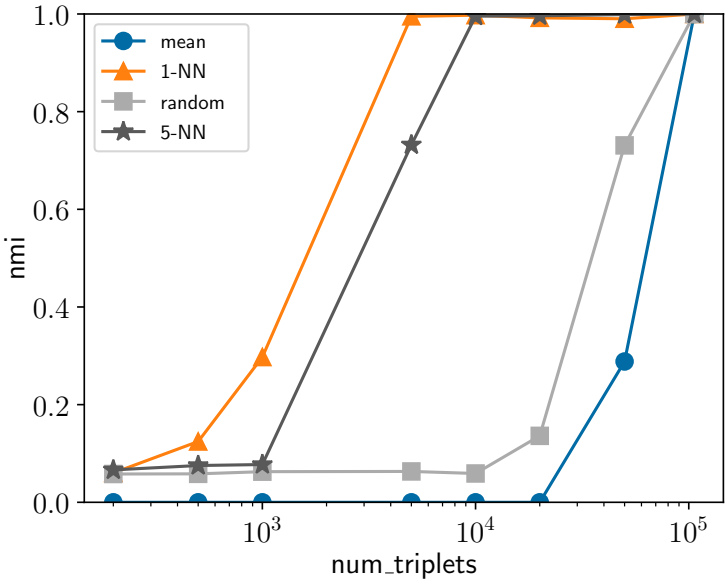


Figure 4.5: We plot the performance of L-Tangles over the number of triplets available for different imputation methods over our small gaussian dataset with 3 clusters and $n = 20$. We use different imputation methods to fill in the missing values and then cluster the resulting cuts with L-Tangles. The *k-NN* methods use k-nearest neighbour imputation, *random* assigns 0 or 1 to the missing values with equal probability, and *mean* assigns the missing values the mean value over all other cuts at that position.

4.3 Hierarchical data

4.3.1 Experimental setup

We generate a noise hierarchical block matrix (Balakrishnan et al., 2011). The hierarchy described by this model has the form of a complete binary tree, and the similarities of the data points are described via a similarity matrix M, where the entry $M_{i,j}$ describes the similarity between the points i, j. In this matrix, the elements in the same cluster have the highest similarity μ_0, and for each level l that two classes are removed from each other in the hierarchy, their similarity decreases by δ. We can also correupt the similarity matrix by an added noise matrix R. We then receive the noisy hierarchical block matrix $M' = M + R$. In our setup, we use a noise matrix R where every entry is

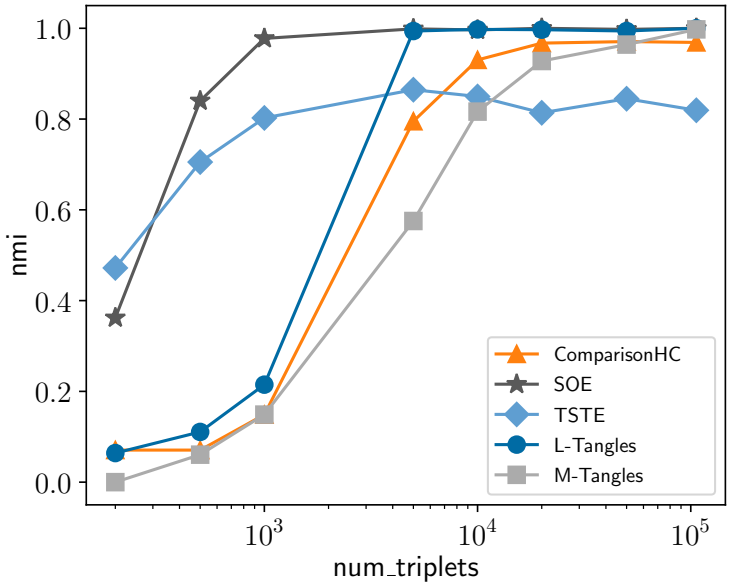


Figure 4.6: Analog to Figure 4.5, but this time we plot the performance of various other evaluation methods on the small gaussian data set against L-Tangles. We impute the missing values in the L-Tangles algorithm with 1-NN.

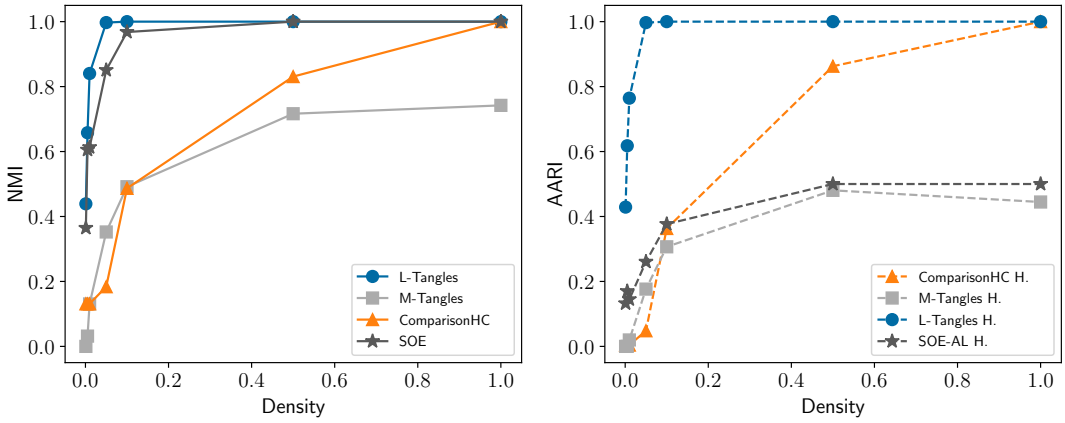
simply drawn from a normal distribution with mean 0 and standard deviation σ. More about the generation process can be read in Ghoshdastidar et al. (2019).

We choose a relatively simple setup of 4 clusters with 10 data points each, an initial class similarity $\mu_0 = 5$ and a similarity decrease of $\delta = 1$. In this setup, there are two kinds of noise we encounter: the noise that is injected into the hierarchy itself via the noise matrix and the noise that is added to the triplets. The triplet noise has the same form as in section 4.2, on noise p we simply flip every triplet with probability p. We will investigate how our algorithm does under both noise models, as well as under a lowered density. When evaluating, we compare both the final clustering (the lowest level of the hierarchical block matrix) as well as the obtained hierarchy to each other. To produce a hierarchy with SOE, we have applied an agglomerative clustering algorithm with average linkage (AL) on the obtained embedding. As in the gaussian setting, all results are the average of 50 runs, where we re-draw both the data points as well as the triplets randomly.

4.3.2 Lowering density

As in subsection 4.2.2, we investigate how our algorithms perform under a lowered density. We again draw the triplets in a landmark-format. The results can be seen in Figure 4.7.

We can see that L-Tangles performs about on-par with SOE in the final clustering case (not taking the hierarchy into account), and greatly outperforms all the other algorithms, while M-Tangles and ComparisonHC perform about equally well, wtih ComparisonHC getting better results in the high triplet case.

