Comparison-based methods in machine learning

Comparison-based methods in machine learning

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

M.Sc. Siavash Haghiri

Tübingen

2019

Tag der mündlichen Qualifikation:	04.07.2019
1. Berichterstatter:	Prof. Dr. Ulrike von Luxburg
2. Berichterstatter:	Prof. Dr. Felix Wichmann

Abstract

This thesis focuses on developing and analyzing novel algorithms for a new setting of machine learning, which we call the "comparison-based" setting. There has been a plethora of methods and analysis over the last decades concerning machine learning tasks for datasets that either have a Euclidean representation or a distance/dissimilarity matrix between pairs of items in the dataset. In contrast to this traditional setting, we consider the setting in which no Euclidean representation of the items is available. We consider a general setting where items come from an arbitrary metric space $(\mathcal{X}, \delta)$. We have no access to the metric function δ, however, we can ask an oracle **triplet comparisons** of the form: "Is item x closer to item y or to item z?" This implies that the oracle can make a judgment on the following distance comparison:

$$\delta(x,y) \overset{?}{>} \delta(x,z).$$

In Chapter 1, the introductory chapter, we elaborate on various aspects of the comparison-based setting. We emphasize on the importance and usefulness of the comparison-based methods, by enumerating numerous applications. We also provide an intuitive example of a machine learning task performed with triplet comparisons. We briefly discuss the few existing approaches to perform machine learning in the comparison-based setting.

The contributions of the thesis are arranged in two parts. In the first part, Chapters 2 and 3, we are concerned with the development and analysis of algorithms which perform common machine learning tasks using only the answers to triplet comparisons. In Chapter 2, we propose a tree structure for the nearest neighbor search problem, using only triplet comparisons. We prove that if the metric space satisfies certain expansion conditions, then with high probability the height of the tree is logarithmic in the number of input items, leading to efficient search performance. We also provide an upper bound for the failure probability to return the true nearest neighbor. Chapter 3 extends the idea of constructing trees for nearest neighbor search to building an ensemble of trees as random forests for classification/regression tasks. We prove the consistency of a simplified version of the proposed trees leading to the consistency of the random forest. In addition, we demonstrate the desirable performance of the proposed random forest through comprehensive experiments in different settings.

In the second part of the thesis, Chapters 4 and 5, we apply the comparison-based approach to the scaling problem in psychophysics. The goal of psychophysical scaling is to find the functional relation between a physical stimulus and the perception of a human observer. The relative feedback in the form of triplet comparisons is not new in psychophysics. However, the approaches based on triplet comparisons are very limited in the psychophysics literature. In Chapter 4, we apply ordinal embedding methods, proposed in the machine learning literature, to psychophysical scaling. We demonstrate the satisfactory performance of ordinal embedding using extensive simulations and a real-world psychophysical scaling experiment. Chapter 5 examines the potential capability of crowdsourcing platforms for psychophysical scaling tasks using the comparison-based setting. Traditionally, the experiments of psychophysics are conducted in well-controlled Lab conditions. We run a representative experiment of psychophysics in both our psychophysics Lab and Amazon's Mechanical Turk (MTurk) platform. We compare the quality of the collected triplet comparisons from the Lab and the MTurk platform. The results show that the MTurk platform produces data which has slightly lower — however acceptable — quality.

Kurzfassung

Diese Dissertation befasst sich mit der Entwicklung und Analyse von Algorithmen in einem neuen Ansatz des maschinellen Lernens, den wir “vergleichbasiertes” Lernen nennen. In den letzten Jahrzehnten gab es eine Fülle von Methoden und Analysen für Datensätze, die entweder eine euklidische Darstellung oder eine Distanz zwischen den Objekten besitzen. Im Gegensatz zu dieser traditionellen Herangehensweise betrachten wir den Fall, wenn keine euklidische Darstellung der Objekte gegeben ist. Wir sagen, dass im Allgemeinen die Objekte aus einem beliebigen metrischen Raum $(\mathcal{X}, \delta)$ stammen. Wir haben allerdings keinen direkten Zugriff auf die Metrik δ, und können lediglich von einem Orakel einen Tripletvergleich abfragen: “Ist das Objekt x näher an Objekt y oder an Objekt z?”. Wir nehmen an, dass das Orakel auf diese Weise den folgenden Vergleich entscheidet:

$$\delta(x,y) \overset{?}{>} \delta(x,z).$$

In der Einführung in Kapitel 1 gehen wir auf die verschiedenen Aspekte des vergleichbasierten Lernens ein. Wie wichtig und nützlich vergleichsbasierte Methoden sind, zeigen wir anhand mehrerer Anwendungsfälle. Außerdem geben wir ein intuitives Beispiel für eine Machine Learning-Aufgabe, die mit Tripletvergleichen durchgeführt wird. Wir diskutieren die wenigen bisherigen Ansätze, vergleichsbasiertes maschinelles Lernen durchzuführen.
Diese Arbeit ist in zwei Teile aufgeteilt. Im ersten Teil, den Kapiteln 2 und 3, befassen wir uns mit der Entwicklung und Analyse von Algorithmen, die klassische Aufgaben des maschinellen Lernens durchführen, wobei *nur* die Antworten auf die Tripletvergleiche verwendet werden. In Kapitel 2 schlagen wir eine Baumstruktur für die Suche nach dem nächsten Nachbarn vor, wobei nur Tripletvergleiche verwendet werden. Wir beweisen, dass, wenn der metrische Raum bestimmte Expansionsbedingungen erfüllt, die Höhe des Baumes mit hoher Wahrscheinlichkeit logarithmisch in der Anzahl der Eingabeelemente ist, was zu einer effizienten Suchleistung führt. Wir liefern auch eine obere Schranke für die Wahrscheinlichkeit, den falschen nächsten Nachbarn zurückzugeben. Kapitel 3 erweitert die Idee dieses Suchbaumes auf den Bau mehrerer Bäume als Random Forests für Klassifizierungs-/Regressionsprobleme. Wir beweisen die Konsistenz einer vereinfachten Version der Bäume, die zur Konsistenz des Random Forest führt. Darüber hinaus zeigen wir die erwünschte Leistung des vorgeschlagenen Random Forests durch umfassende

Experimente in verschiedenen Szenarios.
Im zweiten Teil der Arbeit, den Kapiteln 4 und 5, wenden wir den vergleichsbasierten Ansatz auf das Skalierungsproblem in der Psychophysik an. Das Ziel der psychophysikalischen Skalierung ist es, die funktionelle Beziehung zwischen einem physischen Reiz und der Wahrnehmung desselben Reizes für einen menschlichen Beobachter zu finden. Tatsächlich ist das relative Feedback in Form von Tripletvergleichen in der Psychophysik nicht neu. Die vorgeschlagenen Ansätze zur Nutzung der Tripletvergleiche sind in der psychophysikalischen Literatur jedoch sehr begrenzt. In Kapitel 4 wenden wir ordinale Einbettungsmethoden, die in der Literatur des maschinellen Lernens vorgeschlagen werden, auf das Skalierungsproblem der Psychophysik an. Wir demonstrieren die erwünschte Performance der ordinalen Einbettungen mit Hilfe umfangreicher Simulationen und eines realen psychophysikalischen Experimentes. Kapitel 5 untersucht das Potenzial von Crowdsourcing-Plattformen für psychophysikalische Skalierungsaufgaben in einem vergleichsbasierten Szenario. Traditionell werden die Experimente der Psychophysik unter gut kontrollierten Laborbedingungen durchgeführt. Wir führen ein repräsentatives Experiment der Psychophysik sowohl in unserem Psychophysik-Labor als auch auf der Mechanical Turk (MTurk) Plattform von Amazon durch. Wir vergleichen die Qualität der gesammelten Tripletvergleiche aus dem Labor und der MTurk-Plattform. Die Ergebnisse zeigen, dass die MTurk-Plattform Daten produziert, die eine etwas geringere - aber akzeptable - Qualität aufweisen.

Acknowledgments

First of all, I would like to thank my supervisor Prof. Dr. Ulrike von Luxburg for her thorough support of my PhD study, her patience, her motivating attitude, and her in-depth knowledge. Throughout the four years of my PhD studies, she was always open to listen to my ideas and guide me with her opinions. Besides my supervisor, I am grateful to my co-advisor Prof. Dr. Felix Wichmann for his profound help in my PhD studies and research. Moreover, I thank my second co-advisor Prof. Dr. Philipp Berens for the valuable discussions.

I thank my colleagues, for the fruitful discussions we had, their technical and personal support, and for all the fun activities we had together. Particularly I thank my colleagues and co-authors Debarghya Ghoshdastidar and Damien Garreau. In addition, I would like to thank Matthäus Kleindessner, Lennard Schulz, Leena Chennuru Vankadara, Michael Lohaus and Michael Perrot.

I acknowledge the funding received towards my PhD from the German Research Foundation DFG (SFB 936/Z3). This thesis is also supported by the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

Last but not least, I would like to thank my parents and my sister for their spiritual support and encouragement.

Contents

Chapter 1

Introduction

In traditional settings of machine learning, data is given either in the form of points in a Euclidean space or a matrix of pairwise similarities or distances between points/items. Recently, comparison-based settings have become increasingly popular (Schultz and Joachims, 2003; Agarwal *et al.*, 2007; van der Maaten and Weinberger, 2012; Amid and Ukkonen, 2015; Ukkonen *et al.*, 2015; Balcan *et al.*, 2016). Here the assumption is that points come from some metric space $(\mathcal{X}, \delta)$, but the metric δ is unknown. We only have indirect access to the metric through distance comparisons. There exists a variety of ways to ask distance comparisons. However, the most common types of distance comparisons are as follows:

1 - Triplet comparisons: for a triplet of items (x,y,z) one can ask whether the following inequality is true or false.

$$\delta(x,y) \leq \delta(x,z) \tag{1.1}$$

In other words, the triplet comparison asks: Which of the items — y or z — is more similar to the item x?

2 - Quadruplet comparisons: for a quadruplet of items (x,y,z,t) one can ask whether the following inequality is true or false.

$$\delta(x,y) \leq \delta(z,t) \tag{1.2}$$

The quadruplet comparison can be rephrased as the following question: Is the distance between x and y smaller or greater than the distance between z and t?
Throughout this thesis we use the terms triplet comparison (respectively quadruplet comparison) and triplet question (respectively quadruplet question) interchangeably. Unless it is explicitly mentioned, the assumption throughout the whole thesis is that we have no access to any sort of representation for the items. The exact distance between pairs of items is also not available. The only information that is available is through answers to some of the triplet (or quadruplet) comparisons.

Even though the quadruplet comparisons seem to characterize a more general framework, there are a couple of reasons that make the triplet comparisons more favorable. First, if the feedback is gathered from humans, the triplet question is always easier to answer. In the triplet question, there exists a pivot item that the observer can compare with the two choices. Secondly, it is shown that having access to triplet answers carries the same amount of information as the quadruplet answers to construct a Euclidean embedding (Arias-Castro *et al.*, 2017). The number of triplets is one order of magnitude smaller than quadruplets, and thus one can gather the same information with less queries. Considering the benefits of the triplet comparisons, this thesis is mainly focused on the triplet comparisons.

Our contribution in this thesis is twofold:

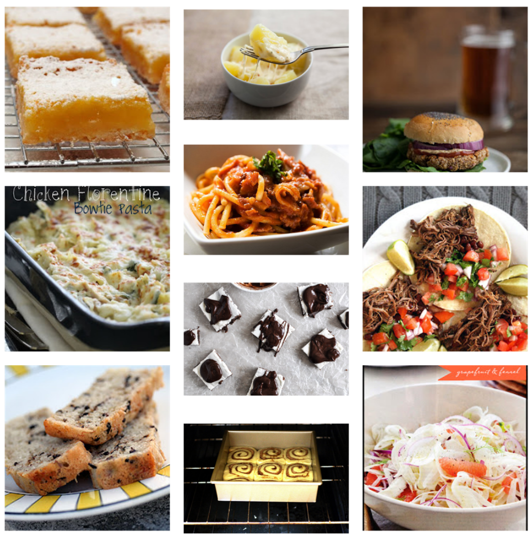
1) **Performing machine learning tasks in the comparison-based setting:** We propose and analyze methods for general machine learning tasks (classification and regression) in the comparison-based setting. We assume that neither a vector space representation nor a distance matrix of items is given. We only have access to an oracle who answers triplet comparisons.

2) **Application of the comparison-based approach in psychophysics**: We apply the comparison-based approach on practical problems of psychophysics. Psychophysical scaling is a long-standing problem at the very heart of psychophysics (Marks and Gescheider, 2002). There exist a large body of research dealing with this problem. Recently, the comparison-based approach is applied to a restricted version of the problem (Maloney and Yang, 2003). We demonstrate the efficiency of comparison-based methods, proposed in the machine learning literature, for general psychophysical scaling tasks.

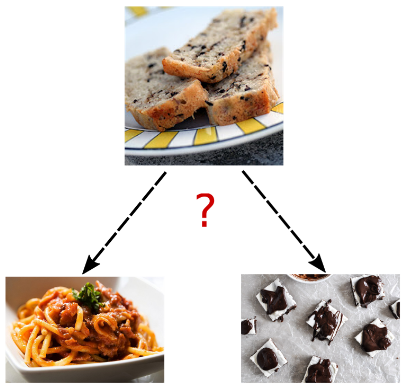
We elaborate on the contributions of the thesis at the end of this chapter. In the rest of this chapter, we first provide an example of the comparison-based approach in Section 1.1. Then, we discuss some general aspects of the comparison-based approach in Sections 1.2 and 1.3. Next, we briefly discuss readily available approaches for machine learning tasks in the comparison-based setting in Section 1.4. Section 1.5 lists the available datasets in the comparison-based setting. Finally, we present an overview of the contributions of the thesis in Section 1.6.

1.1 An example of a machine learning task in the comparison-based setting

In this section, we present a simple illustrative example of a machine learning task in the comparison-based setting. Suppose that we are given 10 food images belonging



(a)



(b)

Figure 1.1: (a) A sample of 10 images from the food dataset. (b) An example of a triplet question made with three images of the food dataset.

to different categories: main dish, salads, desserts, etc. The goal is to cluster the set of food images based on their taste. Since the taste of food is a subjective feature, triplet comparisons are used to gather feedback from people. We use the *food dataset*, which is a dataset of triplet comparisons for 100 food images[1]. The answers to triplet comparisons are asked from the human observers (workers of Amazon's mechanical Turk platform). More details on the dataset are given in Section 1.5. We pick 10 images uniformly at random from the set of 100 food images as shown in Figure 1.1 (a).

As we restrict our experiment to 10 food images, we consequently consider only the triplet comparisons which are formed with the 10 images. There exist 136 triplet comparisons in the dataset associated with the chosen images. An example of such a triplet question is shown in Figure 1.1 (b). The final clustering that we suggest should be consistent with the answers to the triplet comparisons. Let assume a triplet answer suggests that item A is closer to item B than to item C. Then, intuitively speaking, the likelihood of item A sharing a cluster with item B is higher than the likelihood of item A being in the same cluster with item C.

In this example, we use the t-distributed stochastic triplet embedding (t-STE) algorithm to estimate a two-dimensional representation of the 10 images. The algorithm embeds the images into a two-dimensional Euclidean space such that the answers

[1]`https://vision.cornell.edu/se3/projects/cost-effective-hits/`

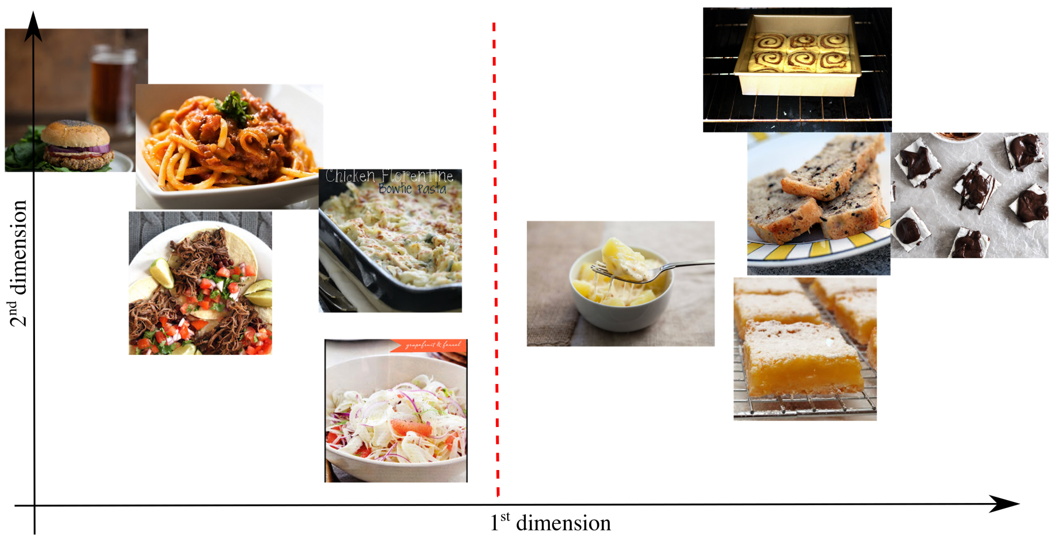


Figure 1.2: The embedding and clustering result of the example with 10 food images. The center of each depicted image corresponds to the two-dimensional embedding estimation of the image. The k-means algorithm clusters the set of images into two sets which is shown by the red dashed line in the middle.

to the triplet comparisons are satisfied. More precisely, the distances of estimated items in the two-dimensional space agree with the answers to the triplet comparisons. After gathering a Euclidean representation of food images in the two-dimensional space, one can run any clustering algorithm, such as the k-means algorithm, to cluster the images into two sets. Figure 1.2 shows the two-dimensional embedding and the clustering of the images into two sets, performed by the k-means algorithm. The clustering is shown by the dashed red line between the two clusters.
The results show meaningful categorization of the food images based on the triplet comparisons. In fact, the left cluster contains main dishes and one salad, while the right cluster contains desserts. Even inside the clusters the distances of images are meaningful. For instance in the left cluster, three food images with meat are at the very left, whereas the salad is located on their opposite side. Note that the embedding algorithm and the k-means clustering is one of the various possible approaches that can be used in the comparison-based setting. We give a brief review of general approaches (related work) later in this chapter.
In addition to the desirable performance of the comparison-based approach we would like to emphasize on some of the other benefits. A triplet question, such as the example in Figure 1.1 (b), is very convenient to answer for a human observer. If we ask the observer to describe the subjective features of the food with precise values, or to provide a similarity value for pairs of images, in either case the response is significantly harder compared to answering triplet questions. There also exists a

different bias for each observer, when we ask for a quantitative response. Eliminating the bias is an intricate task in itself.

1.2 When is the comparison-based setting helpful?

To motivate the study of comparison-based setting we specify some general cases in which the comparison-based approach is favorable. In addition, we believe that independent of the underlying applications, theoretical challenges posed by the setting alone merit further study and analysis.

- **Abstract data/ human feedback:** The comparison-based setting is particularly of interest where the items in the dataset are abstract and human responses are needed to describe them. There are studies, in various fields ranging from psychology to computer graphics, that suggest human beings are better in making relative judgments than providing quantitative (numerical) feedback (Miller, 1956; Stewart *et al.*, 2005; Demiralp *et al.*, 2014; Li *et al.*, 2016).