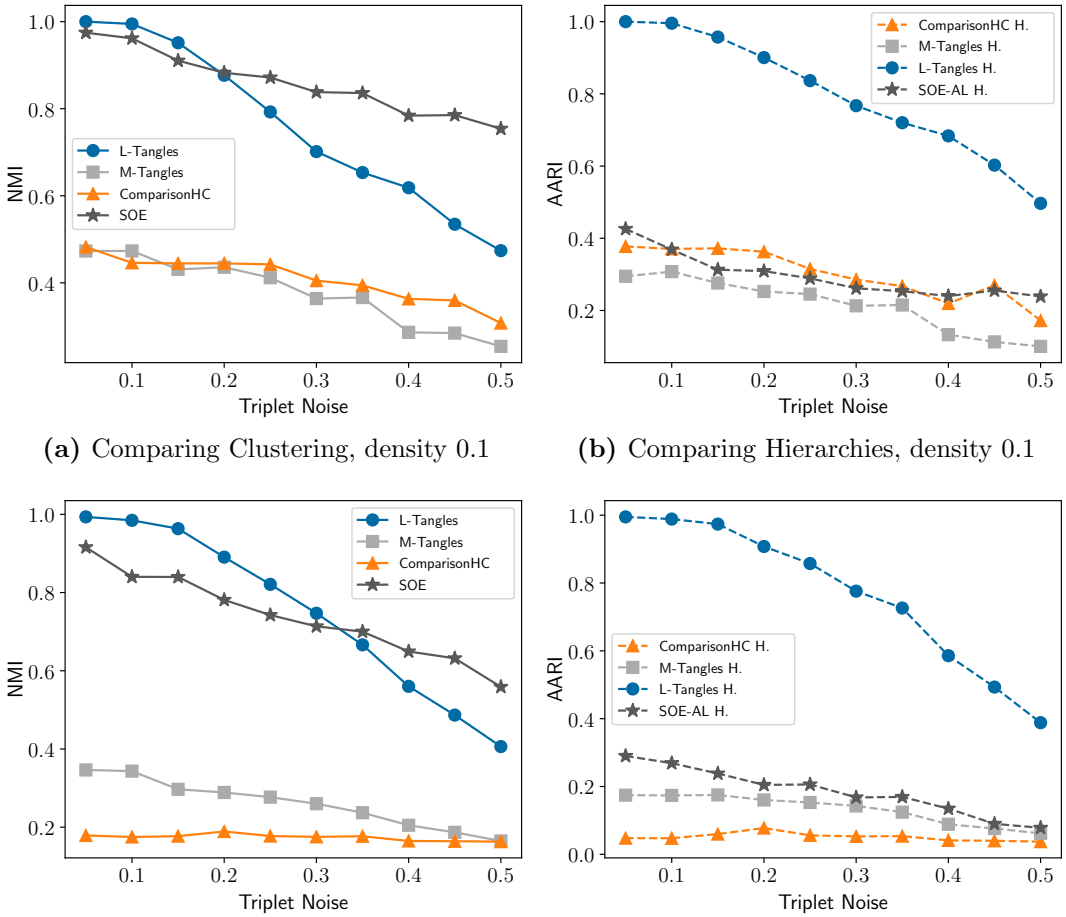


(a) Comparing Clustering **(b)** Comparing Hierarchies

Figure 4.7: We plot the performance of our algorithms against the density of the triplets drawn. We draw the triplets in a landmark format and they are generated from a hierarchical noise matrix with 4 clusters and 10 data points each. In the left, we see the NMI of the obtained clusterings versus the ground truth, where we only use the final clustering to evaluate. The ordinal embedding algorithms (T-STE, SOE) have been followed by k-Means. On the right, we plot the AARI of our methods against the density, where we take the whole hierarchy into account. To obtain a hierarchy for SOE we have applied agglomerative clustering with average linkage to the obtained embedding instead of k-Means.

4.3.3 Adding triplet noise

Similar to the gaussian setup in subsection 4.2.3, we have simply increased the noise on the sampled triplets and have evaluated the performance of our algorithms. The results can be seen in Figure 4.8. We have have repeated the setup under two different densities, with 10% and with 5% to see if the density had influence on the noise suspectibility of the algorithms.



(a) Comparing Clustering, density 0.1

(b) Comparing Hierarchies, density 0.1

(c) Comparing Clustering, density 0.05

(d) Comparing Hierarchies, density 0.05

Figure 4.8: Here we plot the performance of our algorithms against the noise introduced on the triplets, meaning that if we have a noise of 0.1, we flip a triplet with probability 10%. We again use a hierarchical block matrix with 4 clusters and 10 points each. On the top row, we draw all triplets in a landmark format, on the bottom row we drawn 10% of them. We draw 20 data points from each cluster in the left column, and 200 data points in the right one.

First, we will look at the pure clustering. In the 10% density case, SOE outperforms all other methods unless the noise is low (< 0.2), where L-Tangles performs better. If the density decreases (5%), L-Tangles keeps the advantage until higher noise levels (to about 0.35). This is similar to what we have seen in the gaussian setup. L-Tangles and SOE however perform a lot better than ComparisonHC and M-Tangles in for both densities and all noise levels. For the hierarchy, L-Tangles greatly outperforms all other methods for both densities, with the other methods performing on roughly the same level. Interestingly, we see that for both comparing hierarchies as well as clustering, ComparisonHC seems very dependent on the number of triplets present, as the NMI on all

noise levels about doubles when we go from 5% to 10% density.

4.3.4 Adding hierarchy noise

This setup is a bit different than the experiments done on the gaussian data. Here, we vary the noise that we add directly to the hierarchical block matrix $M' = M + R$. R is a matrix whose entries all consist of gaussian noise that is drawn from a normal distribution with mean 0 and variance σ^2. We now set σ^2 to various values and evaluate the performance of our algorithms. The result can be seen in Figure 4.11. In the normal clustering case, SOE performs the best over the board, with L-Tangles falling off pretty sharply on the introduction of noise into the hierarchy, but still outperforming M-Tangles and ComparisonHC. In the hierarchical case, L-Tangles outperforms all other methods until hierarchy noise of 2, whereafter SOE and L-Tangles achieve similar performance. We assume that this is because the similarity difference between two completely unrelated classes is at most 2. After a noise level of 2, we thus get into a regime where the noise simply overpowers the information left in the similarity matrix.

We can see one interesting effect here: L-Tangles shows a steep decline in performance after the introduction of even a small amount of noise, which we not present in our other noise model, where the noise was added to the triplets directly (see subsection 4.3.3). In the other noise model, we saw a much smoother decline in performance with added noise on the triplets. In fact, we have tested this, and even when adding a vanishingly small amount of noise (say 10^{-6}) we see the decline in performance. This could illustrate a potential shortcoming of the tangles algorithm, which we will now elaborate on.

First, we want to get some intuition about the hierarchical model. As it is a bit hard to visualize, We have plotted a representation of our hierarchical model in Figure 4.9. Care should be taken: this does not accurately visualise the distances between the points in any way, but is merely a useful tool to illustrate cuts on the data. We will now look at the landmark cuts that we retrieve under our two noise models, with a noise of 0.1 in the triplet noise case and vanishingly small noise $\epsilon = 10^{-6}$ in the hierarchy noise case.

Assume that we select two points, one from one of the left clusters, and one from one of the right clusters. When we generate the landmark cut that is associated by those two points (by putting all points that are closer to the left point in the cut, and taking all points that are closer to the the right point out of the cut) we receive is a *coarse* cut, that divides a higher level of the hierachy (roughly into left and right). As we see in Figure 4.10a and Figure 4.10b, this cut looks pretty similar under both noise models. In the triplet noise case, we can see that some of the data points have been assigned to the wrong cluster,

but this is expected.

Next, we will take two points, a from the bottom-left, b from the top-left cluster. The resulting landmark cut is a *fine cut*, that separates between lower levels of the hierarchy. In this setting, something interesting happens in the hierarchical noise model: the points from the right clusters get randomly assigned to in-cut or out-cut. To understand this, let's first look at what happens when we have no hierarchy noise. In the hierarchical block model, all distances are only dependent on how far removed the points are in the hierarchy, thus the bottom-left and top-left clusters are correctly separated by the cut. All points from the bottom-left cluster will be in the landmark cut, all cuts from the top-left cluster will be out of it. Let now c be a point from the bottom-right or top-right cluster. Then, $d(c, a) = d(c, b)$. If we don't have noise on the hierarchy, this does not pose a problem: in our landmark cut implementation, we have decided to break ties deterministically, ruling c is in the landmark cut (a, b) only if $d(c, a) < d(c, b)$. Thus, c will not be contained in (a, b). As a result, the cut (a, b) will contain only the bottom-left cluster and no other points. However, when we add the tiniest amount of noise to the hierarchical model, then for all points c' from on of the right clusters it will randomly be either $d(c, a) > d(c, b)$ or $d(c, b) > d(c, a)$. This means that those points will be randomly assigned to the landmark cut (a, b) as either in-cut or out-cut. This can also be seen in Figure 4.10c. For the triplet noise case, we have the ideal assignment (all points from the right clusters are assigned out-cut), but some points are again randomly assigned wrongly, see Figure 4.10d.

Now, how does that influence our clustering? Let us step through an example clustering that L-Tangles would make with a hierarchical noise model. At first, tangles would receive the coarse cuts (they are cheaper as they are more balanced and more frequently present) and subdivide the points into the left and the right clusters. Next, at some point, we would need to align one of the fine cuts. This will subdivide either the left or right clusters (depending on which cut we receive). but the random assignment in the other cluster (that one which is not subdivided) prevents us from orienting the fine cut of the other clusters consistently (depending on agreement, but if we have to align two fine cuts of the same cluster after another, even a very small agreement will not allow consistent alignment). Thus, we will end up with three clusters, the two clusters that are subdivided by the first fine cut that appeared, and the other two clusters merged to one.

On the other hand, if we only have to deal with (low) triplet noise, we can align the first coarse cut in any way we want, and the fine cut then gets aligned in one direction in the left subtree and in the other direction in the other subtree. We end up with the correct amount of clusters, and the miss-classifications will only be a few random points that are assigned to a wrong cluster due to triplet noise.

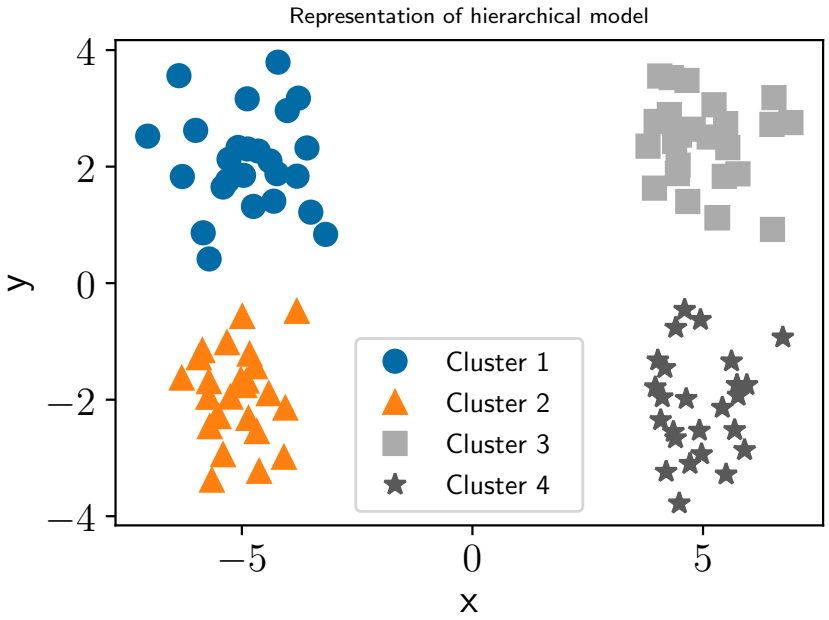
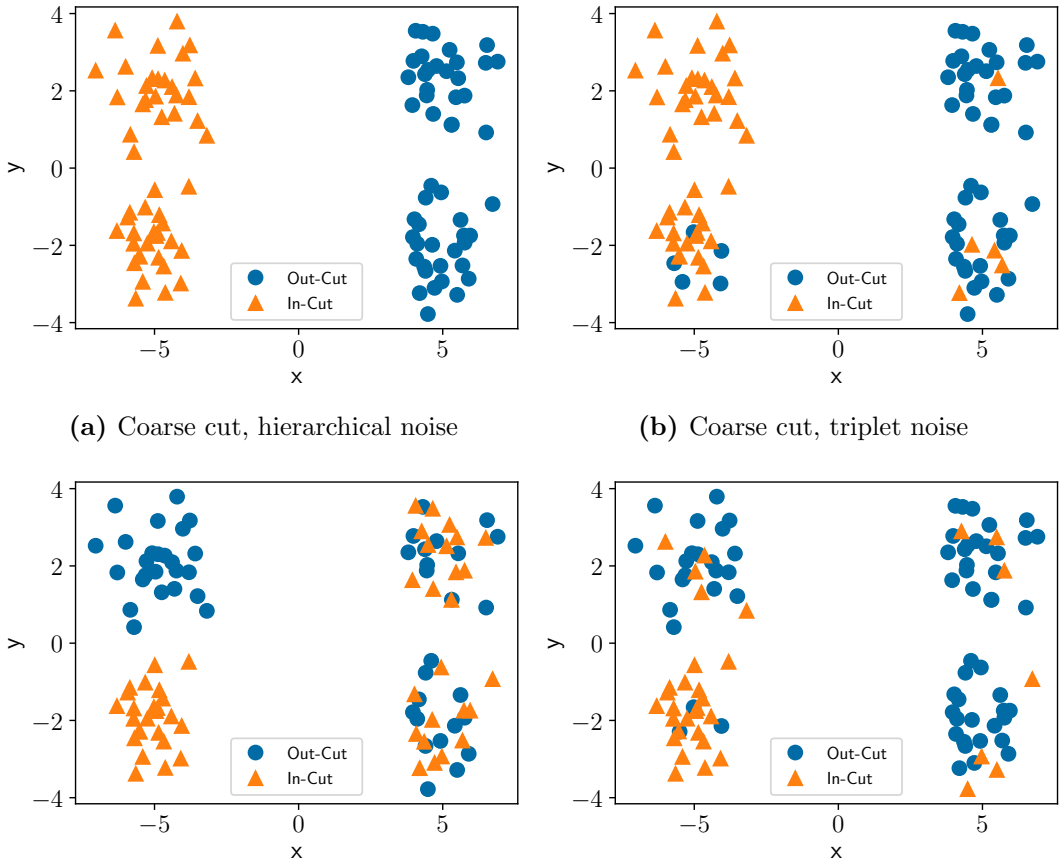