  The main alternative to the triplet comparisons, or relative judgments in general, is to ask quantitative responses from human observers. However, these responses can be very biased. This issue has been particularly investigated in the psychophysics literature. The Stevens' method of magnitude estimation is a well-known psychophysical scaling method based on direct quantitative responses from human observers (Stevens, 1957). In this method, observers are asked to describe the sensation of a physical stimulus with a numerical value (magnitude) such that ratios of stimulus intensities match the ratios of given magnitudes. Shepard observed the intrinsic bias in this method by introducing the "response transformation function " (Shepard, 1981). The response transformation function takes the sensation as input and produces the magnitude value. This function is unknown for a typical observer and may differ among the observers. Therefore, the numerical magnitude values can be biased. Using the relative judgments (such as triplet comparisons) can eliminate the effect of response transformation function (Torgerson, 1958; Maloney and Yang, 2003), leading to unbiased information.

- **Complex data structures:** The input data to machine learning algorithms can sometimes be very complex. Consider, for instance, cancer medications with big complex molecular structures. The molecular structure cannot be represented in Euclidean spaces. In addition, even if we represent molecules with graphs, it is still a cumbersome task to compare the graphs and come up with dissimilarity values between them. However, in many cases a clinician or chemist can provide the answer to a triplet comparison of complex medications based on her expertise in the field.

- **Implicit comparisons:** There are cases that the information is not in the form of triplet comparisons at the first glance, however, one can interpret them as triplet (or quadruplet) comparisons. An example is provided in Schultz and Joachims (2003): Consider the click-through rates of web-pages in a search engine. If three pages X, Y, Z are shown as the search results and X, Y have on average more click-through rates, one can deduce that page X is more similar to Y than to Z. Even though the click-through rates do not explicitly indicate a distance/dissimilarity between the pages, it is possible to interpret them as answers to triplet questions and consequently apply comparison-based methods.

1.3 Sampling triplet comparisons

If we consider a dataset of n items, there will be $3\binom{n}{3}$ possible triplet questions to query. Since the number of possible triplets grows very fast with the number of items, it is impossible in practice to query all the possible triplets. Moreover, a significant number of triplets contain redundant information. To see this fact, one can consider the complete ranking of pairwise distances. The full ranking of distances contains sufficient information to answer all triplet questions. There exist $\binom{n}{2}$ pairwise distances. In order to sort them, we require $O\left(n^2 \log n\right)$ comparisons. This amount is one order of magnitude smaller than the total number of triplets.
In addition, if the underlying metric space is a Euclidean space of dimension d, it is shown that having the answers to $O(nd \log n)$ triplet comparisons is sufficient to reconstruct the Euclidean representation of items (Jain *et al.*, 2016). Therefore, the number of informative triplets is actually significantly smaller than all triplets. A naturally arising question is "how should we pick the subset of triplets from the set of all possible triplets?" There are three general scenarios to query a subset of triplet comparisons:

- **Random triplet scenario:** The subset of triplet questions to query is sampled from the set of all valid triplet questions uniformly at random. The algorithm has no influence on the choice of items appearing in the triplet questions. In this way, the algorithm is provided with the answers to a subset of randomly chosen triplet questions. The main advantage of this scenario is that it does not require a constant interaction between the algorithm and the oracle.

- **Active triplet scenario:** We let the algorithm actively choose the triplet question to query. Triplet questions are queried consecutively and the algorithm can choose the indices of necessary triplets, actively, during the running process. This is the most straightforward scenario from the perspective of algorithm design. In other words, proposing machine learning algorithms is easier in this case. However, in practice it is more challenging to implement, as the algorithm needs constant interaction with the oracle which provides answers to the triplet questions.

- **Batch triplet scenario:** The algorithm can choose a batch (subset) of triplet questions to query. The answers to this batch of triplets is queried from the oracle. In this setting, one needs a one-time query from the oracle, which is less interaction comparing to the active triplet scenario. On the other hand, in contrast to the random triplet scenario, the choice of triplet questions is specified by the algorithm. Therefore, we can consider it as a trade-off between the two previous scenarios.

Our focus in Chapters 2 and 3 is on the algorithms working in the active triplet scenario. In the next two chapters, we apply comparison-based methods based on the random triplet scenario.

1.4 Existing approaches for comparison-based machine learning

Here we briefly discuss the existing approaches to the comparison-based setting in machine learning. As the field itself is quite new, only a few approaches are currently proposed in the literature. We cover two general approaches to the problem. Our approach to the problem is presented in Chapters 2 and 3.

1.4.1 Ordinal embedding

A natural approach in the comparison-based setting of machine learning is *ordinal embedding*. In this procedure, one seeks a Euclidean representation of abstract items which is consistent with the answers to the set of queried triplet (or quadruplet) comparisons. Since there is a vast literature of machine learning dealing with data in Euclidean spaces, the machine learning problem can be solved consequently with the methods designed for Euclidean spaces. We give a formal definition of the ordinal embedding problem and discuss one of the existing solutions to the problem in Chapter 4.
The ordinal embedding approach has a couple of advantages. First, the ordinal embedding problem is studied and analyzed comprehensively (Kleindessner and von Luxburg, 2014; Arias-Castro *et al.*, 2017) and there are a couple of algorithms

provided to solve the problem (Agarwal *et al.*, 2007; Tamuz *et al.*, 2011; van der Maaten and Weinberger, 2012; Terada and Luxburg, 2014). Secondly, the traditional algorithms of machine learning are readily applicable in the Euclidean space. Finally, in two or three dimensions, one can also visualize the output of the embedding which is favorable for visual inspection of algorithms.

In spite of the aforementioned benefits, the approach of ordinal embedding has some drawbacks which makes it less practical. First, the ordinal embedding problem requires one to solve a quadratic optimization problem which has $O\left(n^3\right)$ time complexity. In practice, even the fastest algorithms cannot embed more than 1000 items in a reasonable time. Secondly, the embedding dimension should be given as input to the ordinal embedding algorithm. In a general scenario items are abstract and the proper Euclidean dimension to embed the items is unknown. There exist limited studies on dimensionality estimation without exact distances (Kleindessner and Luxburg, 2015). Incorrect specification of dimension can significantly affect the result of embedding and consequently affect the performance of the machine learning task. Lastly, ordinal embedding finds a Euclidean configuration of the items which is not necessary for machine learning tasks such as classification or regression. A common inductive bias assumption of classification states that items in the same neighborhood tend to have the same label. Therefore, one only needs to identify the neighborhood of a query item to predict the label of the item.

1.4.2 Kernels based on triplet comparisons

Constructing a kernel matrix based on the triplet comparisons is an alternative solution for comparison-based machine learning tasks. Kernel methods are among the most popular methods in machine learning (Scholkopf and Smola, 2001; Bishop, 2006; Hofmann *et al.*, 2008). Building a kernel matrix based on the triplet comparisons allows us to use existing kernel-based methods. There is one recent study that proposes two approaches to produce kernel matrices based on comparison-based data (Kleindessner and von Luxburg, 2017). The proposed kernels are significantly faster than ordinal embedding methods. Moreover, they show desirable performance in some experimental cases. However, they still lack theoretical justification.

1.5 Datasets

The idea of gathering information via triplet comparisons is quite new in machine learning. Currently there exist few available datasets for this setting. Here, we list the available datasets:

- **Music dataset:** This dataset consists of 16,449 triplet comparisons on the similarity of 412 popular musicians (Ellis *et al.*, 2002). The triplets are queried via an online survey with 1,032 attending users. The raw dataset and a filtered version of the dataset are available at `https://labrosa.ee.columbia.edu/projects/musicsim/musicseer.html`.

- **Food dataset:** The dataset contains 100 images of food taken from the website `Yummly.com`; The dataset was initially used in a study to compare triplet comparisons with similar query types (Wilber *et al.*, 2014). The answers to 190376 triplet questions are asked from the workers of Amazon's mechanical Turk platform. This dataset is available at `https://vision.cornell.edu/se3/projects/cost-effective-hits/`

- **Nature and Vogue datasets:** The Nature dataset contains triplet comparisons of 120 natural images in thematic clusters (forest, open country, coast, mountain). The triplet question is not the same as the question we defined in Equation (1.1). For a triplet of items A, B and C, it asks: "which of the items is the outlier?" In other words: "which of the two items are close together and which item is irrelevant?" The Vogue dataset consists of the same triplet question of the 60 cover images of the UK edition of the Vogue magazine, chosen at random from the time interval 1970 to 2012. The answers to triplet comparisons are gathered from six human workers. The two datasets are used in Ukkonen *et al.* (2015); both datasets are available at `http://anttiukkonen.com/nature_and_vogue_triplets.zip`.

- **Car dataset:** This dataset contains distance comparisons of 60 car images. The distance comparison is not exactly in the form of triplet comparisons, which we defined in (1.1). For a triplet of items A, B and C it asks: "which of the items is the *most central*?" The central item should conceptually lie in between the two other items. The information that one gets with such a comparison is slightly different from that of triplet comparisons of the form (1.1). The dataset is available at `http://www.tml.cs.uni-tuebingen.de/team/kleindessner/60_cars_data.zip`

- **Eidolon dataset:** In Chapter 5, we present a psychophysics experiment designed to study human visual perception. Triplet questions are formed based on a set of 100 images generated by the Eidolon factory toolbox (Koenderink *et al.*, 2017). The answers to the triplet questions are obtained from 8 participants in a well-controlled situation in our psychopysics lab, as well as 60 workers of Amazon's mechanical Turk platform. Each lab participants gives answers to 6000 triplet comparisons, while each mechanical Turk worker answers 2000 triplet comparisons.

1.6 Overview of the contributions

- **Chapter 2:** As a stepping stone to solve machine learning problems, such as classification and regression, we propose an efficient algorithm for nearest neighbour search in the comparison-based setting. Machine learning problems can then be solved using the popular nearest neighbor methods (k-NN classifier, k-NN graph clustering and etc.).

  We introduce a data structure called "comparison-tree" for the nearest neighbor search based on triplet comparisons. We have two theoretical results concerning the comparison-tree. First, we prove that the height of proposed trees is in the order of $\log n$ with certain assumptions on the metric space. Consequently this leads to an upper bound for the required number of triplet comparisons. More precisely, $O(n \log n)$ and $O(\log n)$ triplet comparisons are required for comparison-tree construction and nearest neighbor search respectively. Secondly, we provide an upper bound for the failure probability to return the true nearest neighbor. In addition to the theoretical results, we conduct experiments to show the efficiency of comparison-trees in various settings. This chapter is based on Haghiri *et al.* (2017).

- **Chapter 3:** In this chapter, we propose a novel random forest algorithm for regression and classification that relies only on triplet comparisons. We use the comparison-trees (from the previous chapter) as building blocks, to make an ensemble of comparison-trees for classification and regression tasks. As a theoretical result, we establish sufficient conditions for the consistency of the proposed random forest. In a set of comprehensive experiments, we then demonstrate that the proposed random forest is efficient both for classification and regression. In particular, it is even competitive with other methods that have direct access to the Euclidean representation of the data. The results of this chapter has appeared in Haghiri *et al.* (2018).

- **Chapter 4:** This chapter is concerned with the application of comparison-based approaches in psychophysics. We consider a general (difference) "scaling problem" in psychophysics. Even though one can precisely measure physical stimuli (such as light or sound), the perception of physical stimuli is usually hard to measure and quantify. The (difference) scaling problem aims to find the relationship between the perceived magnitude and the physical stimulus. This problem is as old as the psychophysics itself and there exists a large body of research for the problem. The comparison-based approach is also applied to this problem in a very limited scenario (Maloney and Yang, 2003). We apply ordinal embedding methods, proposed in the machine learning literature, to various simulation scenarios and a real experiment of psychophysics.

We demonstrate the efficiency of the comparison-based approach for the psychophysical scaling problem.

- **Chapter 5:** The last chapter also deals with the application of comparison-based setting in the psychophysics. Traditionally, psychophysical experiments are conducted by repeated measurements on few participants under well-controlled conditions, often resulting in, if done properly, high quality data. In recent years, however, crowdsourcing platforms are becoming increasingly popular means of data collection, measuring many participants at the potential cost of obtaining data of worse quality. We study whether the use of comparison-based approach, combined with machine learning algorithms, can boost the reliability of crowdsourcing studies for psychophysics, such that they can achieve a performance close to a lab experiment. To this end, we compare three setups: simulations, a psychophysics lab experiment, and the same experiment on Amazon Mechanical Turk. All these experiments are conducted in a comparison-based setting where participants have to answer triplet questions. We then use ordinal embedding to solve the triplet prediction problem: given a subset of triplet questions with corresponding answers, we predict the answer to the remaining questions. Considering the limitations and noise on MTurk, we find that the accuracy of triplet prediction is surprisingly close—but not equal—to our lab study.

List of publications

This thesis is based on the following papers:

- Haghiri, S., Garreau, D., and von Luxburg U. (2018). Comparison-Based Random Forest. In *International Conference on Machine Learning (ICML).*
  Paper available at: `http://proceedings.mlr.press/v80/haghiri18a/haghiri18a.pdf`
  Code available at: `https://github.com/SiavashCS/CompRF`

- Haghiri, S., Ghoshdastidar, D., and von Luxburg, U. (2017). Comparison-Based Nearest Neighbor Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS).*
  Paper available at: `http://proceedings.mlr.press/v54/haghiri17a/haghiri17a.pdf`
  Code available at: `https://github.com/SiavashCS/CompTree`

- Haghiri, S., Wichmann, F., and von Luxburg, U. Estimation of perceptual scales by ordinal embedding. Manuscript in preparation.

- Haghiri, S., Wichmann, F., and von Luxburg, U. Comparison-based framework for psychophysics: Lab. vs crowdsourcing. Manuscript in preparation.

During the PhD program the author had major contribution to the following paper which is not immediately related to the thesis:

- Forlim, C.G., Haghiri, S., Düzel, S., Lindenberger, U., and Kühn, S. (2019). Efficient small-world and scale-free functional brain networks at rest using k-nearest neighbors thresholding. bioRxiv preprint, bioRxiv: 628453.

Chapter 2

Comparison-based nearest neighbor search

This chapter focuses on the nearest neighbor search problem in the comparison-based setting. Let us assume that S is a set of n items in some metric space $(\mathcal{X}, \delta)$. There is no representation given for the items, nor do we have direct access to the metric distance values. We can only ask the answers to triplet questions of the form (1.1). The goal is to find the approximate nearest neighbor of a query point q in the set S with the fewest number of triplet comparisons and the highest accuracy. To this end, we propose a nearest neighbor search tree structure which relies only on triplet comparisons.

Many algorithms for exact or approximate nearest neighbor search use data structures based on space partitioning. Most popular is the setting where points live in the Euclidean space. KD-Tree (Bentley, 1975), principal axis tree (PA-Tree) (McNames, 2001), Spill-Tree (Liu *et al.*, 2004), random projection tree (RP-Tree) (Dasgupta and Freund, 2008), and max-margin tree (MM-Tree) (Ram and Gray, 2013) are among the algorithms in Euclidean setting. In addition, the setting where data points only lie on a more abstract metric space has been explored considerably. Metric Skip List (Karger and Ruhl, 2002), Navigating Net (Krauthgamer and Lee, 2004), and Cover-Tree (Beygelzimer *et al.*, 2006) are some of the methods in this category. However, in nearly all these algorithms, we either need to know the vector representation of the points or the distance values between the points. Few exceptional methods exist that even work in the setting where we only have access to triplet comparisons (Goyal *et al.*, 2008; Lifshits and Zhang, 2009; Tschopp *et al.*, 2011; Houle and Nett, 2015).

To our taste, the most appealing data structure, applicable in the comparison-based setting, is what we call the comparison-tree. The structure has been introduced as "metric tree based on a generalized hyperplane decomposition" in Uhlmann (1991). For the sake of simplicity we refer to it as the comparison-tree. To partition the space into smaller subsets, two pivot points are picked uniformly at random from the given dataset. The space is then separated into two "generalized half-spaces", namely the two sets of points that are closer to the two respective pivots. This is done recursively

until the resulting subsets become smaller than a given size. Once this tree structure has been constructed, the nearest neighbor search proceeds by comparing the query point to the two pivot points and proceeding into the corresponding half-space. On an intuitive level, comparison-trees are promising: because (i) the splits seem to adapt to the geometry of the data, (ii) it seems that the splits are not extremely unbalanced, (iii) there is “enough randomness” in the construction. Unfortunately, none of these intuitions has been proved or formally investigated yet.
Our first contribution is to analyze the performance of comparison-trees in general metric spaces. Under certain assumptions on the expansion rate of the space, we prove that comparison-trees are nicely balanced and their height is of the order $\Theta(\log n)$. This means that to construct a comparison-tree, we only need of the order $n \log n$ many triplet comparisons. Corollary, a nearest neighbor query requires $\Theta(\log n)$ triplet comparisons. Moreover, we bound the probability that the nearest neighbor algorithm finds the correct nearest neighbor. Our second contribution consists of simulations that compare the behavior of comparison-trees to standard data structures in Euclidean spaces and other comparison-based data structures on metric spaces. We find that the comparison-tree performs surprisingly well even if compared to competitors that access vector representations in Euclidean spaces (KD-Tree, RP-Tree and PA-Tree), and favorably in comparison to one of the recent comparison-based algorithms proposed in Houle and Nett (2015).

2.1 Comparison-tree

To construct a comparison-tree on S, we proceed as follows (see Algorithm 1 below). The root of the tree $T.root$ consists of the whole set S, and each of the subsequent nodes represents a partition of the set S. In each step of the tree construction, the elements of the current node are partitioned into two disjoint sets, which in turn are the root nodes for the left and right sub-trees denoted by $T.leftchild$ and $T.rightchild$. More concretely, to form a partition of the current node of the tree, we randomly choose two pivot elements among its current elements, denoted by $T.leftpivot$ and $T.rightpivot$. Then we group the remaining elements according to whether they are closer to the left or right pivot. Observe that this step does not require actual distance values, but just triplet comparisons. Then we recurse, until the current set of elements has at most size n_0 for some prespecified n_0.
The computational complexity of the tree construction is governed by the number of triplet comparisons required in the procedure, which depends on the height of tree. In the next section, we show that under certain growth assumptions on the metric, the comparison-tree has height $h = O(\log n)$ with high probability.
To use the comparison-tree for (approximate) nearest neighbor search, we employ the obvious greedy procedure (sometimes called the defeatist search; see Algo-

Algorithm 1 *CompTree*(S, n_0): **Comparison-tree construction**

```
Input: $S \subseteq \mathcal{X}$, and maximum leaf size $n_0$
Output: Comparison-tree $T$
 1: $T.root \leftarrow S$
 2: if $|S| > n_0$ then
 3:     Uniformly sample distinct points $x_1, x_2 \in S$
 4:     $S_1 \leftarrow \{x \in S : \delta(x, x_1) \leq \delta(x, x_2)\}$
 5:     $T.leftpivot \leftarrow x_1,\ T.rightpivot \leftarrow x_2$
 6:     $T.leftchild \leftarrow CompTree(S_1, n_0)$
 7:     $T.rightchild \leftarrow CompTree(S \backslash S_1, n_0)$
 8: end if
 9:
10: return $T$
```

rithm 2): starting at the root, we compare the query element q to the current two pivot elements, and using a triplet comparison we decide whether to proceed in the left or right branch. When we reach a leaf, we carry out an exhaustive search among all its elements to determine the one that is closest to the given query. This step requires n_0 triplet comparisons. Overall, the nearest neighbor search in a comparison-tree of height h requires at most $h + n_0$ triplet comparisons, which boils down to $O(\log n)$.