Figure 4.9: Here we plot a euclidean representation of our hierarchical model with a few more data points drawn. We can see a sort-of hierarchy between the clusters, where the left and right side are two far removed clusters, which can be subdivided in bottom-left, top-left and bottom-right, top-right. Keep in mind that this representation is only a visualisation aid and does not accurately reflect the actual similarity between data points.



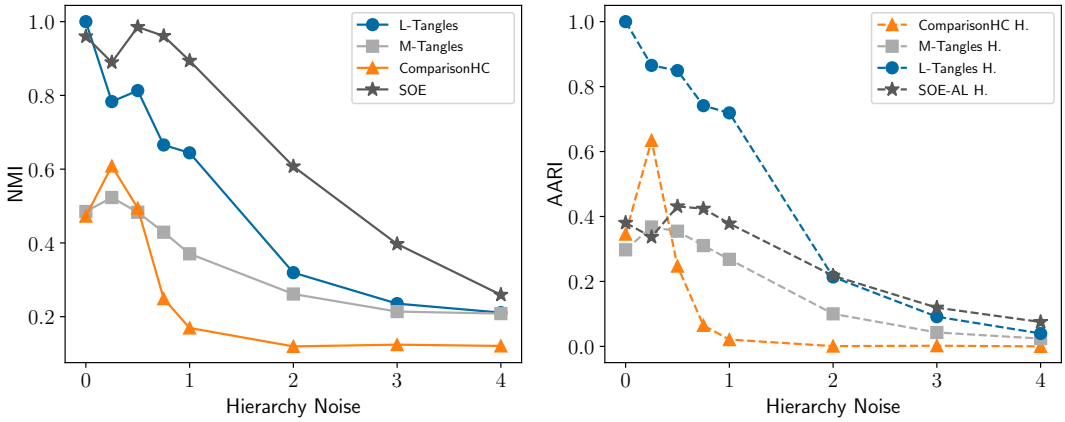
(a) Coarse cut, hierarchical noise

(b) Coarse cut, triplet noise

(c) Fine cut, hierarchical noise

(d) Fine cut, triplet noise

Figure 4.10: Visualisations of the cuts we would receive under both noise models we deal with in a hierarchical setting, analog to Figure 4.9. Hierarchical noise means noise that we add directly to the hierarchical similarity matrix, while triplet noise means the percents of triplets answered wrongly. We assume landmark cuts. The coarse cuts are those that separated higher levels of the hierarchy (so left and right clusters), while the fine cuts further subdivide left and right into bottom-left, top-left, bottom-right and top-right.



(a) Comparing Clustering **(b)** Comparing Hierarchies

Figure 4.11: We plot the performance of our algorithms against the noise on the hierarchical block matrix. A noise of 1 means that the each entry in the similarity matrix of the hierarchies is independently corruped with additive gaussian noise $r_{ij} \sim \mathcal{N}(0, 1)$. We again use a hierarchical block matrix with 4 clusters and 20 data points. On the left, we report the NMI of the final clustering against the ground truth. On the right, we also take the hierarchies into account, reporting the AARI.

4.3.5 Discussion

In this section, we have reported the performance of the Tangles algorithm on two synthetic data sets, gaussian and hierarchical data. We could see that with landmark triplets, L-Tangles consistently had the best or among the best performance, even outperforming the SOE in regimes of low noise. Additionally, L-Tangles proved to be the best performing algorithm when trying to determine the hierarchy in our hierarchical model. If we are not presented with landmark triplets, we could observe that L-Tangles is still capable of reaching an acceptable performance (a bit better than ComparisonHC, which is a state-of-the-art clustering method for clustering triplets without creating an intermediate representation). Interestingly, we found ComparisonHC to be very reliant on a having a large amount of triplets available to have an acceptable performance, being outperformed by L-Tangles in almost all settings (besides in the case of high noise and almost all triplets available).

Overall, M-Tangles did fare worse than L-Tangles, but it still had acceptable performance, that was overall comparable to ComparisonHC. However, M-Tangles introduces another hyperparameter, the radius, which needs to be tuned. Thus, if triplets are present in landmark format, we strongly prefer L-Tangles. Even if the triplets are not in landmark format, imputing the triplet responses with a simple 1-NN method and then using L-Tangles seems pre-

ferrable to using M-Tangles. We have still included M-Tangles because we think it can serve as an interesting baseline from which to build more sophisticated methods, for example one could think about weighing the triplets in a certain way when counting them up (if we have two points A, B and we already know they are very close, then (A, C, B) is a strong indicator of C and A being in the same cluster).

We have also shown some cases where the tangles algorithms performs subpar: for example in the case of high noise or a large amount of missing triplets (non-landmark format). We have also raised some issue with the noise in the hierarchical model, and we now want to discuss whether this points to an artifact in the model or a more serious flaw in the tangles algorithm.

In subsection 4.3.4, we have used two different noise models: adding noise to the triplets and adding noise to the hierarchical block model. This points to two fundamentally different assumptions about our data. Let's think of our data as a hierarchy of 2 clusters, which separate again into 2 clusters. The clusters are fruits (apples, bananas) and vegetables (zucchini and potatoes). If we wanted to cluster this data using triplet data, we would gather a lot of people, present them with three images of our objects, and ask them whether the first image is more similar to the second or third one. In the triplet noise model, we assume that either some people answer "wrongly", or that some objects might actually be more similar to an object from another cluster than to ones from their own cluster (maybe we have an image of a banana and an image of a zucchini that have both been shot in front of a beach and thus look similar). Most of the time however, there is some kind of hidden way to compare items from entirely unrelated hierarchies. In general, apples might always be more similar to potatoes than to zucchinis for example. As a consequence, if we for example have a landmark cut from an apple and a banana, it would contain all the apples and all the potatoes.

In the case of hierarchy noise, we simply have some objects that are flat out more similar to one category than another. For example, in the set of apple images, we might have a lot of green apples, which are almost always more similar to zucchinis than to potatoes, and a lot of yellow apples, for which it is the other way around. For this reason, when we look at a landmark cut of a zucchini, this one might contain all bananas and all green apples. As explained in subsection 4.3.4, clustering this poses a problem for the tangles algorithm. The actual amount of noise added doesn't matter (as long as it is smaller than the similarity decreases between distances), as this information gets lost when turning the similarity matrix into triplets.

Overall, which of the two noise models sounds more realistic might depend completely on the problem setup, how we present the items, how we ask the triplet questions et cetera. Nontheless, even in the hierarchical noise model, L-Tangles has a very good performance, meaning it can be used without thinking

too much about which noise model we are confronted with.

Chapter 5

Case study: data from psychophysics

In this chapter, we want to use the tangles algorithm for real world data. The chapter aims to not only show the viability of tangles as a data evaluation tool, but also to serve as an instructive application for how a researcher might use tangles for their own experiments. We will use the data from the thesis of Schönmann (2021) and will complement some of her analysis using tangles. This could be used to gain some more insight about the data and their relations.

5.1 Data background

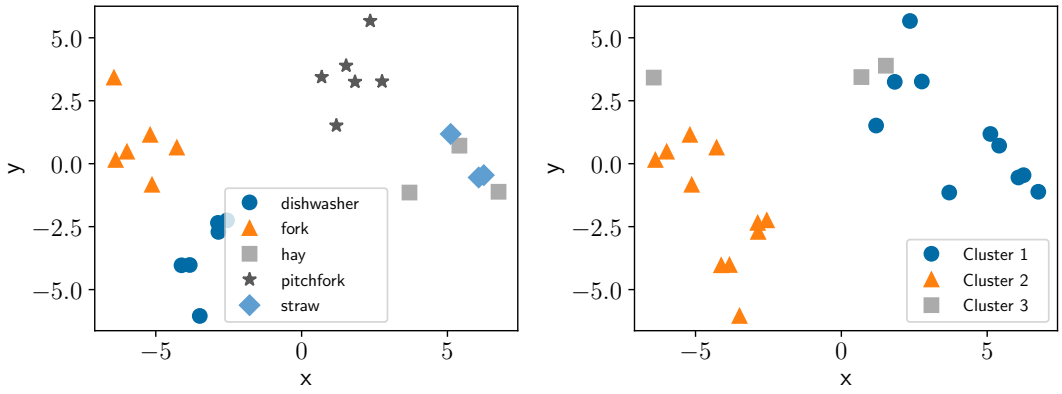
The data we will use was collected by Schönmann in her bachelor thesis. Schönmann originally constructed the dataset to investigate how the formulation of a triplet question the perception of a person. It consists of the triplet data obtained from multiple participants under different triplet questions and image sets. To collect the data, the observers were presented with three images randomly drawn from the a specific set of images, and together with one out of four different questions. The *odd* question, which we analyse in our setting is phrased: *Which is the odd one out?*. As explained above, if out of a, b, c, a is the odd one out, we transform this into two triplets (b, c, a) and (c, b, a). Out of the three image sets, we will study the so-called thematic image set more closely. It consists of 5 classes which can be roughly divided into two themes: barn (straw, hay, pitchforks) and kitchen (forks, dishwashers).

For each observer, image set and question combination, the data set consists of 462 unique triplets. Observer 2 has repeated the experiment again over a month later, but we have decided not to include those triplets to stay faithful to the original evaluation.

5.2 Applying Tangles

In the following, we will make an example of how to apply tangles to the obtained triplet data. As there is a lot of data present, it is not feasible to repeat our evaluations for all possible data points. We will choose one particular example to step through rigorously, and then we will quickly compare it to one other example. For our evaluations, we will use data points with the odd-one out triplets, as these have been reported in the work by Schönmann as having a higher embedding accuracy. We will also use the thematic image set, as Schönmann has reported embeddings that can be cleanly separated into different categories.

We will start out with observer 2. We have first reproduced the results by Schönmann by embedding the data with SOE into two dimensions (see Figure 5.1a). One can linearly separate two sets of clusters, which correspond to a divide between kitchen objects (dish-washer, fork) and barn objects (hay, straw, pitch fork). We have then applied majority tangles with an agreement of 3 and a radius of 1 (a lower radius produced only or two clusters) to the triplets and obtained a clustering. These cluster labels have been visualised in Figure 5.1b) using the previously obtained embedding. Care must be taken: the embedding from SOE is not a ground-truth embedding and just serves as a visualisation aid. If two clusters in the tangles-visualisation are far apart, this does not mean that the clustering is wrong – it could just as well be that the embedding is wrong or simply cannot capture the cluster structure appropriately. This is similar to how T-SNE can produce very deceptive embeddings of data.



(a) Original classes **(b)** Tangles clusters

Figure 5.1: We have embedded the obtained odd-one-out triplet data from observer 2 on the thematic image set. On the left, we see the original classes, on the right we see the predictions that we received by applying majority tangles with an agreement of 3 and a radius of 1.

We can directly see that the clustering from tangles also produces a similar divide between kitchen (orange triangles) and barn (blue circles). However, we see a third cluster structure (grey squares), which is a mix between two pitch forks and a kitchen fork. When we look at these items (depicted in Figure 5.2a), we can notice that they look more dissimilar to their counterparts in the kitchen or barn cluster. The fork (left in Figure 5.2a) has a design that is more reminiscent of pitch forks, and the two pitch forks look more clean than the other pitchforks in Figure 5.2c (no dirt on them, not depicted lying in grass). Thus, it can make sense that these three items are judged as more similar to each than their counterparts in the kitchen and barn cluster and thus get put into a separate clustering. Interestingly, that is an insight that could not be reached from the SOE embedding alone and might provide valuable information for a researcher.

Next, we want to see how one might use the hierarchy reconstruction and the explanatory power that tangles provides. For this, we first plot the hierarchy that we receive from the tangles algorithm in Figure 5.3. To get a better idea of how the hierarchies look like, we plot the soft clustering at each node, which can be seen in Figure 5.4. Here, each cut corresponds to one node of the hierachy and we can make out, which clusters belong to which nodes. As expected, we first see a coarse, thematic divide between kitchen and barn (nodes 18T, 0F, see Figure 5.4a, with the clean-looking pitchforks being placed together with the kitchen cluster. We then see a finer clustering of the kitchen cluster (node 0F), which is split into a set of various kitchen items (node 19F), and the cluster of special forks (node 4F). The fact that the special forks belong to the coarse kitchen cluster and only get separated off in a later step could be interpreted as the participant thinking of the clean-looking pitch forks as thematically belonging more in a kitchen than a barn. This is an insight that we couldn't have gotten from the ordinal embedding alone, highlighting a possible strength of tangles.