Algorithm 2 NN(q, T): **Nearest neighbor search**

```
Input: Comparison-tree $T$, and query $q$.
Output: $\hat{x}_q$ = approximate nearest neighbor of $q$ in $S$
 1: if $T.leftchild \neq null$ then
 2:     if $\delta(q, T.leftpivot) \leq \delta(q, T.rightpivot)$ then
 3:         $\hat{x}_q \leftarrow \mathrm{NN}(q, T.leftchild)$
 4:     else
 5:         $\hat{x}_q \leftarrow \mathrm{NN}(q, T.rightchild)$
 6:     end if
 7: else
 8:     choose $\hat{x}_q$ s.t. $\delta(q, \hat{x}_q) \leq \delta(q, x)\ \forall x \in T.root$
 9: end if
10:
11: return $\hat{x}_q$
```

2.2 Theoretical analysis

In this section, we analyze the complexity and performance of the tree construction as well as the nearest neighbor search. We first provide a high probability bound on the height of the comparison-tree, which in turn bounds the number of triplet comparisons required for both tree construction and nearest neighbor search. In addition, we derive an upper bound on the probability with which the above approach fails to return the exact nearest neighbor of a given query.

2.2.1 Expansion conditions

Finding the nearest neighbor for a query point q in a general metric space $(\mathcal{X}, \delta)$ can require up to $\Omega(n)$ comparisons in the worst case, using any data structure built on the given set S (Beygelzimer *et al.*, 2006). Hence, most similarity search methods are analyzed under the natural assumption that the metric δ is growth-restricted (Karger and Ruhl, 2002; Krauthgamer and Lee, 2004). Informally, such restrictions imply that the volume of a closed ball in the space does not increase drastically when its radius is increased by a certain factor. Various related notions are used to characterize the growth rate of such metrics, for instance Assouad dimension (Assouad, 1979), doubling dimension (Gupta *et al.*, 2003), homogeneity (Luukkainen and Saksman, 1998), and expansion rate (Karger and Ruhl, 2002) among others. The analysis of most tree based search methods in the Euclidean setting requires only a doubling property of the metric (Dasgupta and Sinha, 2015). Such results hold for metric spaces with finite doubling dimension or Assouad dimension. When dealing with general metric spaces, it is more convenient to consider the expansion rate, which is an empirical variant of the doubling property defined for a given finite set $S \subseteq \mathcal{X}$. Typically, the analysis of data dependent tree constructions requires even stronger restrictions.
In this work, we use a slightly weaker variant of the strong expansion rate condition used in Ram and Gray (2013). Intuitively, we need bounds on the expansion rate for all the finite point sets that can possibly occur in the non-leaf nodes of the comparison-tree. Let $(\mathcal{X}, \delta)$ be a metric space, $S \subseteq \mathcal{X}$ and $n_0 < |S|$. We construct a collection $\mathcal{C}_S \subseteq 2^S$ as follows:

1. $S \in \mathcal{C}_S$,
2. If $A \in 2^S$ and there exist $x, y \in S$ such that $\delta(z,x) \leq \delta(z,y)\ \forall z \in A$, then $A \in \mathcal{C}_S$, $S \backslash A \in \mathcal{C}_S$,
3. If $A, B \in \mathcal{C}_S$, then $A \cap B \in \mathcal{C}_S$.

We finally remove all $A \in \mathcal{C}_S$ with size $|A| \leq n_0$. Observe that $\mathcal{C}_S$ characterizes the collection of all possible non-leaf nodes of the tree[1]. We define the **strong expansion rate** of S as the smallest $\tilde{c} \geq 1$ such that

$$|B(x,2r) \cap A| \leq \tilde{c}|B(x,r) \cap A| \tag{2.1}$$

for all $A \in \mathcal{C}_S$, $x \in A$ and $r > 0$, where $B(x,r)$ is the closed ball in $(\mathcal{X}, \delta)$ centered at x with radius r.
Inequality (2.1) states that every $A \in \mathcal{C}_S$ has an expansion rate at most $\tilde{c}$ similar to the definition in Karger and Ruhl (2002). This requirement for all $A \in \mathcal{C}_S$ is strong, but seems unavoidable due to the data dependent, yet random, tree construction.

2.2.2 Main results

The following theorem provides an upper bound on the height of the comparison-tree.

Theorem 1 (Height of a comparison-tree) *Consider a set S of size n in a metric space that satisfies the strong expansion rate condition with constant $\tilde{c}$. Fix some $n_0 \in \mathbb{N}$. Then for any $\varepsilon > 0$, with probability $1 - \varepsilon$, the comparison-tree construction algorithm returns a tree with height smaller than*

$$h^* = 3\log\left(\frac{e}{\varepsilon}\right) + 96\tilde{c}^2 \log\left(\frac{n}{n_0}\right) \tag{2.2}$$

We prove the theorem later in the section. Theorem 1 implies that if the expansion rate $\tilde{c} = O(1)$, then the height of the randomly constructed tree is bounded by $O(\log n)$ with high probability. In particular, one can expect this to happen if the set of points is sampled from an "evenly" spread distribution in a growth-restricted space $\mathcal{X}$. As a consequence of Theorem 1, one can comment on the number of triplet comparisons required for tree construction and nearest neighbor search. We state this in the following corollary, which is a simple consequence of Theorem 1.

Corollary 2 (Number of triplet comparisons) *For $\varepsilon > 0$, let h^* be the constant defined as in (2.2). Then with probability $1 - \varepsilon$, the comparison-tree construction algorithm requires at most nh^* triplet comparisons to construct the comparison-tree. Furthermore, for any $q \in \mathcal{X}$, with probability $1 - \varepsilon$, nearest neighbor search algorithm uses at most $h^* + n_0$ triplet comparisons to find an approximate nearest neighbor of q.*

[1]Technically, $\mathcal{C}_S$ is a subset of the algebra generated by the "generalized half spaces" of the induced space (S, δ).

The other applicable methods in the comparison-based setting make different assumptions on the dataset, thus the upper bound on the required number of triplets is hardly comparable with them. If we neglect this fact and only compare the dependency on n, we can summarize the theoretical comparison in Table 2.1.

Table 2.1: Theoretical comparison with existing methods

Method	Construction	Query
Comparison Tree	$n\log n$	$\log n$
(Goyal *et al.*, 2008)	$n^2\log n$	$\log n$
(Lifshits and Zhang, 2009)	$n\log^2 n$	$\log n$
(Tschopp *et al.*, 2011)	$n\log^2 n$	$\log^2 n$
(Houle and Nett, 2015)	$n\log^3 n$	$\log^3 n$

While the above discussion sheds light on the required number of triplet comparisons, it still leaves one wondering about the quality of the nearest neighbor obtained from the comparison-tree. In the following result, we show that under certain conditions on the behavior of the metric in a neighborhood of a given query q, the search method succeeds in finding the true nearest neighbor of q. We use $x_q \in S$ to denote the true nearest neighbor of q, while $\hat{x}_q$ is the element returned by the nearest neighbor search on a randomly constructed comparison-tree. We write $B(x,r)$ and $B^\circ(x,r)$ to denote closed and open balls, respectively.

Theorem 3 (Exact nearest neighbor) *Given $S \subseteq \mathcal{X}$ and $q \in \mathcal{X}$. If there exist constants $C > 0$ and $\alpha \in (0,1]$ such that for every $A \in \mathcal{C}_S$ containing x_q, and for all $x \in A\backslash\{x_q\}$,*

$$\left|B\big(q,\delta(q,x)+2\delta(q,x_q)\big)\cap A\right| \leq \left|B^\circ\big(q,\delta(q,x)\big)\cap A\right| + C|A|^{1-\alpha} , \tag{2.3}$$

then

$$\mathsf{P}\left(\hat{x}_q \neq x_q\right) \leq \frac{360C\tilde{c}^2}{\alpha} n_0^{-\alpha} , \tag{2.4}$$

where the probability is with respect to the random construction of the tree.

The condition on q implies that there are not many points that have the same distance to q as the nearest neighbor. Under the local restrictions defined in (2.3) on the query q, the error bound (2.4) states that one can achieve an arbitrarily small error probability if n_0 is chosen large enough, depending on the strong expansion rate $\tilde{c}$.

Remark 4 *The assumption in* (2.3) *can also be substituted by alternative conditions. For instance, the error bound in* (2.4) *holds (up to constants) if there exists $D > 1$ such that*

$$|B(q,\lambda r)\cap A| \leq \lambda^D |B(q,r)\cap A| \tag{2.5}$$

for all $A \in \mathcal{C}_S$, $\lambda > 1$, and $r > \delta(q,x_q)$. One can see the inherent resemblance of (2.5) *to the notion of strong expansion rate* (2.1).

We remark again on the required conditions. The notion of strong expansion rate, though used in Ram and Gray (2013), is stronger than standard conditions used in many works (Dasgupta and Sinha, 2015; Karger and Ruhl, 2002). Our main reason for resorting to this notion is because of the data dependent random splits used in comparison-tree construction. While projection-based methods also use random hyperplanes for splitting each node, such hyperplanes are independent of the given set, making the analysis simpler (Dasgupta and Sinha, 2015). On the other hand, prior works in non-Euclidean setting construct data structures that naturally adhere to the structure of the metric balls (Karger and Ruhl, 2002). Unlike both these works, in the present setting, one cannot guarantee that a condition defined on the whole set will also hold for each of the partitions obtained during splits. Hence, the condition of strong expansion rate has been used in our analysis. The additional assumption on q seems essential since one can construct trivial examples where the nearest neighbor search is quite likely to fail.

Proof of Theorem 1

We now prove Theorem 1 using two lemmas.

Lemma 5 (Probability of unbalanced split) *For any $A \in \mathcal{C}_S$ and $\beta \in (0,1)$, the probability that the random split in the comparison-tree construction algorithm creates a child of A with less than $\beta|A|$ elements is at most $4\tilde{c}^2\beta$.*

Thus, each split in the tree is reasonably balanced, and hence, it is likely that the size of the nodes decays rapidly with their depth. This fact is formalized below.

Lemma 6 (Maximum node size at depth h) *Let $A \in \mathcal{C}_S$ be a node at depth h of the tree. If $4\tilde{c}^2\beta \leq 1$, then the probability that A has more than m elements is at most $(8\tilde{c}^2\beta)^{h-1}(n/m)^{(1/\beta)\log(1/4\tilde{c}^2\beta)}$.*

We finish the proof of Theorem 1 by observing that the height of the tree is greater than h only if there is a node at depth h of size greater than n_0. By taking a union over all possible 2^h nodes at depth h, one can see that the probability of this event is at most $2(16\tilde{c}^2\beta)^{h-1}(n/n_0)^{(1/\beta)\log(1/4\tilde{c}^2\beta)}$. This probability is less than ε if we fix $\beta = 1/32\tilde{c}^2$, and

$$h = \left(1 + 2\log\left(\frac{e}{\varepsilon}\right) + 96\tilde{c}^2\log\left(\frac{n}{n_0}\right)\right) \leq h^* .$$

We now prove the above two lemmas.

Proof. [Proof of Lemma 5] Let $|A| = m$, and $\mathbf{1}(\cdot)$ denote the indicator function. Then the probability of splitting A to create a child of size smaller than βm is at most

$$\begin{aligned}
&\tfrac{1}{m(m-1)} \sum_{\substack{x_1 \in A \\ x_2 \in A\setminus\{x_1\}}} \mathbf{1}\Big(\{|\{x \in A : \delta(x,x_2) \leq \delta(x,x_1)\}| \leq \beta m\} \\
&\qquad\qquad \cup \{|\{x \in A : \delta(x,x_1) < \delta(x,x_2)\}| \leq \beta m\}\Big) \\
&\leq \tfrac{1}{\binom{m}{2}} \sum_{\substack{x_1 \in A \\ x_2 \in A\setminus\{x_1\}}} \mathbf{1}\Big(\big|\{x \in A : \delta(x,x_1) < \delta(x,x_2)\}\big| \leq \beta m\Big).
\end{aligned}$$

Note that the set $\{x \in A : \delta(x,x_1) < \delta(x,x_2)\}$ contains the set $B\left(x_1, \frac{1}{4}\delta(x_1,x_2)\right) \cap A$, where $B(\cdot,\cdot)$ is the closed ball. Thus, one may bound the above probability by the fraction of x_1, x_2 pairs for which this ball contains less than βm elements. Moreover, using the condition of strong expansion rate (2.1), one has

$$\big|B\big(x_1, \delta(x_1,x_2)\big) \cap A\big| \leq \tilde{c}^2 \left|B\left(x_1, \tfrac{1}{4}\delta(x_1,x_2)\right) \cap A\right| .$$

Thus, one may only count the x_1, x_2 pairs for which $B(x_1, \delta(x_1,x_2)) \cap A$ contains at most $\tilde{c}^2\beta m$ elements. Now, for every x_1, if one sorts x_2 in the increasing order of $\delta(x_1,x_2)$, then the indicator is true only for the first $\tilde{c}^2\beta m$ of x_2's. Thus, the probability of an unbalanced split is at most $2\tilde{c}^2\beta m^2/m(m-1) \leq 4\tilde{c}^2\beta$. □

Proof. [Proof of Lemma 6] We denote the path from the root of the tree to A by $S = A_1 \supset A_2 \supset \ldots \supset A_{h-1} \supset A_h = A$. Let A'_j denote the sibling of A_j for $j \geq 2$. By the Markov inequality, one can write for any $t > 0$,

$$\begin{aligned}
\mathsf{P}(|A| > m) &\leq \left(\frac{n}{m}\right)^t \mathsf{E}\left[\frac{|A_h|^t}{|A_1|^t}\right] \\
&= \left(\frac{n}{m}\right)^t \mathsf{E}\left[\frac{|A_{h-1}|^t}{|A_1|^t} \mathsf{E}\left[\frac{|A_h|^t}{|A_{h-1}|^t}\middle| A_{h-1}\right]\right].
\end{aligned}$$

One can bound the inner conditional expectation as

$$\begin{aligned}
\mathsf{E}\left[\frac{|A_h|^t}{|A_{h-1}|^t}\middle| A_{h-1}\right] &= \mathsf{E}\left[\frac{|A_h|^t}{|A_{h-1}|^t}\mathbf{1}\big(|A_h| > (1-\beta)|A_{h-1}|\big)\middle| A_{h-1}\right] \\
&\quad + \mathsf{E}\left[\frac{|A_h|^t}{|A_{h-1}|^t}\mathbf{1}\big(|A_h| \leq (1-\beta)|A_{h-1}|\big)\middle| A_{h-1}\right]
\end{aligned}$$

$$\leq \mathsf{P}\big(|A'_h| \leq \beta |A_{h-1}|\big) + (1-\beta)^t \ ,$$

where the inequality follows by replacing the ratio by 1 in the first expectation, and $|A_h|$ by its upper bound in the second one. Due to Lemma 5, one can see that this bound is at most $\big(4\tilde{c}^2\beta + (1-\beta)^t\big)$. For $t = (1/\beta)\log(1/4\tilde{c}^2\beta)$, one can use the fact that $(1/\beta)\log(1/1-\beta) \geq 1$ to show that the above expectation is at most $8\tilde{c}^2\beta$. Subsequently, we use the same technique of conditioning with every A_j, $j = 2,\ldots,h-2$ to obtain

$$\mathsf{P}(|A| > m) \leq \Big(\frac{n}{m}\Big)^{\frac{1}{\beta}\log(\frac{1}{4\tilde{c}^2\beta})} (8\tilde{c}^2\beta)^{h-1}.$$

□

Proof of Theorem 3

The nearest neighbor search for a query point q is done by traversing the tree from the root to one of the leaves. Let us denote the visited path by $S = A_1 \supset A_2 \supset \ldots \supset A_{k-1} \supset A_k$, where A_k is the leaf node containing $\hat{x}_q$. We assume that $q \notin S$, as otherwise the nearest neighbor search algorithm returns the query. By simple reasoning, it follows that $\hat{x}_q \neq x_q$ only if $x_q \notin A_k$, which happens if there is $l \in \{1,2,\ldots,k-1\}$ such that $x_q \in A_l \backslash A_{l+1}$. Hence,

$$\begin{aligned}
&\mathsf{P}(\hat{x}_q \neq x_q) = \mathsf{P}\left(\bigcup_{l=1}^{k-1}\{x_q \in A_l, x_q \notin A_{l+1}\}\right) \\
&\leq \sum_{l=1}^{k-1} \mathsf{P}(x_q \notin A_{l+1} | x_q \in A_l)\mathsf{P}(x_q \in A_l) \\
&\leq \sum_{l=1}^{k-1} \mathsf{P}\left(x_q \notin A_{l+1} \,|x_q \in A_l, |A_l| \geq m_l\right) \mathsf{P}\left(|A_l| \geq m_l\right) + \\
&\mathsf{P}\left(x_q \notin A_{l+1} \,|x_q \in A_l, |A_l| < m_l\right) \mathsf{P}\left(|A_l| < m_l\right)
\end{aligned} \tag{2.6}$$

where $m_l = n_0/(1-\gamma)^{k-1-l}$ for some $\gamma \in (0,1)$. The first inequality is due to union bound, while the second one uses $\mathsf{P}(x_q \in A_l) \leq 1$ and further decomposes based on $|A_l|$.

Lemma 7 (Probability of missing nearest neighbor in one branch) *Under the condition on q stated in Theorem 3, for any $l = 1,2,\ldots,k-1$,*

$$\mathsf{P}(x_q \notin A_{l+1} | A_l, x_q \in A_l) \leq C|A_l|^{-\alpha} \ .$$

Note that for the two conditional probabilities in (2.6), $|A_l|$ is at least m_l and n_0, respectively. Using Lemma 7, one can bound these probabilities. To obtain a bound on $\mathsf{P}(|A_l| < m_l)$, we follow Lemma 6.

Observe that Lemma 6 implies that after repeated splits, it is less likely that the ratio of the final node to the root node will be large. In the present context, we know that $|A_{k-1}| \geq n_0$, and so, for any $l < k-1$, the bound in Lemma 6 can be used to argue that $|A_{k-1}|/|A_l|$ cannot be large. Formally,

$$\begin{aligned}\mathsf{P}(|A_l| < m_l) &= \mathsf{P}\left(\frac{|A_{k-1}|}{|A_l|} > \frac{n_0}{m_l}\right)\\ &\leq (8\tilde{c}^2\beta)^{k-1-l}\left(\frac{1}{(1-\gamma)^{k-1-l}}\right)^{\frac{1}{\beta}\log\left(\frac{1}{4\tilde{c}^2\beta}\right)}\\ &= \left(\frac{0.25}{(1-\gamma)^{96\tilde{c}^2\log 2}}\right)^{k-1-l},\end{aligned}$$

using $\beta = 1/32\tilde{c}^2$ as in Theorem 1. Substituting the above bound in (2.6) and using Lemma 7, we have

$$\mathsf{P}(\hat{x}_q \neq x_q) \leq \sum_{l=1}^{k-1} Cn_0^{-\alpha}(1-\gamma)^{\alpha(k-1-l)} + Cn_0^{-\alpha}\left(\frac{0.25}{(1-\gamma)^{96\tilde{c}^2\log 2}}\right)^{k-1-l}.$$

Choosing $\gamma = 1-(0.25)^{1/(\alpha+96\tilde{c}^2\log 2)}$, one can see that the second term is smaller than the first, and hence,

$$\mathsf{P}(\hat{x}_q \neq x_q) \leq \frac{2Cn_0^{-\alpha}}{1-(1-\gamma)^{\alpha}} \leq \frac{2Cn_0^{-\alpha}}{1-(0.25)^{\alpha/67.5\tilde{c}^2}}.$$

From above, we obtain the bound in (2.4) by using the relation $1-a \leq b(1-a^{1/b})$, which holds for any $a \in (0,1)$ and $b > 1$. This proves Theorem 3.