(a) Third cluster of different forks (gray squares)

(b) Other kitchen forks

(c) Other pitch forks

Figure 5.2: The images of all forks that are present in the thematic data set. In a), we have put the pitch forks and forks that landed in the gray squares cluster in the tangles clustering (see Figure 5.1), which contained a mixture of kitchen and barn items. In b), we have depicted all other kitchen forks (all contained in the orange triangle cluster associated with kitchen items) and in c) we have depicted all other pitch forks (contained in the blue circle cluster associated with barn items).

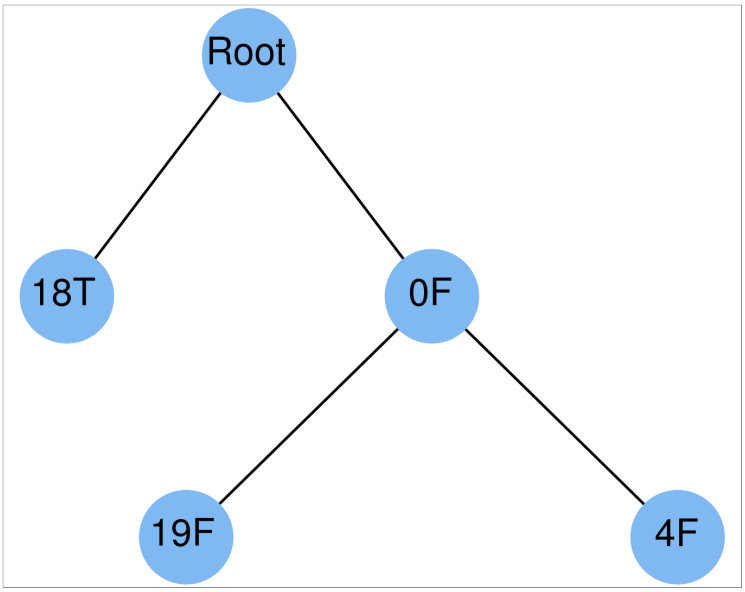


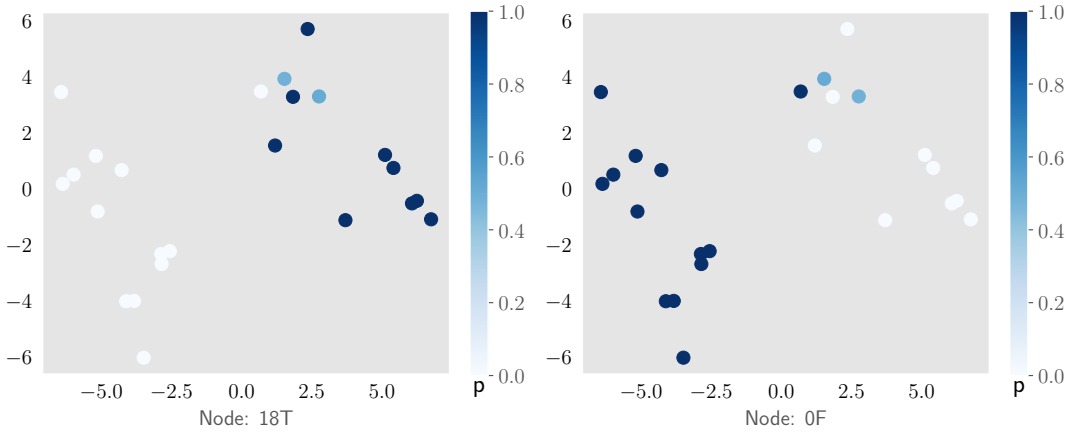
Figure 5.3: Here, we have plotted the hierarchy we receive from the tangles algorithm. The hierarchy starts at the root and each other node represents a splitting cut, which separates the data further. The label of the node is the number of the cut that caused the split to happen, together with the orientation (T means we have the cut in it's original arrangement, F means we have inverted the cut). This label can be used to identify the node in later processing steps.

The last step showcased the strength of the hierarchical clustering of tangles. Next, we want to show how we can use tangles to explain the clustering: why does a particular image belong to the kitchen or barn cluster? For this, we can look at the characterizing cuts of the clusters. As a reminder, if we look at a certain splitting node, its characterizing cuts are those that are always oriented in the same direction in the left subtree and in the other direction in the right subtree.

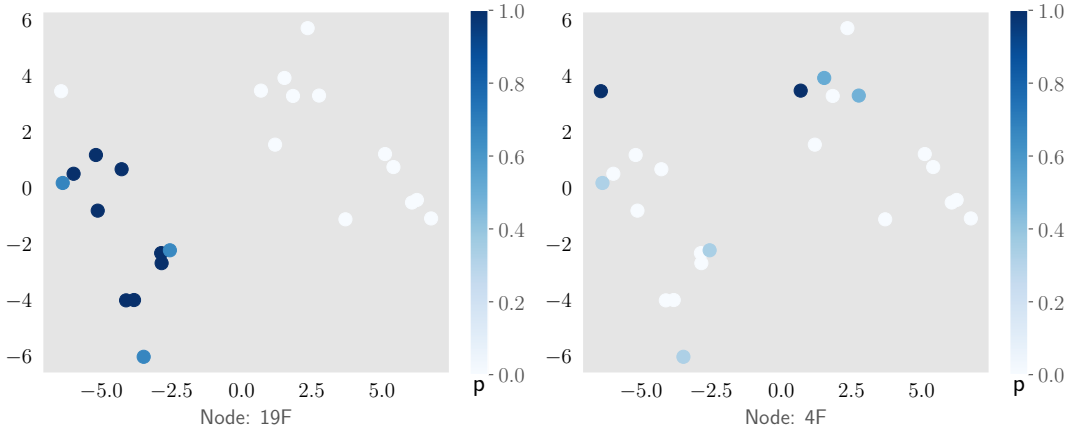
We have visualised the characterising cuts in Figure 5.5, together with the images that induced the particular cuts. As our cuts are interpretable (a majority cut with anchor point a contains points that are in some sense close to a), we can directly use this interpretation for our clustering. As we can see in Figure 5.5a, the items that are in the barn cluster have landed there because they are similar to the two straw/hay images that we have plotted in Figure 5.5b. This is similar to our intuition, straw and hay intuitively belong in a barn setting and definitely not in a kitchen, while the pitchforks might be a bit more ambiguous.

Our interpretation is that the items that have landed in the cluster of other kitchen items are there because they are similar to the dishwashers we have plotted in Figure 5.5d. This also makes sense, as we would interpret the dishwashers to very clearly belong into a kitchen environment, while the forks might be more ambiguous.

Overall, there is some small caveat to our explanation: In a majority cut, we only have a satisfying explanation in one direction: We know that if a point b is in the majority cut that has anchor point a, then b is close to a. However, the reverse direction might be a bit unsatisfying: If a point c is not in the majority cut, we know that it is not close to a. If we want to know how the cluster of forks and pitchforks formed, saying that they are dissimilar to the dishwashers plotted in Figure 5.5d is not a really strong argument, we would rather prefer to know that they are maybe similar to a certain other item. To remedy this, we might want to use more interpretable cuts. If we would have sampled the data in a landmark format, we would have been able say something like: a is in a certain cluster because it is closer to b than to c.



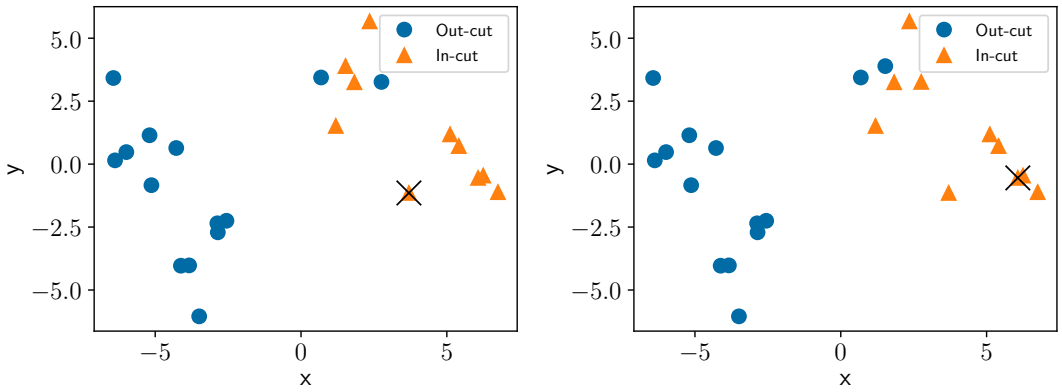
(a) Coarse cluster (barn - kitchen)



(b) Fine cluster (kitchen items - special forks)

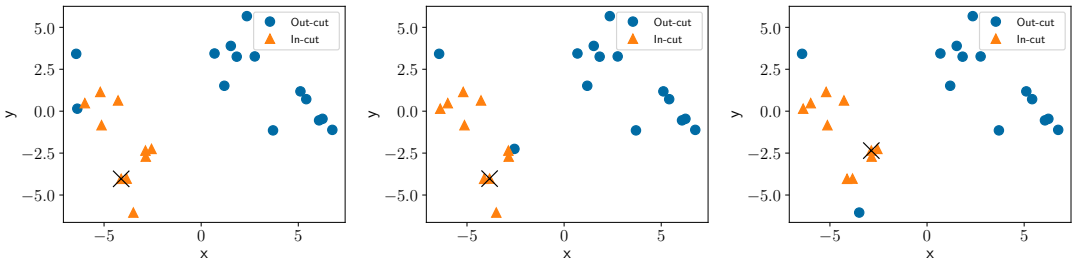
Figure 5.4: We have plotted the soft cuts that correspond to the nodes in Figure 5.3. We plot the SOE-embedding that we have calculated on the triplet data as a visualisation aid. Each cut corresponds to a node and thus to a cluster of a group of clusters. The color of a point corresponds to the probability that the point belongs to the given cluster(s), the darker, the more probable. In a), we see the coarse clustering that corresponds to the nodes directly below the root. On the left, we see the barn cluster, on the right the kitchen cluster. In b), we see the clusters that node 1F (the kitchen cluster) is made of. On the left, we have most of the kitchen items, on the right, we have the cluster of ktichen forks that look a bit more like pitchforks together with the clean-looking pitch forks (see Figure 5.2 and the corresponding discussion).

Next, to show that the last evaluation wasn't just a particular edge case, we are going to repeat some of the basic evaluations for another observer. We have selected the odd triplets and the thematic image set for person 3 and expect to see a similar behaviour than the one for person 3. In Figure 5.6 we have plotted the embedding from Schönmann again together with the tangles clustering. This time, we see three clusters (hay-straw, pitchforks and dishwashers-forks). These clusters coincide nicely with our classes, aside from one fork being clustered together with the pitchforks. We note that this is not the fork from Figure 5.2a that was clustered together with the pitchforks, so this might be a missclassification. If we look at the hierarchy (not plotted for space reasons) this time, we see that the pitchforks first get split off, and then the straw-hay from the kitchen dishwashers-forks cluster. This could again lead to interesting conclusions, which should probably be discussed by someone with more expert knowledge in the field, possibly using other evaluation data.



(a) Characterising cuts coarse cluster (kitchen-barn)

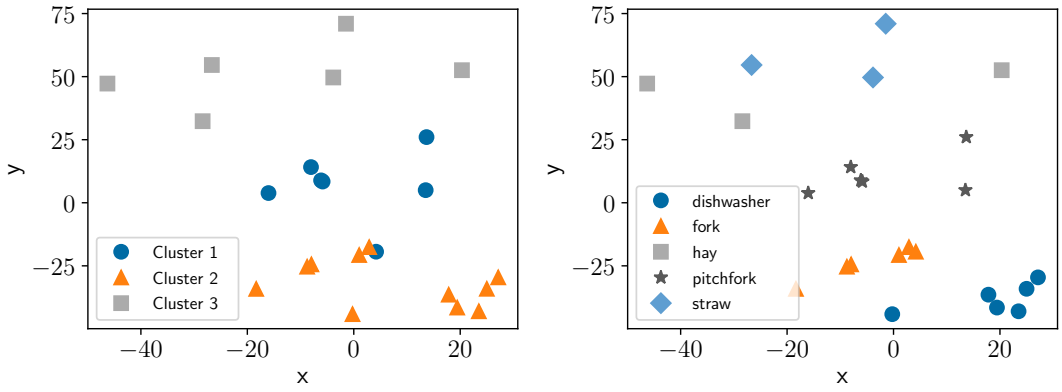
(b) Images of data points point inducing characterising cuts for coarse cluster



(c) Characterising cuts fine cluster (kitchen items - special forks)

(d) Images of data points inducing characterising cuts for fine cluster

Figure 5.5: We have plotted the characterising cuts of our splitting nodes. We draw only the orientation that corresponds to the left subtree, the orientation for the right one would just be all cuts reversed. This means, if a datapoint is often contained in the cuts that we depicted above, it is placed in the left subtree with high probability. In a), we plotted the characterising cuts for the root node (coarse split) and in c) have plotted the characterising cuts for the fine split (node 0F). As these cuts come from our original set of majority cuts, we have marked the data point that induced the particular cut with a black X. Below the characterising cuts in b) and d), we have plotted the images of the points that induced them.



(a) Original classes **(b)** Tangles clusters

Figure 5.6: Analog to Figure 5.1, on the left we plot the prediction of majority tangles with agreement 3 and radius 1 over a 2-dimensional SOE-embedding. On the right, we plot the original classes.

5.3 Discussion

In this chapter, we have shown that tangles can be used as an additional tool in evaluating triplet experiments, providing valuable insights. In particular, the more flexible clustering can show new dependencies between data points than the euclidean embedding by SOE, as we could make out a cluster of forks being perceived differently by observer 2. The hierarchy provided by tangles can help put these new dependencies into a better perspective, allowing us to gather that the pitch forks in the special fork cluster were perceived to belong more in a kitchen setting than in a barn setting. The explanations by tangles were used to make out the more defining items of a cluster (straw/hay for a barn, dishwashers for a kitchen). To our knowledge, hierarchical and explainable methods have not been used in psychophysics to this date, allowing tangles to eventually fill a gap.