We end the section with the proof of Lemma 7.

Proof. [Proof of Lemma 7] Let $|A_l| = m$, and let us order the elements such that $\delta(q,x_i) \leq \delta(q,x_j)$ for $i < j$. Since $x_q \in A_l$, we have $x_1 = x_q$. Note that if $x_q \notin A_{l+1}$, then it is certainly not a pivot element. Moreover, if $x_i, x_j \in A_l$ are the pivot elements for $i < j$, then $x_q \notin A_{l+1}$ implies $\delta(x_q,x_i) \geq \delta(x_q,x_j)$. The inequality can even be strict depending on which is chosen as the left pivot. Hence, we have

$$\mathsf{P}(x_q \notin A_{l+1} | A_l, x_q \in A_l) \leq \frac{1}{m(m-1)}\sum_{2\leq i<j\leq m} \mathbf{1}\big(\delta(x_q,x_i) \geq \delta(x_q,x_j)\big).$$

By the triangle inequality,

$$\begin{aligned}\delta(x_q,x_i) &\leq \delta(q,x_i)+\delta(q,x_q) \text{ and}\\ \delta(x_q,x_j) &\geq \delta(q,x_j)-\delta(q,x_q)\,.\end{aligned}$$

Hence, one may count the pairs $i < j$ for which

$$\delta(q,x_j) \leq \delta(q,x_i)+2\delta(q,x_q)\,.$$

As a consequence, we can write

$$\mathsf{P}(x_q \notin A_{l+1}|A_l,x_q \in A_l) \leq \frac{1}{m(m-1)} \cdot \sum_{i=2}^{m} \big|\{x \in A : \delta(q,x_i) \leq \delta(q,x) \leq \delta(q,x_i){+}2\delta(q,x_q)\}\big|,$$

where each term in the sum is at most $Cm^{1-\alpha}$ due to the assumption on q. Thus, the claim of the lemma is true. □

2.3 Experiments

The experiments are divided into three subsections. In the first part we compare the performance of comparison-trees with other binary space partitioning trees. We used datasets in Euclidean space as the other tree structures require the vector representations. In the next part we compare the performance of comparison-tree with the Rank Cover Tree (Houle and Nett, 2015). Only a few methods have considered the nearest neighbor search with distance comparisons (Houle and Nett, 2015; Tschopp *et al.*, 2011). We choose the more recent method proposed by Houle and

Table 2.2: Description of datasets

Dataset	Size	Dimension	Distance
MNIST	70000	784	Euclidean
Gisette	12500	5000	Euclidean
CoverType	50000	53	Euclidean
Corel	19787	44	Euclidean
Chess	28056	6	Mismatch
CoAuth	11204	-	Shortest Path
MSC	10848	-	Shortest Path

Nett (2015) as a competitor to our proposed comparison-tree. In the last subsection, we investigate the defined growth parameter on real datasets.

2.3.1 Euclidean setting

In this subsection, we compare the performance of comparison-trees to standard space partitioning trees in Euclidean spaces. Note that the latter have access to the vector representation of the points (and thus also to all pairwise distances), whereas the comparison-tree only has access to triplet comparisons. Thus, the purpose of this comparison is not to show that the comparison-tree "outperforms" the other ones, but to examine whether it is much worse or not. There are numerous tree constructions in Euclidean spaces. Based on the results in Ram and Gray (2013), we compare our method with KD-Tree (Bentley, 1975), RP-Tree (Dasgupta and Freund, 2008) and PA-Tree (McNames, 2001).

A description of the datasets is presented in Table 2.2. MNIST is a dataset of hand-written digits (Lecun and Cortes, 1998). Gisette, CoverType and Chess (King-Rook vs. King) are from the UCI repository (Lichman, 2013). Corel is a subset of histograms defined and used in Liu *et al.* (2004). CoAuth is the collaboration network of Arxiv High Energy Physics from Davis and Hu (2011). We consider the largest connected component of the graph as the set of items, and the shortest path in the graph as the distance metric. MSC (Boeing/msc10848) is a similar but weighted graph from Davis and Hu (2011)[2]. Note that only the first four datasets live in Euclidean spaces.

We assess the performance of the nearest neighbor search by the leave-one-out method. As a performance measure, we report the empirical probability of missing the nearest neighbor:

$$1/|S| \sum_{q \in S} \mathbb{1}\left(\mathrm{NN}_{alg}(q) \neq \mathrm{NN}(q)\right).$$

Here S is the whole dataset, NN_{alg} denotes the result of nearest neighbor search by the algorithm while the true nearest neighbor is $\mathrm{NN}(q)$.

Figure 2.1 shows the performance of comparison-tree versus other methods in the Euclidean space. The comparison-tree has less error compared to RP-Tree and KD-Tree, and has slightly worse performance comparing with the PA-Tree. However, the differences are not huge, and we find the behavior of the comparison-tree quite satisfactory, given that it receives much less input information than the other methods.

[2] There are negative edge weights in the graph, however we use absolute values of edge weights to have a metric by using shortest path distances.

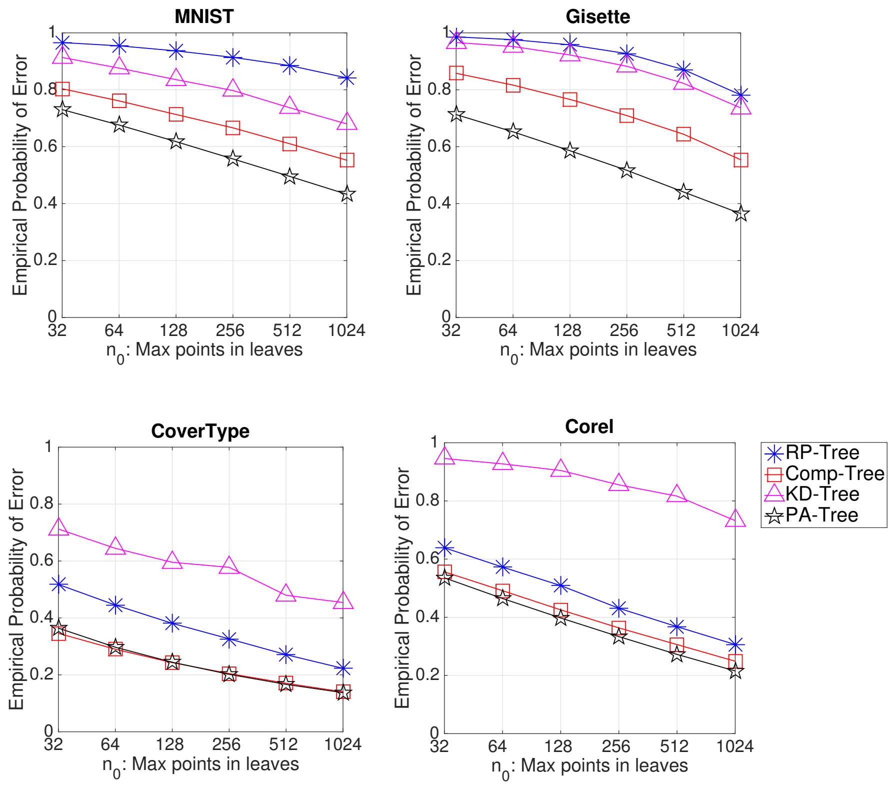


Figure 2.1: Performance comparison of comparison-tree versus other binary space partitioning trees with respect to the maximum leaf size, n_0. The experiment is performed on four datasets; the name of dataset is appeared on top of each plot.

2.3.2 Comparison-based setting

Among the few comparison-based methods cited in the introduction, many are not practical or have already been shown to perform sub-optimally (see the theoretical comparison in table 2.1). The most promising competitor to our method is the Rank Cover Tree (RCT) (Houle and Nett, 2015). As the original implementation of the authors was not available, we implemented the method ourselves in MATLAB.

We have two objectives when comparing the two comparison-based trees: the number of required triplet questions, and the accuracy they achieve in the nearest neighbor search. While the latter is easy to compare, the former is more of a challenge. It is impossible to construct an RCT with the same number of triplets that the comparison-tree requires in construction phase, since the RCT needs orders of magnitude more triplets in construction. Thus, we construct both trees in such a way that the number of triplet comparisons in the *query phase* is matched. We then compare the nearest

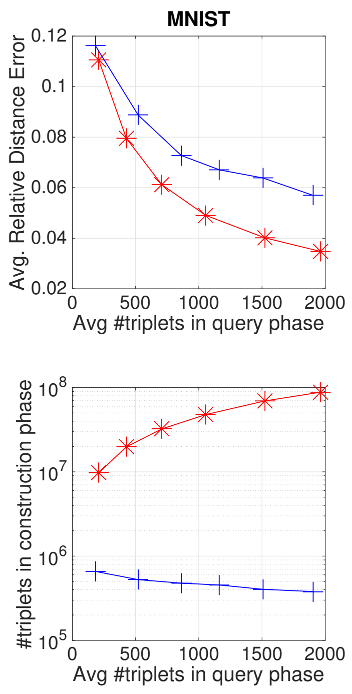
neighbor search performance, but also the number of triplets in the tree construction phase. For the RCT, the performance and the number of comparisons can be balanced by adjusting the coverage parameter ω (see Houle and Nett, 2015, Section 4). For comparison-trees, n_0 plays a similar role. By varying these two parameters we match the number of comparisons in the query phase.
We randomly choose 1000 data points in each experiment as test set for the query phase and the rest of the dataset for the tree construction. The empirical error defined in previous subsection is not well-defined for some of datasets in this subsection. In CoAuth and Chess datasets, many points have more than one nearest neighbor. In this way, considering the probability of eTrro for exact nearest neighbor search can be missleading. Thus, we report the average relative distance error (Liu *et al.*, 2004) defined as the following:

$$(1/|S|)\sum_{q\in S}\left(\frac{d_{alg}}{d_{\mathrm{NN}}}-1\right).$$

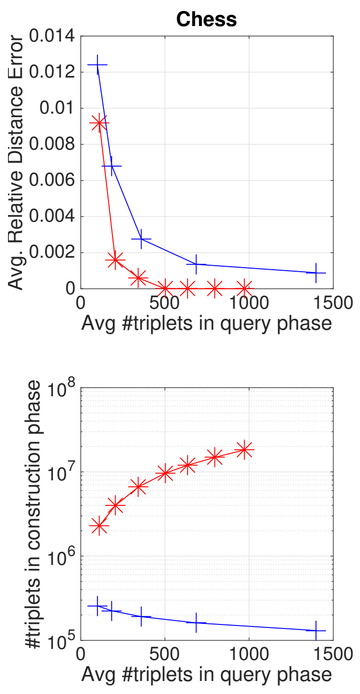
Here d_{alg} denotes the distance of the query to the predicted nearest neighbor by the algorithm and d_{NN} denotes the distance of the query to the true nearest neighbor.
Figure 2.2 shows the performance of the comparison-tree compared to the RCT on four datasets from Table 2.2. The results on the remaining Euclidean datasets are very similar to MNIST, hence we do not present them. We consider different parameter settings and match the average number of triplets used in the query phase. In terms of the relative distance errors, the RCT works slightly better in datasets with low intrinsic dimension, specially when we are provided with more triplets in query phase. However, as the bottom rows in Figure 2.2 show, to achieve this performance the RCT needs orders of magnitude more triplet comparisons in the tree construction phase. Therefore, if answering triplet comparisons is expensive, then the comparison-tree clearly is a good alternative to the RCT.

2.3.3 Expansion rate approximation

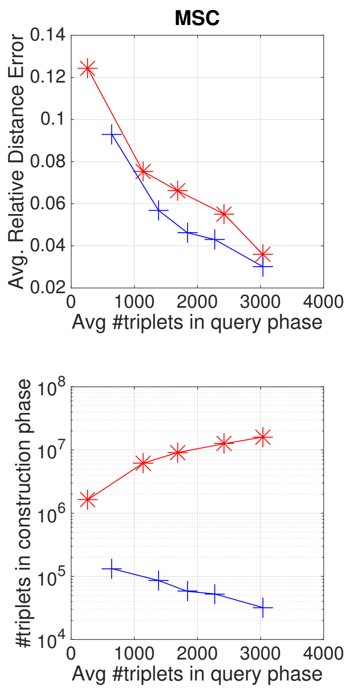
In our theoretical analysis, we used the strong expansion condition defined in Equation (2.1). It is an obvious question to find out how strong these conditions really are and what the corresponding constants in our datasets would be. To this end, we provide a method to estimate the expansion rates for our datasets. We fix a dataset, for each point x we look for the smallest $\tilde{c}$ such that Equation (2.1) holds for that particular point. We find the smallest value with respect to various radii r. In this way we estimate an empirical pointwise $\tilde{c}(x)$ value for each point. Since the definition depends on the number of points in the dataset, we randomly choose 10000 points from each dataset for these experiments. The distribution of empirical expansion rates $\tilde{c}(x)$ are plotted by box-and-whisker plots in Figure 2.3.



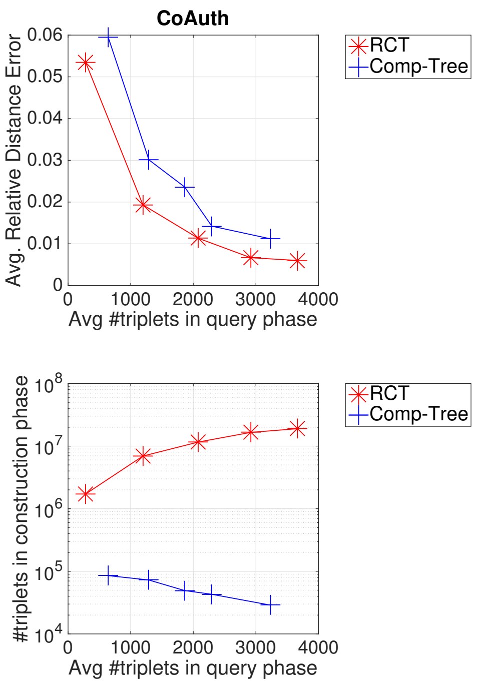
(a) MNIST



(b) Chess



(c) MSC



(d) CoAuth

Figure 2.2: Comparing the performance of the comparison-tree versus the RCT with various parameters in the construction phase. In each of the four plots, corresponding to the four datasets, the upper row denotes the average relative distance error, while the bottom row is the number of triplets used in the construction phase.

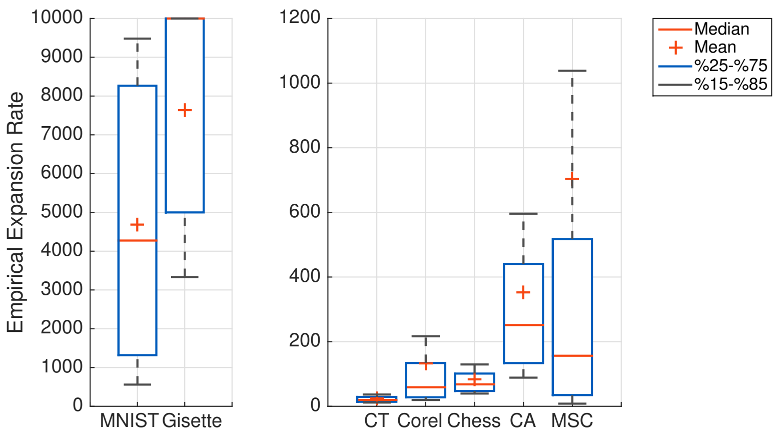


Figure 2.3: Distribution of empirical expansion rates $\tilde{c}(x)$ estimated for various datasets. Each bar represents the distribution of pointwise expansion rates for the corresponding dataset. CoAuth and CoverType datasets are abbreviated as "CA" and "CT" respectively. Note that the range of expansion rates is 10 times higher for the first two datasets, thus we plotted them separately.

For our theoretical analysis, we used the smallest possible $\tilde{c}$ for the whole dataset. However the distribution of pointwise values $\tilde{c}(x)$ is a more practical criterion to consider. Except for MNIST and Gisette, these values are reasonably small. Therefore, the values can justify the validity of the assumption on real datasets.

2.4 Conclusion

Comparison-based nearest neighbor search is a fundamental ingredient in machine learning algorithms in the comparison-based setting. Because triplet comparisons are expensive, we investigate the query complexity of comparison-based nearest neighbor algorithms. In particular, we study the comparison-tree, which leads to a nice and simple, yet adaptive data structure. We prove that under strong conditions on the underlying metric, the comparison-tree has logarithmic height, and we can bound the error in nearest neighbor search. We also show in simulations that comparison-trees perform not much worse than Euclidean data structures (albeit using much less information about the data), and perform favorably to other comparison-based methods if we take both the number of triplet comparisons and the nearest neighbor errors into account.

There are still a number of interesting open questions to address. The conditions we use in our analysis are rather strong, and this seems to be the case for all other studies in this area as well. Can they be considerably weakened? Can we prove that our conditions will be satisfied with small constants if we sample point from a nice metric or Euclidean space? Finally, all the above work assumes that a ground truth for the triplet comparisons exists and that the answers to the triplet queries are always correct. It would be interesting to see how the query complexity of the comparison-tree increases if the error in the triplets increases.

Chapter 3

Comparison-based random forests

In this chapter, we consider classification and regression problems in a comparison-based setting where we are given the labels $y_1, \dots, y_n$ of unknown objects $x_1, \dots, x_n$, and we can *actively* query triplet comparisons, as defined in Equation 1.1. We solve the classification/regression problems by a new random forest algorithm that requires only triplet comparisons. Standard random forests (Biau and Scornet, 2016) are one of the most popular and successful classification/regression algorithms in Euclidean spaces (Fernández-Delgado *et al.*, 2014). However, they *heavily* rely on the underlying vector space structure. In our comparison-based setting we require a completely different tree building strategy. We use a modified version of the comparison-tree defined in the previous chapter.

We introduce comparison-based random forest (CompRF), which consists of a collection of comparison-trees built on the training set. The construction of each tree requires only some triplet comparisons of the training items. In the test phase, we again need only the answers to some of the triplet comparisons. Triplet comparisons are used to traverse the tree and reach the leaf node that contains the query point. The final prediction of the forest is an aggregation of the label of items in the leaf nodes of trees. Similar to traditional random forests, aggregating a bunch of trees leads to a desirable generalization.

We study the proposed CompRF both from a theoretical and a practical point of view. In Section 3.2, we give sufficient conditions under which a slightly simplified variant of the comparison-based random forest is statistically consistent. The required techniques are already tricky for standard random forests (Scornet, 2016), but require a lot of extra work for comparison-based random forests. In Section 3.3, we apply the CompRF to various datasets. In the first set of experiments we compare our random forests to traditional random forests, based on classification and regression tree (CART), on Euclidean data. In the second set of experiments, the distances between objects are known while their representation is missing. Here, traditional random forests cannot be used any more. However, the comparison-based random forests are still applicable in this setting. Finally, we consider a case in which only triplet comparisons are available. In this setting we compare the performance of our method with the embedding algorithms and subsequently applying the KNN classifier.

3.1 Comparison-Based random forests

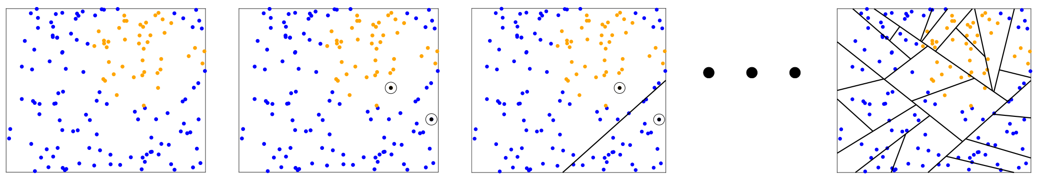


Figure 3.1: Construction of the comparison tree, illustrated in the Euclidean setting. (i) The current cell contains points with two different labels. (ii) Two pivot points with opposite labels are chosen randomly from all sample points in the current cell (circled black dots). (iii) The current cell is split according to whether points are closer to the one or the other pivot; in the Euclidean setting this corresponds to a hyperplane split. (iv) Result after recursive application of this principle with final leaf size $n_0 = 10$.

Random forests, first introduced in Breiman (2001), are one of the most popular algorithms for classification and regression in Euclidean spaces. In a comprehensive study on more than 100 classification tasks, random forests show the best performance among many other general purpose methods (Fernández-Delgado *et al.*, 2014). However, standard random forests heavily rely on the vector space representation of the underlying data points, which is not available in a comparison-based setting. Instead, we propose a comparison-based random forest algorithm for classification and regression tasks. The main ingredient is the comparison-tree, which only uses triplet comparisons and does not rely on Euclidean representation or distances between items.

We first recall the **CART random forest**. The input consists of a labeled set $D_n = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$. To build an individual tree, we first draw a random subsample D_s of a_n points from D_n. Second, we select a random subset Dim_s of size `mtry` of all possible dimensions $\{1,2,\ldots,d\}$. The tree is then built based on recursive, axis-aligned splits along a dimension randomly chosen from Dim_s. The exact splitting point along this direction is determined via the CART criterion, which also involves the labels of the subset D_s of points (see Biau and Scornet (2016) for details). The tree is grown until each cell contains at most n_0 points—these cells then correspond to the leaf nodes of the tree. To estimate a regression function $m(x)$, each individual tree routes the query point to the appropriate leaf and outputs the average response over all points in this leaf. The random forest aggregates M such trees. Let us denote the prediction of tree i at point x by $m_i(x,\Theta_i,D_n)$, where Θ_i encodes the randomness in the tree construction. Then the final forest estimation at x is the average result over all trees (for classification, the average is replaced by a majority vote):

$$m_{M,n}(x;(\Theta_i)_{1\leq i\leq M},D_n) = \frac{1}{M}\sum_{i=1}^{M} m_i(x,\Theta_i,D_n)\,.$$

The general consensus in the literature is that CART forests are surprisingly robust to parameter choices. Consequently, people use explicit rules of thumb, for example to set `mtry` $= \lceil d/3 \rceil$, and $n_0 = 5$ (resp. $n_0 = 1$) for regression (resp. classification) tasks.
We now suggest to replace CART trees by comparison trees, leading to **comparison-based random forests (CompRF)**. We suggest to use the comparison-tree structure (as described in previous chapter) for regression tasks. However, for classification tasks we propose a supervised variant of the comparison tree, which we refer to as "supervised comparison-tree". The tree structure is quite similar to the main definition proposed in section 2.1. We describe the two variations of the CompRF algorithm for the regression and classification tasks separately.

Algorithm 3 *CompTree*(S,n_0):
Supervised comparison tree construction

```
Input: Labeled data $S$ and maximum leaf size $n_0$
Output: Comparison tree $T$
1: $T.root \leftarrow S$
2: if $|S| > n_0$ then
3:     Sample distinct $(x_1,y_1),(x_2,y_2) \in S$ s.t. $y_1 \neq y_2$
       (if all points have the same label choose randomly)
4:     $S_1 \leftarrow \{(x,y) \in S : \delta(x,x_1) \leq \delta(x,x_2)\}$
5:     $T.leftpivot \leftarrow x_1,\ T.rightpivot \leftarrow x_2$
6:     $T.leftchild \leftarrow CompTree(S_1,n_0)$
7:     $T.rightchild \leftarrow CompTree(S \backslash S_1,n_0)$
8: end if
9: Return $T$
```

CompRF for classification

The supervised comparison tree construction for a labeled set $S \subset \mathcal{X} \times \{0,1\}$ is as follows (see Algorithm 3 and Figure 3.1): we randomly choose two pivot points x_1 and x_2 with different labels y_1 and y_2 among the points in S (the case where all

the points in S have the same label is trivial). For every remaining point $(x,y) \in S$, we request the triplet comparison "$\delta(x,x_1) < \delta(x,x_2)$." The answer to this query determines the relative position of x with respect to the generalized hyperplane separating x_1 and x_2. We assign the points closer to x_1 to the first child node of S and the points closer to x_2 to the other one. We now recurse the algorithm on the child nodes until less than n_0 points remain in every leaf node of the tree.
The supervised pivot selection is analogous to the CART criterion. However, instead of a costly optimization over the choice of split, it only requires to choose pivots with different labels. In Section 3.3.1, we empirically show that the supervised split procedure leads to a better performance than the CART forests for classification tasks.

CompRF for regression

In contrast to the classification task, it is not obvious how the pivot selection should depend on the output values for a regression task. Here we use an unsupervised version of the forest (unsupervised CompRF): we choose the pivots x_1,x_2 without considering y_1,y_2.(line 3 of Algorithm 3)

Label prediction with the CompRF

The final comparison-based random forest consists of M independently constructed comparison trees. To assign a label to a query point, we traverse every tree to a leaf node, then we aggregate all the items in the leaf nodes of M trees to estimate the label of the query item. For classification, the final label is the majority vote over the labels of the accumulated set (in the multiclass case we use a one vs. one approach). For regression we use the mean output value. The pseudocode of the CompRF algorithm is provided in Algorithm 4.

Intuitive comparison with the CART random forests

The general understanding is that the efficiency of CART random forests is due to: (1) the randomness due to subsampling of dimensions and data points (Breiman, 1996); (2) the CART splitting criterion that exploits the label information already in the tree construction (Breiman *et al.*, 1984). A weakness of CART splits is that they are necessarily axis-aligned, and thus not well-adapted to the geometry of the data.
In comparison trees, randomness is involved in the tree construction as well. But once a splitting direction has been determined by choosing the pivot points, the exact splitting point along this direction cannot be influenced any more, due to the lack of a vector space representation. On the other hand, the comparison tree splits are

Algorithm 4 *CompRF*(D_n, q, M, n_0, r):
CompRF prediction at query q

```
Input: Labeled dataset $D_n \subset \mathcal{X} \times \{0,1\}$, query $q \in \mathcal{X}$, leaf size $n_0$, trees $M$ and
    subsampling ratio $r$.
Output: $y_q$ = label prediction for $q$
1: Set $C = \emptyset$ as the list of predictor items
2: for j=1,...,M do
3:     Take a random subsample $D_s \subset D_n$, s.t., $\frac{|D_s|}{|D_n|} = r$
4:     $T_j \leftarrow CompTree(D_s, n_0)$
5:     Given $q$, traverse the tree $T_j$ to the leaf node $N_j$
6:     $C \leftarrow C \cup N_j$
7: end for
8: Return MajorityVote($\{y | (x,y) \in C\}$)
```

well adapted to the data geometry by construction, giving some advantage to the comparison trees.
All in all, the comparison-based random forest is a promising candidate with slightly different strengths and weaknesses than CART random forest. Our empirical comparison in Section 3.3.1 reveals that it performs surprisingly well and can even outperform CART random forests in certain settings.

3.2 Theoretical analysis

Despite their intensive use in practice, theoretical questions regarding the consistency of the original procedure of Breiman (2001) are still under investigation. Most of the research focuses on simplified models in which the construction of the forest does not depend on the training set at all (Biau, 2012), or only via the x_is but not the y_is (Biau *et al.*, 2008; Ishwaran and Kogalur, 2010; Denil *et al.*, 2013). Recent efforts nearly closed this gap, notably Scornet *et al.* (2015), where it is shown that the original algorithm is consistent in the context of additive regression models and under suitable assumptions. However, there is no previous work on the consistency of random forests constructed only with triplet comparisons.
As a first step in this direction, we investigate the consistency of individual comparison trees, which is the first building block in the study of random forests consistency. As it is common in the theoretical literature on random forests, we consider a slightly modified version of the comparison tree. We assume that the pivot points are not randomly drawn from the underlying sample but according to the true distribution of the data. In this setting, we show that, when the number of observations grows to infinity, (i) the diameter of the cells converges to zero in probability, and (ii) each

cell contains an arbitrarily large number of observations. Using a result of Devroye *et al.* (1996), we deduce that the associated classifier is consistent. The challenging part of the proof is to obtain control over the diameter of the cells. Intuitively, as in Dasgupta and Freund (2008, Lemma 12), it suffices to show that each cut has a larger probability to decrease the diameter of the current cell than that of leaving it unchanged. To prove this in our case is very challenging since both the position and the decrease in diameter caused by the next cut depend on the *geometry* of the cell.

3.2.1 Continuous comparison-tree

As it is the case for most theoretical results on random forests, we carry out our analysis in a Euclidean setting (however, the comparison-forest only has indirect access to the Euclidean metric via triplet queries). We assume that the input space is $\mathcal{X} = [0,1]^d$ with distance δ given by the Euclidean norm, that is, $\delta(x,y) = \|x-y\|$. Let X be a random variable with support included in $[0,1]^d$. We assume that the observations $X_1,\dots,X_n \in [0,1]^d$ are drawn independently according to the distribution of X. We make the following assumptions:

Assumption 3.2.1 (Bounded density on the unit cube) *The random variable $X \in [0,1]^d$ has density f with respect to the Lebesgue measure on $[0,1]^d$. Additionally, there exist constants $0 < f_{\min} \le f_{\max} < +\infty$ such that*

$$\forall x \in \mathcal{X}, \quad f_{\min} \le f(x) \le f_{\max}.$$

For any $x, y \in \mathbb{R}^d$, let us define

$$\Delta(x,y) := \left\{ z \in \mathbb{R}^d \mid \delta(x,z) = \delta(y,z) \right\}.$$

In the Euclidean setting, $\Delta(x,y)$ is a hyperplane that separates $\mathbb{R}^d$ in two half-spaces. We call H_x (resp. H_y) the open half-space containing x (resp. y). The set S_1 in Algorithm 3 corresponds to $S \cap H_{x_1}$.
We can now define the continuous comparison tree.

Definition 8 (Continuous comparison tree) *A* continuous comparison tree *is a random infinite binary tree T^0 obtained* via *the following iterative construction:*

- *The root of T^0 is $[0,1]^d$;*
- *Assuming that level ℓ of T^0 has been built already, then level $\ell+1$ is built as follows: For every cell C at height ℓ, draw $X_1, X_2 \in C$ independently according to the distribution of X restricted to C. The children of C are defined as the closure of $C \cap H_{X_1}$ and $C \cap H_{X_2}$.*

For any sequence $(p_n)_{n\geq 0}$, a truncated, *continuous comparison tree $T^0(p_n)$ consists of the first $\lfloor p_n \rfloor$ levels of T^0.*

From a mathematical point of view, the continuous tree has a number of advantages. (i) Its construction does not depend on the responses $Y_1, \ldots, Y_n$. Such a simplification is quite common because data-dependent random tree structures are notoriously difficult to analyze (Biau *et al.*, 2008). (ii) Its construction is formally independent of the finite set of data points, but "close in spirit": Rather than sampling the pivots among the data points in a cell, pivots are independently sampled according to the underlying distribution. Whenever a cell contains a large number of sample points, both distributions are close, but they may drift apart when the diameter of the cells go to 0. (iii) In the continuous comparison tree, we stop splitting cells at height $\lfloor p_n \rfloor$, whereas in the discrete setting we stop if there is less than n_0 observations in the current cell. As a consequence, $T^0(p_n)$ is a perfect binary tree: each interior node has exactly 2 children. This is typically not the case for comparison trees.

3.2.2 Consistency

To each realization of $T^0(p_n)$ is associated a partition of $[0,1]^d$ into disjoint cells $A_{1,n}, A_{2,n}, \ldots, A_{2^{p_n},n}$. These cells correspond to the leaf nodes of the continuous comparison-tree $T^0(p_n)$. For any $x \in [0,1]^d$, let $A(x)$ be the cell of $T^0(p_n)$ containing x. Let us assume that the responses $(Y_i)_{1\leq i\leq n}$ are binary labels. We consider the classifier defined by majority vote in each cell, that is,

$$g_n(x) := \begin{cases} 1 & \text{if } \sum_{X_i \in A(x)} \mathbb{1}_{Y_i=1} \geq \sum_{X_i \in A(x)} \mathbb{1}_{Y_i=0} \\ 0 & \text{otherwise.} \end{cases}$$

Define $L_n := \mathbb{P}\left(g_n(X) \neq Y | D_n\right)$. Following Devroye *et al.* (1996), we say that the classifier g_n is *consistent* if

$$\mathbb{E}\left[L_n\right] = \mathbb{P}\left(g_n(X) \neq Y\right) \xrightarrow[n\to+\infty]{} L^\star ,$$

where $L^\star$ is the Bayes error probability. Our main result is the consistency of the classifier associated with the continuous comparison tree truncated to a logarithmic height.

Theorem 9 (Consistency of comparison-based trees) *Under Assumption 3.2.1, the classifier associated to the continuous, truncated tree $T^0(\alpha \log n)$ is consistent for any constant* $0 < \alpha < 1/\log 2$.

In particular, since each individual tree is consistent, a random forest with base tree $T^0(p_n)$ is also consistent. Theorem 9 is a first step towards explaining why comparison-based trees perform well without having access to the representation of the points. Also note that, even though the continuous tree is a simplified version of the discrete tree, they are quite similar and share all important characteristics. In particular, they roughly have the same depth—with high probability, the comparison tree has logarithmic depth according to Theorem 1.

3.2.3 Proof of consistency

Since the construction of $T^0(p_n)$ does not depend on the labels, we can use Theorem 6.1 of Devroye *et al.* (1996) to prove the consistency of the tree. It gives sufficient conditions for classification rules based on space partitioning to be consistent. In particular, we have to show that the partition satisfies two properties: first, the leaf cells should be small enough, so that local changes of the distribution can be detected; second, the leaf cells should contain a sufficiently large number of points so that averaging among the labels makes sense. More precisely, we have to show that

(i) $\operatorname{diam} A(X) \to 0$ in probability, where $\operatorname{diam} A := \sup_{x,y \in A} \delta(x,y)$ is the diameter of A

(ii) $N(X) \to \infty$ in probability, where $N(X)$ is the number of points in the cell containing X.

We first prove the second point (ii) which is easier.
Proof of (ii). According to Lemma 20.1 in Devroye *et al.* (1996) and the remark that follows, it is sufficient to show that the number of regions is $\mathrm{o}(n)$. For each n, by construction, $T^0(\alpha \log n)$ has $2^{\alpha \log n} = n^{\alpha \log 2}$ leaves. Since $\alpha \log 2 < 1$, $2^{\alpha \log n} = \mathrm{o}(n)$ as $n \to +\infty$. □

Proving (i) is much more challenging. The critical part of the proof is to show that, for any cell of the continuous comparison tree, the diameter of its descendants at least k levels below is halved with high probability. More precisely, the following proposition shows that this probability is lower bounded by $1-\beta$, where β is exponentially decreasing in k.

Proposition 10 (Diameter control) *Assume that Assumption 3.2.1 holds. Let C be a cell of T^0 such that $\operatorname{diam} C \leq D$. Then the probability that there exists a descendant of C which is more than k levels below and yet has diameter greater than $D/2$ is at most $N_{f,d}(N_{f,d}+1)\gamma_{f,d}^k/2$, where $0 < N_{f,d}$ and $0 < \gamma_{f,d} < 1$ are constants depending only on d, $f_{\min}$, and $f_{\max}$.*

Proposition 10 is an analogous of Lemma 12 in Dasgupta and Freund (2008). In plain words, it states that for any cell of the continuous comparison tree, the diameter of any descendant at least k levels below is halved with high probability depending on k. Our proof follows closely that of Dasgupta and Freund (2008, Lemma 12), the main difference being in the auxiliary lemmas used to control the probability of certain events, due to the radically different nature of the random tree that we consider.

Proof. Consider a cover of C by balls of radius $r = D/c_r$, with $c_r := 2^6 \cdot d \cdot 25^d \cdot \frac{f_{\max}^2}{f_{\min}^2}$. According to Shalev-Shwartz and Ben-David (2014, Section 27.1), at most

$$\left(\frac{2D\sqrt{d}}{r}\right)^d = \left(2^7 \cdot d^{3/2} \cdot 25^d \cdot \frac{f_{\max}^2}{f_{\min}^2}\right)^d =: N_{d,f}$$

such balls are needed, since $\operatorname{diam} C \leq D$.

Fix any pair of balls B, B' from this cover whose centers are at distance at least $D/2 - r$ from one another. Given any x and y, we say that the split according to $\Delta(x,y)$ is a *good cut* if it cleanly separates B from B', i.e., if $B \subset H_x$ and $B' \subset H_y$ or $B' \subset H_x$ and $B \subset H_y$. If the split cuts both B and B', that is, $B \cap \Delta(x,y) \neq \emptyset$ and $B' \cap \Delta(x,y) \neq \emptyset$, we say that it is a *bad cut*. See Figure 3.2 for illustration.

For any $k \geq 1$, let ρ_k be the probability that there is some cell k levels below C which contains points from both B and B'. We write

$$\begin{aligned}\rho_k &\leq \mathbb{P}\left(\text{top split is a good cut}\right) \cdot 0 + \mathbb{P}\left(\text{top split is a bad cut}\right) \cdot 2\rho_{k-1} \\ &\quad + \mathbb{P}\left(\text{all other split configurations}\right) \cdot \rho_{k-1} \\ &\leq \left(1 + \mathbb{P}\left(\text{top split is a bad cut}\right) - \mathbb{P}\left(\text{top split is a good cut}\right)\right) \rho_{k-1} .\end{aligned}$$

Since $d \geq 1$ and $c_r > 50$, according to Lemma 11 and 12 (stated later in Section 3.2.4),

$$\begin{aligned}\mathbb{P}\left(\text{top split is a bad cut}\right) - \mathbb{P}\left(\text{top split is a good cut}\right) &\leq \frac{f_{\max}}{f_{\min}} \cdot \frac{64d}{c_r} - 2 \cdot \frac{f_{\min}}{f_{\max}} \cdot \frac{1}{25^d} \\ &= -\frac{f_{\min}}{f_{\max}} \cdot \frac{1}{25^d} < 0 .\end{aligned}$$

Set $\gamma_{f,d} := 1 - \frac{f_{\min}}{f_{\max}} \cdot \frac{1}{25^d}$, we just showed that

$$\rho_k \leq \gamma_{f,d} \rho_{k-1} . \tag{3.1}$$

Since $\rho_0 = 1$, we deduce that $\rho_k \leq \gamma_{f,d}^k$. We conclude by a union bound over all the pairs from the cover that are at the prescribed minimum distance from each other. □

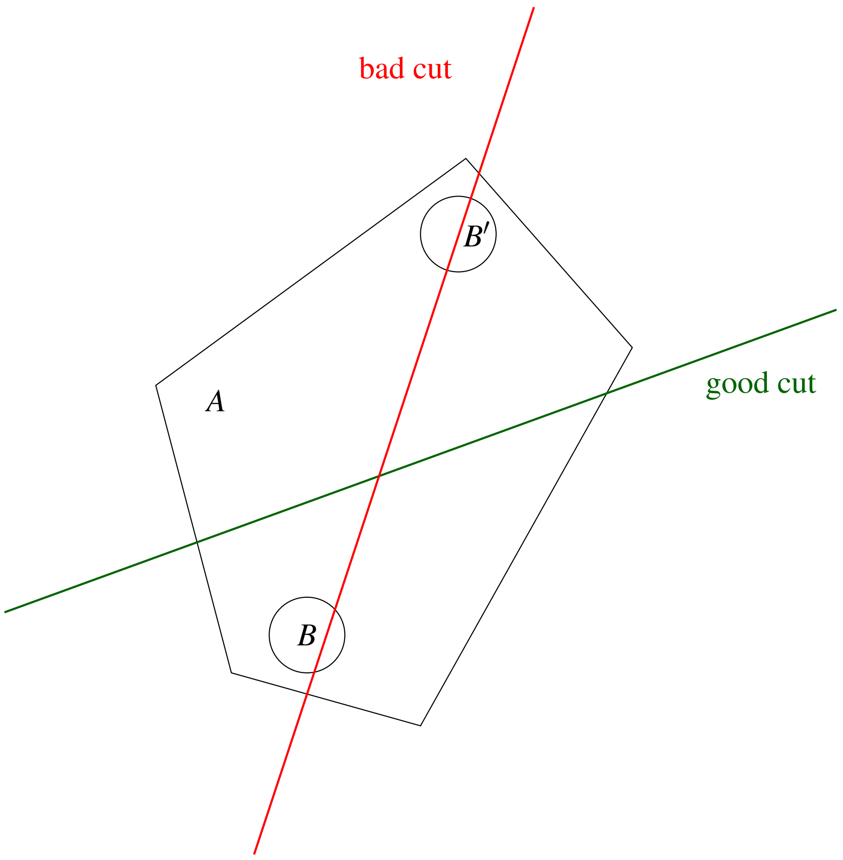


Figure 3.2: Good cuts and bad cuts. The current cell A contains B and B', two faraway balls of small radius—with respect to the diameter of A. A *good cut* (in green) cleanly separates B and B', whereas a *bad cut* (in red) intersects both.

Proof of (i). Let $0 < \varepsilon < 1$, we show that

$$\mathbb{P}\left(\operatorname{diam} A(X) > \varepsilon\right) \longrightarrow 0 \quad \text{when} \quad n \to +\infty .$$

We first notice that

$$\mathbb{P}\left(\operatorname{diam} A(X) > \varepsilon\right) \leq \max_i \mathbb{P}\left(\operatorname{diam} A_{i,n} > \varepsilon\right) .$$

Let A be the leaf of $T^0(p_n)$ with maximal diameter and define $\pi := \left\lceil \frac{\log(\sqrt{d}) - \log \varepsilon}{\log 2} \right\rceil$, so that $\varepsilon > \sqrt{d}/2^{\pi}$. We write

$$\mathbb{P}\left(\operatorname{diam} A > \varepsilon\right) \leq \mathbb{P}\left(\operatorname{diam} A > \frac{\sqrt{d}}{2^{\pi}}\right) .$$

Define $C_1, \dots, C_{p_n}$ the path from $C_0 = [0,1]^d$ to $C_{p_n} = A$ in the tree $T^0(p_n)$. Set $k = \left\lfloor \frac{p_n}{\pi} \right\rfloor$. Set $A^{(0)} = C_0$, $A^{(1)} = C_k$, $A^{(2)} = C_{2k}$, $\dots$, $A^{(\pi-1)} = C_{(\pi-1)k}$ and $A^{(\pi)} = A$. We define the event $E_j := \left\{\operatorname{diam} A^{(j)} > \sqrt{d}/2^j\right\}$. Then

$$\begin{aligned}\mathbb{P}\left(\operatorname{diam} A > \frac{\sqrt{d}}{2^{\pi}}\right) &= \mathbb{P}\left(E_{\pi} | E_{\pi-1}\right) \cdot \mathbb{P}\left(E_{\pi-1}\right) + \mathbb{P}\left(E_{\pi} \middle| E_{\pi-1}^{\mathsf{c}}\right) \cdot \mathbb{P}\left(E_{\pi-1}^{\mathsf{c}}\right) \\ &\qquad\qquad \text{(law of total probability)} \\ &\leq \mathbb{P}\left(E_{\pi} \middle| E_{\pi-1}^{\mathsf{c}}\right) + \mathbb{P}\left(E_{\pi-1}\right) .\end{aligned}$$

Repeating π times this reasoning, and since $\operatorname{diam} A^{(0)} \leq \sqrt{d}$ almost surely, we deduce that

$$\mathbb{P}\left(\operatorname{diam} A^{(\pi)} > \varepsilon\right) \leq \sum_{t=1}^{\pi} \mathbb{P}\left(\operatorname{diam} A^{(t)} > \frac{\sqrt{d}}{2^t} \middle| \operatorname{diam} A^{(t-1)} \leq \frac{\sqrt{d}}{2^{t-1}}\right) .$$

There are always more than k levels between $A^{(tk)}$ and $A^{((t-1)k)}$ by construction. Hence, according to Proposition 10,

$$\mathbb{P}\left(\operatorname{diam} A > \varepsilon\right) \leq \pi \cdot \frac{N_{f,d}(N_{f,d}+1)}{2} \cdot \gamma_{f,d}^{k} .$$

Since $k = O(\log n)$ and $\gamma_{f,d} \in (0,1)$, we can conclude the proof. □

3.2.4 Auxiliary results

In this subsection we present two necessary Lemmas for the proof of Proposition 10. The key in proving Proposition 10 is to show that, for a given cell, the probability of a "good cut" is greater than the probability of a "bad cut." We thus proceed to prove a lower bound for the probability of a good cut and an upper bound for the probability of a bad cut. Since the first cell is the unit cube and all subsequent cells are obtained by intersection with half-spaces, note that any cell of the comparison tree is a full-dimensional convex polytope almost surely. Thus we state and prove our results for such objects.

Good cuts

The following Lemma is an analogous of Lemma 10 in Dasgupta and Freund (2008). It provides a lower bound on the probability of cleanly separating faraway balls.

Lemma 11 (Probability of good cut is lower bounded) *Let Assumption 3.2.1 be true, and $A \subset \mathbb{R}^d$ be a full-dimensional convex polytope such that* $\operatorname{diam} A \leq D < +\infty$. *Let $c_r > 10$ be a constant. Pick any two balls $B := \mathcal{B}(z,r)$ and $B' := \mathcal{B}(z',r)$ such that*

(i) both B and B' intersect A;

(ii) their radius is at most D/c_r;

(iii) the distance between their centers satisfies $\|z - z'\| \geq D/2 - r$.

Then, if X_1 and X_2 are chosen independently from A according to the distribution of X,

$$\mathbb{P}\left(A \cap B \subset A \cap H_{X_1} \text{ and } A \cap B' \subset A \cap H_{X_2}\right) \geq 2\frac{f_{\min}}{f_{\max}}\left(\frac{c_r - 10}{4c_r}\right)^{2d}.$$

As a direct consequence, if $c_r > 50$,

$$\mathbb{P}\left(A \cap B \subset A \cap H_{X_1} \text{ and } A \cap B' \subset A \cap H_{X_2}\right) \geq \frac{f_{\min}}{f_{\max}}\frac{2}{25^d}.$$

While the statement of Lemma 11 is close to that of of Lemma 10 in Dasgupta and Freund (2008), a major difference lies in the quality of the bound we obtain. Indeed, our bound depends exponentially in the dimension, therefore becoming arbitrarily loose for large values of d.

Proof. The proof follows the following scheme. First, we conveniently restrict ourselves to the case where the centers of B and B' both belong to A by geometric arguments. We then use Lemma 15 to lower bound the probability of a good split by the probability that x and y belong to certain balls γ and γ'. We conclude the proof by finding an upper bound for the volume of A and a lower bound for the volume of $\gamma \cap A$. We refer to Figure 3.3 throughout this proof.

Preliminary computations. Set $a := \pi_A(z)$, $a' := \pi_A(z')$, $\beta := \mathcal{B}(a,r)$, and $\beta' := \mathcal{B}(a',r)$. Then, according to Lemma 13, $A \cap B \subset \beta$ and $A \cap B' \subset \beta'$. For any $x, y \in A$ such that $\beta \subset H_x$ and $\beta' \subset H_y$. Since $A \cap B \subset \beta$, we have $A \cap B \subset H_x$. Furthermore, $A \cap B \subset A$, thus $A \cap B \subset A \cap H_x$. A similar reasoning shows that $A \cap B' \subset A \cap H_y$. Hence

$$\mathbb{P}\left(A \cap B \subset A \cap H_{X_1} \text{ and } A \cap B' \subset A \cap H_{X_2}\right) \geq \mathbb{P}\left(\beta \subset H_{X_1} \text{ and } \beta' \subset H_{X_2}\right).$$

Set $\delta := \|a - a'\|$. Since $a \in B$ and $a' \in B'$, by the triangle inequality, $\|a - a'\| \geq \|z - z'\| - 2r$. By hypothesis, $\|z - z'\| \geq D/2 - r$ and $r \leq D/c_r$, thus

$$\left\|a - a'\right\| \geq \frac{D}{2} - 3r \geq \frac{c_r - 6}{2c_r} \cdot D.$$

Define $\rho := \|a - a'\|/2 - r$. We have $\rho \geq \frac{c_r - 10}{4c_r} \cdot D$. In particular, as $c_r > 10$, $\rho > 0$. Then, according to Lemma 15,

$$\mathbb{P}\left(\beta \subset H_{X_1} \text{ and } \beta' \subset H_{X_2}\right) \geq \mathbb{P}\left(X_1 \in \gamma \text{ and } X_2 \in \gamma' \text{ or } X_2 \in \gamma \text{ and } X_1 \in \gamma'\right),$$

where $\gamma := \mathcal{B}(a, \rho)$ and $\gamma' := \mathcal{B}(a', \rho)$. Since X_1 and X_2 are independent and identically distributed and $\gamma \cap \gamma' = \emptyset$,

$$\mathbb{P}\left(X_1 \in \gamma \text{ and } X_2 \in \gamma' \text{ or } X_2 \in \gamma \text{ and } X_1 \in \gamma'\right) \geq 2\,\mathbb{P}\left(X_1 \in \gamma\right)\mathbb{P}\left(X_2 \in \gamma'\right).$$

Since we sample X_1 and X_2 according to the law of X restricted to A and since Assumption 3.2.1 holds,

$$\mathbb{P}\left(X_1 \in \gamma\right) \geq \frac{f_{\min}}{f_{\max}} \frac{\mathrm{Vol}_d\left(\gamma \cap A\right)}{\mathrm{Vol}_d\left(A\right)}.$$

In the next paragraphs, we find an upper bound for $\mathrm{Vol}_d(A)$ and a lower bound for $\mathrm{Vol}_d(\gamma \cap A)$. We will see that the latter also holds for γ'.

Upper bound for $\mathrm{Vol}_d(A)$. We refer to Figure 3.3 for the geometric constructions that follow. Let us first define $\Omega := \Delta(a, a') \cap A$ the intersection between the convex polytope A and the hyperplane $\Delta(a, a')$. We also need to define Π the set of all

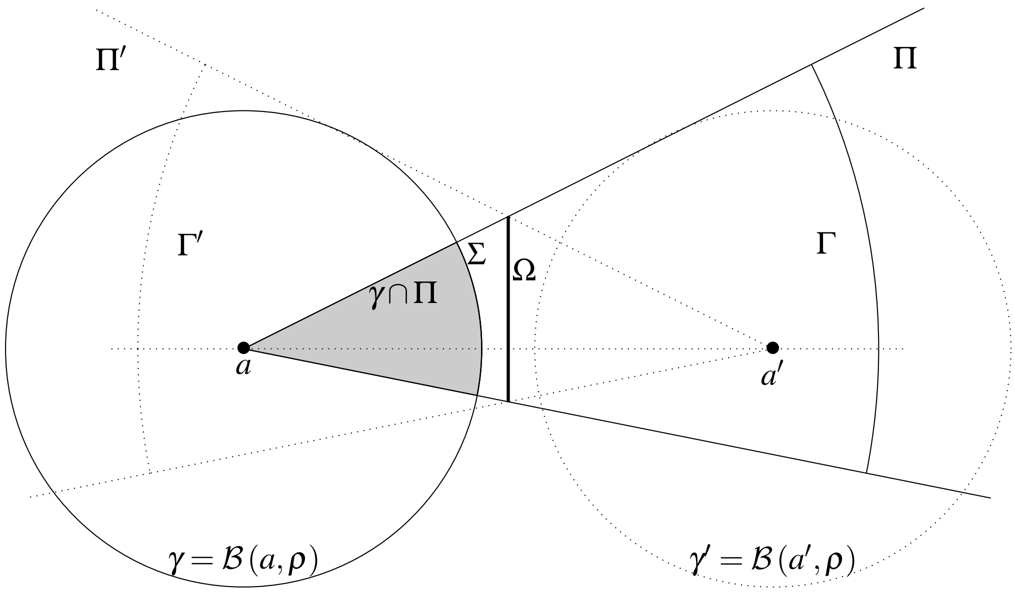


Figure 3.3: Construction of Ω, Π and Σ. The central thick line represents Ω, the intersection between A and the hyperplane $\Delta(a,a')$. The half-cone Π is the union for all $\omega \in \Omega$ of the half-lines $[a,\omega)$. Finally, the spherical cap Σ is defined as the intersection between $\mathcal{S}(a,\rho)$ and Π. In dotted lines we draw the counter-parts of these objects for a'. The gray area represents $\gamma \cap \Pi$.

half-lines going from a through Ω, namely,

$$\Pi := \left\{a + t(w-a) \mid \omega \in \Omega \text{ and } t > 0\right\},$$

and the conic section $\Gamma := \mathcal{B}(a, \operatorname{diam} A) \cap \Pi$. We claim that $A \cap H_{a'} \subset \Gamma$. Indeed, let $\xi \in A \cap H_{a'}$. Since $\xi \in H_{a'}$, $[a, \xi]$ intersects $\Delta(a, a')$ in a unique point, say ζ. By convexity, the segment $[a, \xi]$ is contained into A. In particular, $\zeta \in A$. Thus $\zeta \in \Delta(a, a') \cap A = \Omega$, and

$$\xi = a + \frac{\|\xi - a\|}{\|\zeta - a\|}(\zeta - a) \in A\,.$$

Moreover, since $\xi \in A$,

$$\|a - \xi\| \leq \sup_{s \in A} \|a - s\| = \operatorname{diam} A\,,$$

and $\xi \in \mathcal{B}(a, \operatorname{diam} A)$. A similar reasoning shows that $A \cap H_a \subset \Gamma'$, where Γ' is the symmetric of Γ with respect to $\Delta(a, a')$. Therefore,

$$\operatorname{Vol}_d(A) \leq 2 \operatorname{Vol}_d(\Gamma)\,.$$

Define the hyperspherical cap $\Sigma := \mathcal{S}(a, \rho) \cap \Pi$. Then we can express the volume of the conic section Γ as

$$\operatorname{Vol}_d(\Gamma) = \frac{\operatorname{Vol}_{d-1}(\Sigma)}{\operatorname{Vol}_{d-1}(\mathcal{S}(a, \rho))} \operatorname{Vol}_d(\mathcal{B}(a, \operatorname{diam} A))\,,$$

which leads to

$$\operatorname{Vol}_d(A) \leq \frac{2 \operatorname{Vol}_{d-1}(\Sigma)}{\operatorname{Vol}_{d-1}(\mathcal{S}(a, \rho))} \operatorname{Vol}_d(\mathcal{B}(a, \operatorname{diam} A))\,. \tag{3.2}$$

Lower bound for $\operatorname{Vol}_d(\gamma \cap A)$**.** By convexity, $\gamma \cap \Pi \subset \gamma \cap A$. Moreover,

$$\operatorname{Vol}_d(\gamma \cap \Pi) = \frac{\operatorname{Vol}_{d-1}(\Sigma)}{\operatorname{Vol}_{d-1}(\mathcal{S}(a, \rho))} \operatorname{Vol}_d(\mathcal{B}(a, \rho))\,.$$

Hence the following lower bound holds:

$$\operatorname{Vol}_d(\gamma \cap A) \geq \frac{\operatorname{Vol}_{d-1}(\Sigma)}{\operatorname{Vol}_{d-1}(\mathcal{S}(a, \rho))} \operatorname{Vol}_d(\mathcal{B}(a, \rho))\,. \tag{3.3}$$

Conclusion. Putting together Eq. (3.2) and (3.3), we obtain

$$\mathbb{P}(X_1 \in \gamma) \geq \frac{f_{\min}}{f_{\max}} \frac{\operatorname{Vol}_d(\mathcal{B}(a,\rho))}{\operatorname{Vol}_d(\mathcal{B}(a, \operatorname{diam} A))} = \frac{f_{\min}}{f_{\max}} \left(\frac{\rho}{\operatorname{diam} A}\right)^d .$$

Since $\rho \geq (c_r - 10)/(4Dc_r)$ and $\operatorname{diam} A \leq D$, we deduce that

$$\mathbb{P}(X_1 \in \gamma) \geq \frac{f_{\min}}{f_{\max}} \left(\frac{c_r - 10}{4c_r}\right)^d .$$

We conclude the proof by using the preliminary computations. □

Bad cuts

We now focus on the probability of a “bad split,” that is, $\Delta(x,y)$ intersects both $\mathcal{B}(z,r)$ and $\mathcal{B}(z',r)$. The following result is an analogous of Lemma 11 in Dasgupta and Freund (2008).

Lemma 12 (Probability of bad cut is upper bounded) *Let Assumption 3.2.1 be true, and $A \subset \mathbb{R}^d$ be a full-dimensional convex polytope such that* $\operatorname{diam} A \leq D < +\infty$*. Let $c_r > 10$ be a constant. Pick any two balls $B := \mathcal{B}(z,r)$ and $B' := \mathcal{B}(z',r)$ such that*

(i) both B and B' intersect A;

(ii) their radius is at most D/c_r;

(iii) the distance between their centers satisfies $\|z - z'\| \geq D/2 - r$.

Then, if X_1 and X_2 are chosen independently from A according to the distribution of X,

$$\mathbb{P}\left(A \cap B \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } A \cap B' \cap \Delta(X_1,X_2) \neq \emptyset\right) \leq \frac{f_{\max}}{f_{\min}} \frac{32dc_r}{(c_r-2)(c_r-6)} .$$

As a direct consequence, if $c_r > 15$,

$$\mathbb{P}\left(A \cap B \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } A \cap B' \cap \Delta(X_1,X_2) \neq \emptyset\right) \leq \frac{f_{\max}}{f_{\min}} \frac{64d}{c_r} .$$

Note that, as in Lemma 11, the bound we obtain worsens as the dimension increases.
Proof. We first restrict ourselves to the case where the centers of B and B' both belong to A with the same argument than in the proof of Lemma 11. Namely, define

$a := \pi_A(z)$, $a' := \pi_A(z')$, $\beta := \mathcal{B}(a,r)$, $\beta' := \mathcal{B}(a',r)$. According to Lemma 13, $A \cap B \subset \beta$ and $A \cap B' \subset \beta'$. Thus

$$\begin{aligned}&\mathbb{P}\left(A \cap B \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } A \cap B' \cap \Delta(X_1,X_2) \neq \emptyset\right)\\ &\qquad \leq \mathbb{P}\left(\beta \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } \beta' \cap \Delta(X_1,X_2) \neq \emptyset\right).\end{aligned}$$

For any $x \in \mathbb{R}^d$, define B_x the set of points y such that $\Delta(x,y)$ is a bad cut, that is,

$$B_x := \left\{y \in \mathbb{R}^d \mid \beta \cap \Delta(x,y) \neq \emptyset \text{ and } \beta' \cap \Delta(x,y) \neq \emptyset\right\}.$$

Then, since we sample X_1 according to the law of X restricted to A and since we assume Assumption 3.2.1 to be true,

$$\mathbb{P}\left(\beta \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } \beta' \cap \Delta(X_1,X_2) \neq \emptyset\right) \leq \frac{f_{\max}}{f_{\min}} \frac{\mathbb{E}\left[\mathrm{Vol}_d\left(B_{X_1} \cap A\right)\right]}{\mathrm{Vol}_d\left(A\right)},$$

where the expectation is relative to the random variable X_1.

Upper bound for $\mathrm{Vol}_d\left(B_x \cap A\right)$**.** Let $x \in A$ and $y \in B_x$. By Lemma 14,

$$\begin{cases}(\|x-a\|-2r)^+ & \leq \|y-a\| \leq \|x-a\|+2r\\ (\|x-a'\|-2r)^+ & \leq \|y-a'\| \leq \|x-a'\|+2r.\end{cases}$$

Equivalently, $B_x \subset \mathcal{A}(a,r_1,r_2) \cap B_x \subset \mathcal{A}(a',r_1',r_2')$, where we defined $r_1 := \|x-a\| - 2r$, $r_2 := \|x-a\|+2r$, $r_1' := \|x-a'\|-2r$ and $r_2' := \|x-a'\|+2r$. Recall that $\mathcal{A}(a,r_1,r_2) = \mathcal{B}(a,r_2)$ whenever $r_1 \leq 0$. See Figure 3.4 for an illustration.

For any $\xi \in (a,a')$, denote by D_ξ the hyperplane orthogonal to (a,a') and passing through ξ. According to Lemma 17, the width of $\mathcal{A}(a,r_1,r_2) \cap \mathcal{A}(a',r_1',r_2')$ along the (a,a') axis is upper bounded by $16D/(c_r-2)$, hence there exists ξ^- and $\xi^+ \in (a,a')$ such that $\|\xi^+ - \xi^-\| \leq 16D/(c_r-2)$ and $B_x \cap A$ is contained between D_{ξ^-} and D_{ξ^+}. For each $\xi \in (a,a')$, set $\Omega_\xi := D_\xi \cap A$. There exists $\xi^\star \in [\xi^-,\xi^+]$ such that $\mathrm{Vol}_{d-1}\left(\Omega_{\xi^\star}\right)$ is maximal, and

$$\begin{aligned}\mathrm{Vol}_d\left(B_x \cap A\right) &= \int_{\xi \in [\xi^-,\xi^+]} \mathrm{Vol}_{d-1}\left(\Omega_\xi\right) \mathrm{d}\xi\\ &\leq \left\|\xi^+ - \xi^-\right\| \cdot \mathrm{Vol}_{d-1}\left(\Omega_{\xi^\star}\right)\\ \mathrm{Vol}_d\left(B_x \cap A\right) &\leq \frac{16}{c_r-2} \cdot D\,\mathrm{Vol}_{d-1}\left(\Omega_{\xi^\star}\right). \qquad (3.4)\end{aligned}$$

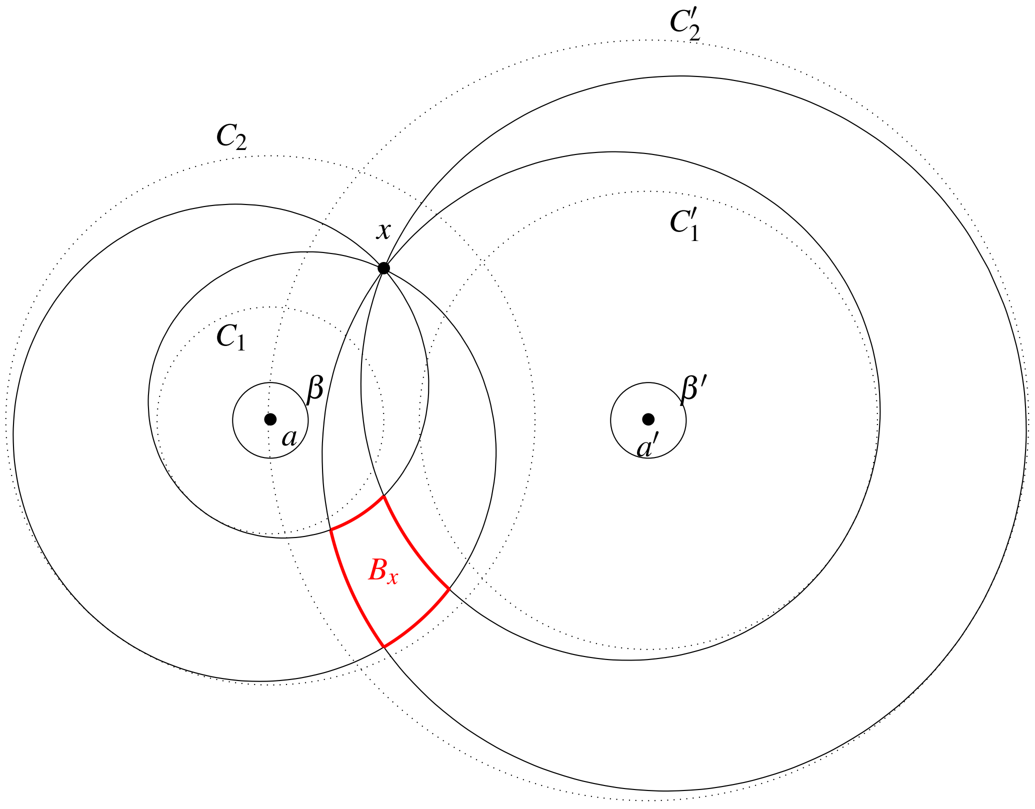


Figure 3.4: Sketch of B_x in $\mathbb{R}^2$. For a fixed x, B_x is the set of all y such that $\Delta(x,y)$ cuts both β and β' (border marked in red). We show that B_x is the intersection of two geometric loci (solid lines border). In particular, B_x is included in the intersection of two annuli (border in dotted lines).

Lower bound for $\mathrm{Vol}_d(A)$. Suppose that $\xi^\star$ belongs to the segment $[a,a']$. By convexity, A contains the (disjoint) union of the two hyperpyramids of apexes a and a' with $(d-1)$-dimensional basis $\Omega_{\xi^\star}$, which we denote by Q and Q'. Therefore,

$$\begin{aligned}\mathrm{Vol}_d(A) &\geq \mathrm{Vol}_d(Q) + \mathrm{Vol}_d(Q') \\ &= \frac{\|a-\xi^\star\| \mathrm{Vol}_{d-1}(\Omega_{\xi^\star})}{d} + \frac{\|a'-\xi^\star\| \mathrm{Vol}_{d-1}(\Omega_{\xi^\star})}{d} \\ &= \frac{\delta \mathrm{Vol}_{d-1}(\Omega_{\xi^\star})}{d}.\end{aligned}$$

Since $\delta \geq (c_r-6)D/(2c_r)$,

$$\mathrm{Vol}_d(A) \geq \frac{c_r-6}{2dc_r} \cdot D \mathrm{Vol}_{d-1}(\Omega_{\xi^\star}). \tag{3.5}$$

A similar reasoning holds whenever $\xi^\star$ does not belong to $[a,a']$.

Conclusion. Putting together Eq. (3.4) and (3.5), we obtain

$$\mathbb{P}\left(\beta \cap \Delta(X_1,X_2) \neq \emptyset \text{ and } \beta' \cap \Delta(X_1,X_2) \neq \emptyset\right) \leq \frac{f_{\max}}{f_{\min}} \frac{32dc_r}{(c_r-2)(c_r-6)},$$

which concludes the proof. □

Note that in the plane defined by a, a' and x, we can actually describe precisely the shape of the curves defining the border of B_x—see Figure 3.4. These curves correspond to the images of x by all the symmetries with respect to a line tangent to β or β'. Individually, they are called the *orthotomics of a circle*, or second caustic (Lawrence, 2013, p. 60).

3.2.5 Technical results

This first lemma is used in the proofs of Lemma 11 and 12 to deal with cases where the center of B or B' does not belong to A. See Figure 3.5 for an illustration of such a situation.

Lemma 13 (Construction of β) *Let $A \subset \mathbb{R}^d$ be a convex compact set and $\mathcal{B}(z,r)$ be a ball that intersects A. Define $\beta := \mathcal{B}(\pi_A(z),r)$. Then $A \cap B \subset \beta$.*

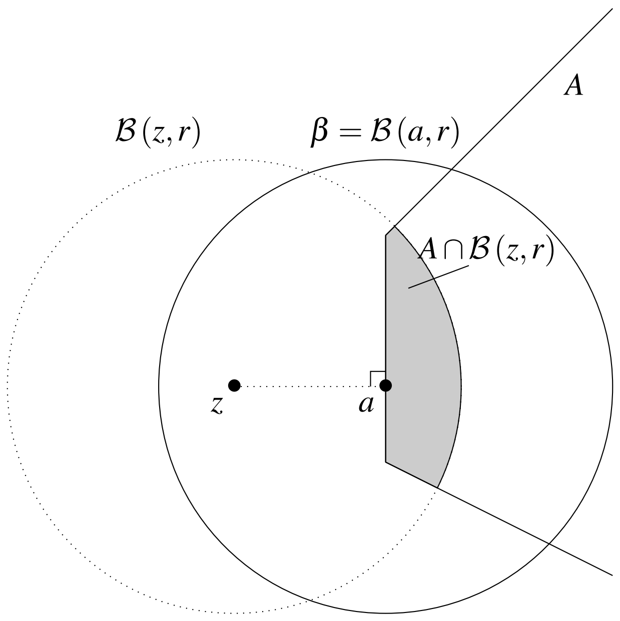


Figure 3.5: Construction of β. The point a is the image of z by the orthogonal projection on A. The ball β has the same radius than $\mathcal{B}(z,r)$ and contains $A \cap \mathcal{B}(z,r)$, which is marked in gray.

Proof. Set $a := \pi_A(z)$. Let x be an element of $A \cap B$. Then,

$$\begin{aligned}
\|x-a\|^2 &= \langle x-a, x-a\rangle \\
&= \langle x-z+z-a, x-z+z-a\rangle \\
&= \|x-z\|^2 + 2\langle x-z, z-a\rangle + \|z-a\|^2 \\
\|x-a\|^2 &= \|x-z\|^2 + 2\langle x-a, z-a\rangle - \|z-a\|^2 \,.
\end{aligned}$$

Since π_A is a the orthogonal projection, given that $x \in A$, we have $\langle x-a, z-a\rangle \leq 0$. Moreover, $\|z-a\| \geq 0$, thus $\|x-a\|^2 \leq \|x-z\|^2$. But x also belongs to B, hence $\|x-z\| \leq r$. As a consequence, $\|x-a\| \leq r$, that is, $x \in \beta$. □

The next lemma shows that, for a given x, the set of every possible y such that $\Delta(x,y)$ intersects $\mathcal{B}(a,r)$ is contained into an annulus centered in a. We refer to Figure 3.6 for an illustration.

Lemma 14 (Localization of B_x) *Let $a, x \in \mathbb{R}^d$ and $r > 0$. Then, for any $y \in \mathbb{R}^d$ such that $\Delta(x,y) \cap \mathcal{B}(a,r)$ is non-empty,*

$$(\|x-a\| - 2r)^+ \leq \|y-a\| \leq \|x-a\| + 2r \,.$$

Proof. Let $y \in \mathbb{R}^d$ such that $\Delta(x,y) \cap \mathcal{B}(a,r)$ is non-empty. In particular, there exists $b \in \mathbb{R}^d$ such that $\|y-b\| = \|x-b\|$ and $\|a-b\| \leq r$. By the triangle inequality,

$$|\|y-a\| - \|y-b\|| \leq \|a-b\| \leq r\,.$$

Hence

$$\begin{cases} \|y-a\| \leq r + \|y-b\| = r + \|x-b\| \\ \|y-a\| \geq -r + \|y-b\| = -r + \|x-b\|\,. \end{cases}$$

Since $|\|x-b\| - \|a-b\|| \leq \|x-a\|$ (again by the triangle inequality), we have

$$\begin{cases} \|y-a\| \leq \|x-a\| + 2r \\ \|y-a\| \geq \|x-a\| - 2r\,. \end{cases}$$

□

We now present a result stating that, for any two points $a, a' \in \mathbb{R}^d$, there exists a simple set of possible x and y such that balls with center a and a' are well-separated by $\Delta(x,y)$. It is the key element in the proof of Lemma 11.

Lemma 15 (Sufficient conditions for a good cut) *Let $a, a' \in \mathbb{R}^d$, and let $0 < r < \|a-a'\|/2$. Set $\rho := \|a-a'\|/2 - r$. Then, for any $x \in \mathcal{B}(a,\rho)$ and $y \in \mathcal{B}(a',\rho)$, we have $\mathcal{B}(a,r) \subset H_x$ and $\mathcal{B}(a',r) \subset H_y$.*

Remark 16 *Note that Lemma 15 holds in any metric space $(\mathcal{X},\delta)$ since the proof only uses the triangle inequality.*

Proof. We refer to Figure 3.7 for this proof. We have to prove that for any $s \in \mathcal{B}(a,r)$, $\delta(s,x) \leq \delta(s,y)$ (the case $t \in \mathcal{B}(a',r)$ is identical up to notations). We first write

$$\delta(s,x) \leq \delta(s,a) + \delta(a,x) \leq r + \rho = \delta\left(a,a'\right)/2\,,$$

where we used (i) the triangle inequality, (ii) $s \in \mathcal{B}(a,r)$ and $x \in \mathcal{B}(a,\rho)$, (iii) the definition of ρ. Then,

$$\delta\left(a,a'\right) \leq \delta(a,y) + \delta\left(a',y\right) \leq \delta(a,y) + \rho\,,$$

where we used (i) triangle inequality, (ii) $y \in \mathcal{B}(a',\rho)$. Thus $\delta(a,y) \geq \delta(a,a') - \rho$. Moreover,

$$\delta(a,y) \leq \delta(a,s) + \delta(s,y) \leq r + \delta(s,y)\,,$$

where we used (i) triangle inequality, (ii) $s \in \mathcal{B}(a,r)$. Combining the two, we get

$$\delta(s,y) \geq \delta\left(a,a'\right) - (r+\rho) = \delta\left(a,a'\right)/2\,.$$

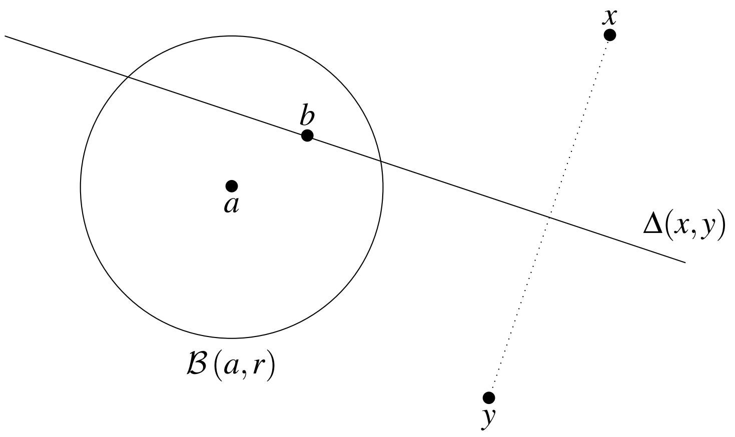


Figure 3.6: B_x is included in the intersection of two annuli. As in the proof of Lemma 14, a and x are fixed, and y is such that $\Delta(x,y)$ intersects $\mathcal{B}(a,r)$. Then y belongs to an annulus of radii $(\|x-a\|-2r)^+$ and $\|x-a\|+2r$.

Therefore $\delta(s,y) \geq \delta(s,x)$ and we can conclude. □

Finally, we state and prove a technical lemma used in the proof of Lemma 12 to control the size of the intersection of two annuli.

Lemma 17 (B_x has small width) *Assume the set of hypotheses of Lemma 12 and define r_1, r_2, r_1' and r_2' as in the proof of Lemma 12. Then there exist two hyperplanes L_x and L_x', orthogonal to (a,a'), such that the intersection of $\mathcal{A}(a,r_1,r_2)$ and $\mathcal{A}(a',r_1',r_2')$ is included between L_x and L_x'. Additionally,*

$$\delta\left(L_x,L_x'\right) \leq \frac{16D}{c_r-2}. \tag{3.6}$$

Even though the statement of Lemma 17 may seem intuitive at first sight (since the intersection is contained in two annuli of width $O(D/c_r)$, one would expect its width to be of the same order), we do not know of a simpler proof. We believe that it is necessary to describe precisely the intersection of the two annuli depending on the radii in order to make sure that the situation where the two annuli are overlapping is excluded. Indeed, in this case the width of the intersection is *not* bounded by a quantity depending on D/c_r but rather on D, since it has the same order than the diameter of the annuli.
Proof. By rotational symmetry around (a,a'), it suffices to prove the result in a 2-plane containing a and a'. Hence from now on we work in the plane P defined by the triple (x,a,a'). We first describe the shape of the intersection between the two

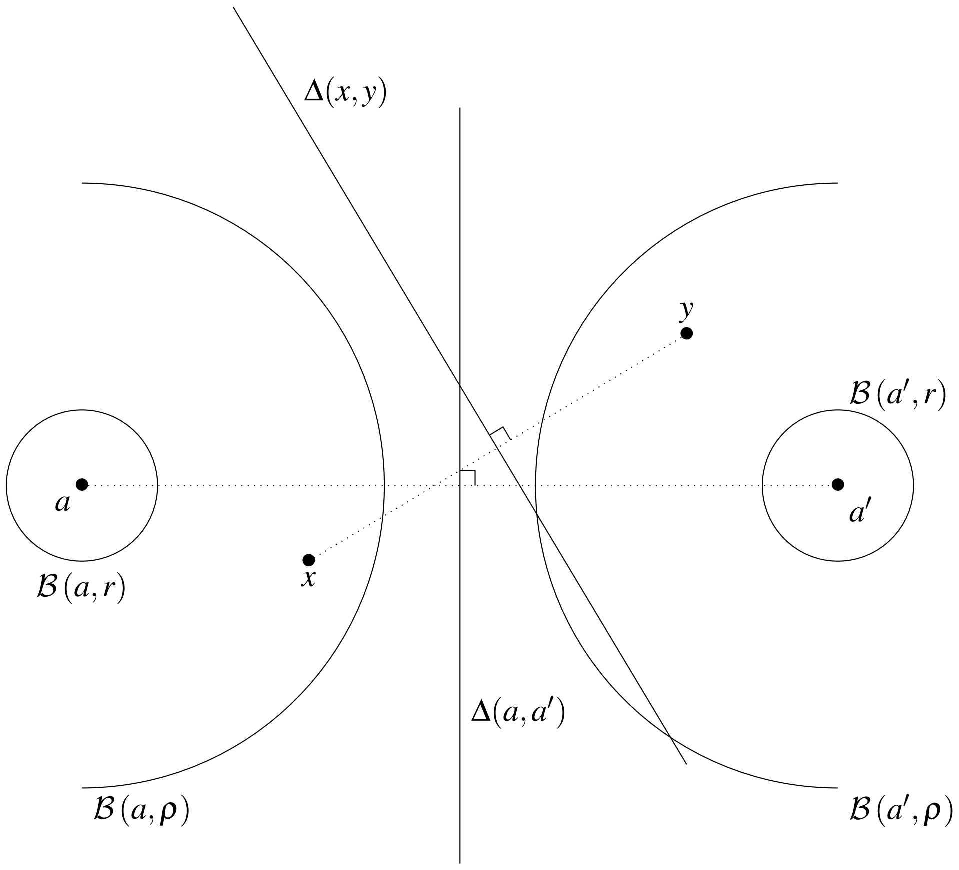


Figure 3.7: Guaranteed good cut. Set $a, a' \in \mathbb{R}^d$ and $\rho = \|a - a'\|/2 - r$. Then, for any $x \in \mathcal{B}(a,\rho)$ and $y \in \mathcal{B}(a',\rho)$, the hyperplane $\Delta(x,y)$ separates cleanly $\mathcal{B}(a,r)$ from $\mathcal{B}(a',r)$.

annuli depending on the position of x relatively to a and a'. Then, in each case, we bound the width of the intersection in the direction of the (a,a') axis.

Shape of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$. Let us equip P with an orthogonal frame such that $a=(0,0)$, $a'=(\delta,0)$ and $x=(x_1,x_2)$. The width of the intersection is invariant by line symmetry with respect to $\Delta(a,a')$ and (a,a'), thus we can restrict our analysis to the quadrant defined by $x_1\leq\delta/2$ and $x_2>0$. In particular, $\|x-a\|\leq\|x-a'\|$. Define $C_i:=\mathcal{C}(a,r_i)$ and $C_i':=\mathcal{C}(a',r_i')$ for $i\in\{1,2\}$. The shape of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ depends on the mutual intersections between C_1,C_2,C_1' and C_2'. Recall that $\mathcal{C}(a,\rho)\cap\mathcal{C}(a',\rho)\neq\emptyset$ if, and only if,

$$\left|\rho-\rho'\right|\leq\delta\leq\rho+\rho'\,.$$

We now proceed to describe these intersection depending on the position of x relatively to a and a'.

- Since $r>0$, $r_1<r_2$ and $r_1'<r_2'$ and thus $C_1\cap C_2=C_1'\cap C_2'=\emptyset$;
- By the triangle inequality, $|r_2-r_2'|=|\|x-a\|-\|x-a'\||\leq\delta$ and $r_2+r_2'=\|x-a\|+\|x-a'\|+4r\geq\delta$, hence $C_2\cap C_2'$ is always non-empty;
- By the triangle inequality, $|r_1-r_1'|=|\|x-a\|-\|x-a'\||\leq\delta$. Hence $C_1\cap C_1'$ is non-empty if, and only if, $r_1+r_1'\geq\delta$, that is, $\|x-a\|+\|x-a'\|\geq\delta-4r$. The border is an ellipse with focal points a,a' and semi-major axis $(\delta+4r)/2$.
- By the triangle inequality, $r_1+r_2'=\|x-a\|+\|x-a'\|\leq\delta$. Since $\|x-a\|\leq\|x-a'\|$, $|r_1-r_2'|=4r-\|x-a\|+\|x-a'\|$. Thus $C_1\cap C_2'$ is non-empty if, and only if, $\|x-a'\|-\|x-a\|\leq\delta-4r$. The border is a hyperbola with focal points a,a' and semi-major axis $(\delta-4r)/2$.
- Using the triangle inequality, $r_1'+r_2=\|x-a'\|+\|x-a\|\geq\delta$. Moreover, $|r_1'-r_2|=|\|x-a'\|-\|x-a\|-4r|$. Again, the triangle inequality yields $\|x-a'\|-\|x-a\|\leq\delta\leq\delta+4r$. On the other side, $\|x-a\|-\|x-a'\|\leq 0\leq\delta-4r$ because $r<\delta/4$. Hence $C_2\cap C_1'$ is always non-empty.

The different cases are summarized in Figure 3.8, and we provide a visual depiction of the intersection for each case in Figure 3.9. Note that in Case III, $\|x-a\|\leq 2r$ is a possibility, implying $r_1<0$. In this event, we see in Figure 3.9 that the extremal points are the same.

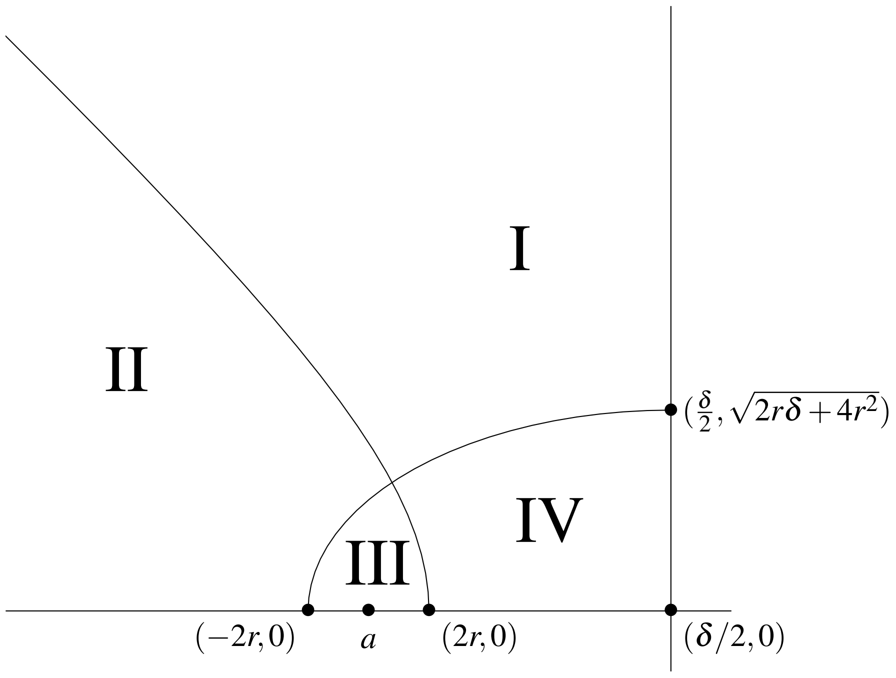


Figure 3.8: Shape of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r'_1,r'_2)$ (I). Depending on the relative position of x with respect to a and a', the shape of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r'_1,r'_2)$ changes. *Case I:* $C_1\cap C'_1$ and $C_1\cap C'_2$ are both non-empty. *Case II:* $C_1\cap C'_1$ is non-empty, whereas $C_1\cap C'_2$ is. *Case III:* $C_1\cap C'_1$ and $C_1\cap C'_2$ are both empty. *Case IV:* $C_1\cap C'_2$ is non-empty whereas $C_1\cap C'_1$ is. The shape of B_x as well as $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r'_1,r'_2)$ in this last case is depicted in Figure 3.4.

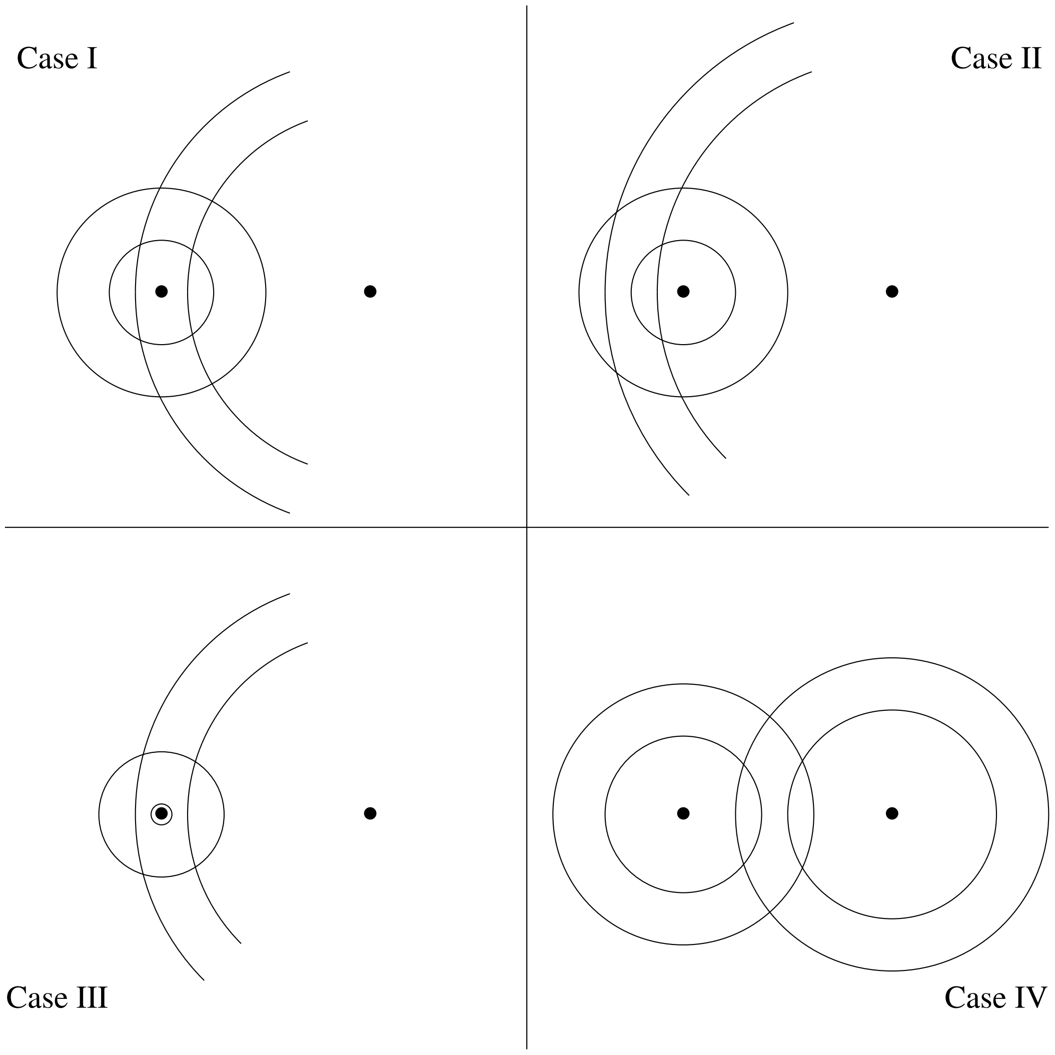


Figure 3.9: Shape of $\mathcal{A}(a,r_1,r_2) \cap \mathcal{A}(a',r_1',r_2')$ (II). For each case described in Figure 3.8, we sketch $\mathcal{A}(a,r_1,r_2) \cap \mathcal{A}(a',r_1',r_2')$. Note that the points realizing the minimum and maximum abscissa in each case are different, leading to different bounds on the width of $\mathcal{A}(a,r_1,r_2) \cap \mathcal{A}(a',r_1',r_2')$.

Width of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$. For each case, we show that Eq. (3.6) holds. Recall that we assumed $r/\delta \le 2/(c_r-2)$ and $\operatorname{diam}A \le D$. We will use the fact that

$$\frac{\|x-a\|^2-\|x-a'\|^2}{2\delta} = \frac{x_1^2+x_2^2-x_1^2+2\delta x_1+\delta^2+x_2^2}{2\delta} = x_1-\frac{\delta}{2}\,.$$

- Case I: The left-most points of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ belong to $C_1\cap C_2'$. We solve
$$\begin{cases} \xi_1^2+\xi_2^2 & = r_1^2 = (\|x-a\|-2r)^2 \\ (\xi_1-\delta)^2+\xi_2^2 & = r_2'^2 = (\|x-a'\|+2r)^2\,. \end{cases}$$
and find
$$\xi_1 = x_1 - \frac{2r}{\delta}\left(\|x-a\|+\left\|x-a'\right\|\right)\,.$$
The right-most points of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ belong to $C_2\cap C_1'$. We solve
$$\begin{cases} \zeta_1^2+\zeta_2^2 & = r_2^2 = (\|x-a\|+2r)^2 \\ (\zeta_1-\delta)^2+\zeta_2^2 & = r_1'^2 = (\|x-a'\|-2r)^2\,. \end{cases}$$
and find
$$\zeta_1 = x_1 + \frac{2r}{\delta}\left(\|x-a\|+\left\|x-a'\right\|\right)\,.$$
Thus the width of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ along (a,a') is given by
$$|\zeta_1-\xi_1| = \frac{4r}{\delta}\left(\|x-a\|+\left\|x-a'\right\|\right) \le \frac{16D}{c_r-2}\,.$$

- Case II: The left-most point of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ belongs to $C_2'\cap(a,a')$, and we have
$$\xi_1 = \delta - r_2' = \delta - \left\|x-a'\right\| - 2r\,.$$
The right-most points belongs to $C_2\cap C_1'$, and we have, as in Case I,
$$\zeta_1 = x_1 + \frac{2r\left(\|x-a\|+\|x-a'\|\right)}{\delta}\,.$$
Thus the width of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ along (a,a') is given by
$$|\zeta_1-\xi_1| = \frac{2r}{\delta}\left(\|x-a\|+\left\|x-a'\right\|\right) + \left\|x-a'\right\| + x_1 - \delta - 2r\,.$$
The equation of the asymptotes of the hyperbola $\|x-a'\|-\|x-a\| = \delta-4r$

are given by

$$x_2 = \pm \frac{2\sqrt{2r\delta - 4r^2}}{\delta - 4r}(x_1 - \delta/2)\,,$$

and considering the lines parallel to these asymptotes passing through $(\delta, 0)$ we deduce that, in case II,

$$\frac{x_2^2}{(x_1-\delta)^2} \le \frac{4(2r\delta - 4r^2)}{(\delta - 4r)^2} \le \frac{8r\delta}{(\delta-4r)^2} \le 8\frac{r}{\delta}\frac{1}{\left(1-4\frac{r}{\delta}\right)^2} \le \frac{16\,(c_r-10)^2}{(c_r-2)^3}\,.$$

Thus

$$\left\|x-a'\right\| = \sqrt{(x_1-\delta)^2 + x_2^2} \le |x_1 - \delta| \sqrt{1 + \frac{16\,(c_r-10)^2}{(c_r-2)^3}}\,,$$

and we have

$$\left\|x-a'\right\| + x_1 - \delta \le |x_1-\delta| \left(\sqrt{1+\frac{16\,(c_r-10)^2}{(c_r-2)^3}} - 1\right) \le \frac{8D\,(c_r-10)^2}{(c_r-2)^3}\,,$$

where we used $\sqrt{1+x^2} - 1 \le x/2$ in the last inequality. Finally,

$$|\zeta_1 - \xi_1| \le \frac{8D(c_r-10)^2}{(c_r-2)^3} + \frac{8D}{c_r-2}\,.$$

- Case III: The left-most point of $\mathcal{A}(a, r_1, r_2) \cap \mathcal{A}(a', r_1', r_2')$ belongs to $C_2 \cap C_2'$. We solve

$$\begin{cases} \xi_1^2 + \xi_2^2 & = r_2^2 = (\|x-a\| + 2r)^2 \\ (\xi_1 - \delta)^2 + \xi_2^2 & = r_2'^2 = (\|x-a'\| + 2r)^2\,, \end{cases}$$

which yields

$$\xi_1 = x_1 + \frac{2r}{\delta}\left(\|x-a\| - \left\|x-a'\right\|\right)\,.$$

The right-most points belongs to $C_2 \cap C_1'$, and we have, as in Case I,

$$\zeta_1 = x_1 + \frac{2r\,(\|x-a\| + \|x-a'\|)}{\delta}\,.$$

Thus the width of $\mathcal{A}(a, r_1, r_2) \cap \mathcal{A}(a', r_1', r_2')$ along (a, a') is given by

$$|\zeta_1 - \xi_1| = \frac{4r}{\delta}\left\|x-a'\right\| \le \frac{8D}{c_r-2}\,.$$

- Case IV: as in Case I, the width of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ is given by

$$|\xi_1-\zeta_1|=\frac{4r\left(\|x-a\|+\|x-a'\|\right)}{\delta}\,.$$

  Since in this case $\|x-a\|+\|x-a'\|\leq\delta-4r$, we have

$$|\xi_1-\zeta_1|\leq 4r\leq\frac{4D}{c_r}\,.$$

Overall, since $c_r>10$, we have shown that the width of $\mathcal{A}(a,r_1,r_2)\cap\mathcal{A}(a',r_1',r_2')$ along (a,a') is upper bounded by $16D/(c_r-2)$. □

3.3 Experiments

In this section, we first examine comparison-based random forests in the Euclidean setting. Secondly, we apply the CompRF method to non-Euclidean datasets with a general metric available. Finally, we run experiments in the setting where we are only given triplet comparisons.

3.3.1 Euclidean setting

Here we examine the performance of CompRF on classification and regression tasks in the Euclidean setting, and compare it against CART random forests as well as the KNN classifier as a baseline. As distance function for KNN and CompRF we use the standard Euclidean distance. Since the CompRF only has access to distance comparisons, the amount of information it uses is considerably lower than the information available to the CART forest. Hence, the goal of this experiment is not to show that comparison-based random forests can perform better, but rather to find out whether the performance is still acceptable.

To emphasize the role of supervised pivot selection, we report the performance of the unsupervised CompRF algorithm in classification tasks as well. The tree structure in the unsupervised CompRF chooses the pivot points uniformly at random without considering the labels.

For the sake of simplicity, we do not perform subsampling when building the CompRF trees. We report some experiments concerning the role of subsampling in a separate subsection. All other parameters of CompRF are adjusted by cross-validation.