However, we think that not all of the potential of tangles could be showcased here. As shown in our simulations, L-Tangles performs much better than M-Tangles on all data sets. Also, L-Tangles can provide explanations that are more intuitive than those from M-Tangles, as the cuts are inherently more explainble. A very interesting addition to our research would be applying tangles to our *ideal* real-world data set. This means that we have objects that exhibit a cluster structure, and that we have triplets sampled in a landmark format. To our knowledge, such a data set does not exist yet.

Chapter 6

Conclusion

In this work, we have demonstrated a suitable extension for the tangles algorithm to apply it to clustering and hierarchy reconstruction. We have validated its properties on simulated and experimental data under different external circumstances.

As we have compared different algorithms in our experiments, we can also give some recommendations as when to use which clustering algorithm. Especially, we can make out some conditions where tangles could provide a substantial benefit over others.

First of all, if the goal is explainability, tangles can be a great choice. We could not find any other clustering algorithm that works with triplet data directly, which produces explainable clusterings. An alternative could be using an explainable algorithm such as *explainable k-Means* (Moshkovitz et al., 2020) on the embedding created by an ordinal embedding directly, but this was out of the scope of this work. Explainable algorithms take up a more and more important role, especially with a *right to explanation* shifting more into the focus of policy makers and legal scholar (Selbst and Powles, 2017).

For standard clustering, we have observed a very good overall performance of landmark tangles, provided the triplets were sampled in a landmark-approach. If data is already present in this format, we can recommend tangles as a clustering method, as it often showed better performance than the state-of-the-art algorithm SOE. This is especially true in the low-triplet regime. If a new experiment is to be designed, the experimentator might think about whether it is feasible to sample data in a landmark format to utilize tangles, especially if its other properties (explainability and hierarchy reconstruction) are desired. If the data is expected to be very noisy however, SOE might still outperform tangles.

If the data is not in landmark format, we would only recommend using tangles (majority tangles in this case), if explainability and/or reconstructing hierarchies are desired, as SOE outperforms majority tangles in all our

simulated experiments.

In the case of hierarchy reconstruction, we can give a clear recommendation of tangles. We have seen that landmark tangles performs best out of our evaluated algorithms, and majority tangles come directly afterwards at around the performance of soft ordinal embedding combined with average linkage. Problematic can be that tangles cannot construct a true dendrogram, which is what the hierarchical clustering literature focusses on (such as in Ghoshdastidar et al. (2019)). For practical applications however, this can be good enough.

We have also seen that despite the setting not being optimal (no landmark triplets), we could still receive good practical results with majority tangles in the setting of psychophyics, confirming and expanding on the insights we have gained through an ordinal embedding. We envision that tangles could also be used as an additional tool supplementing an evaluation done via an ordinal embedding.

Through our work, we have also uncovered some problems and potential new research directions for clustering triplet data with tangles. The tangles framework by **?** is very flexible and has multiple areas to work on. As we have demonstrated, changing how to preprocess triplets to cuts is an effective method. Further work could explore other ways of doing this preprocessing step, which is still lacking for non-landmark triplets. A part of the tangles algorithm that we have not touched is the cost function. One could imagine different cost functions than the one we used, for example calculating some type of cost on the cuts using the triplet information. Additionally, one could think about modifying parts of the tangles framework themselves, which could maybe help with the problem of hierarchy noise we encountered in subsection 4.3.4.

A big missing piece is still the performance of landmark tangles on real data. So far, we could not find any data sets that exhibit a useable cluster structure, and consisted of landmark triplets. This data could be easily generated in a controlled environment, for example comparable to the approach used in Schönmann (2021), evaluated and compared to ordinal embeddings.

Bibliography

S Agarwal, J Wills, L Cayton, G Lanckriet, D Kriegman, and S Belongie. Generalized Non-metric Multidimensional Scaling. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

J Anderton and J Aslam. Scaling Up Ordinal Embedding: A Landmark Approach. *International Conference on Machine Learning (ICML)*, 2019.

S Balakrishnan, M Xu, A Krishnamurthy, and A Singh. Noise Thresholds for Spectral Clustering. *Neural Information Processing Systems (NeurIPS)*, 2011.

R Diestel. Tangles in the social sciences. *arXiv:1907.07341 [physics]*, 2019.

N Ghosh, Y Chen, and Y Yue. Landmark Ordinal Embedding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

D Ghoshdastidar, M Perrot, and U von Luxburg. Foundations of Comparison-Based Hierarchical Clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

S Haghiri, P Rubisch, R Geirhos, F Wichmann, and U von Luxburg. Comparison-Based Framework for Psychophysics: Lab versus Crowdsourcing. *arXiv preprint arXiv:1905.07234*, 2019.

S Haghiri, F. A Wichmann, and U von Luxburg. Estimation of perceptual scales using ordinal embedding. *Journal of Vision*, 20(9):14–14, 2020.

L Jain, K. G Jamieson, and R. D Nowak. Finite Sample Prediction and Recovery Bounds for Ordinal Embedding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

M Kleindessner and U von Luxburg. Kernel functions based on triplet comparisons. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

M Kleindessner and U von Luxburg. Lens Depth Function and k-Relative Neighborhood Graph: Versatile Tools for Ordinal Data Analysis. *Journal of Machine Learning Research*, 18(58):1–52, 2017.

L. van der Maaten and K. Weinberger. Stochastic triplet embedding. *IEEE International Workshop on Machine Learning for Signal Processing*, 2012.

S Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

M Moshkovitz, S Dasgupta, C Rashtchian, and N Frost. Explainable k-Means and k-Medians Clustering. *International Conference on Machine Learning (ICML)*, 2020.

B. D Roads and B Love. Enriching ImageNet with Human Similarity Judgments and Psychological Embeddings. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

N Robertson and P. D Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

I Schönmann. *Similarity Judgements of Natural Images: Instructions Affect Observers' Decision Criteria and Consistency*. Bachelor thesis, Universität Tübingen, 2021.

A. D Selbst and J Powles. Meaningful information and the right to explanation. *International Data Privacy Law*, 7(4):233–242, 2017.

R. N Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962.

N Stewart, G. D. A Brown, and N Chater. Absolute identification by relative judgment. *Psychol Rev*, 112(4):881–911, 2005.

O Tamuz, C Liu, S Belongie, O Shamir, and A. T Kalai. Adaptively learning the crowd kernel. *International Conference on Machine Learning (ICML)*, 2011.

Y Terada and U Luxburg. Local ordinal embedding. *International Conference on Machine Learning (ICML)*, 2014.

A Ukkonen. Crowdsourced Correlation Clustering with Relative Distance Comparisons. *International Conference on Data Mining (ICDM)*, 2017.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift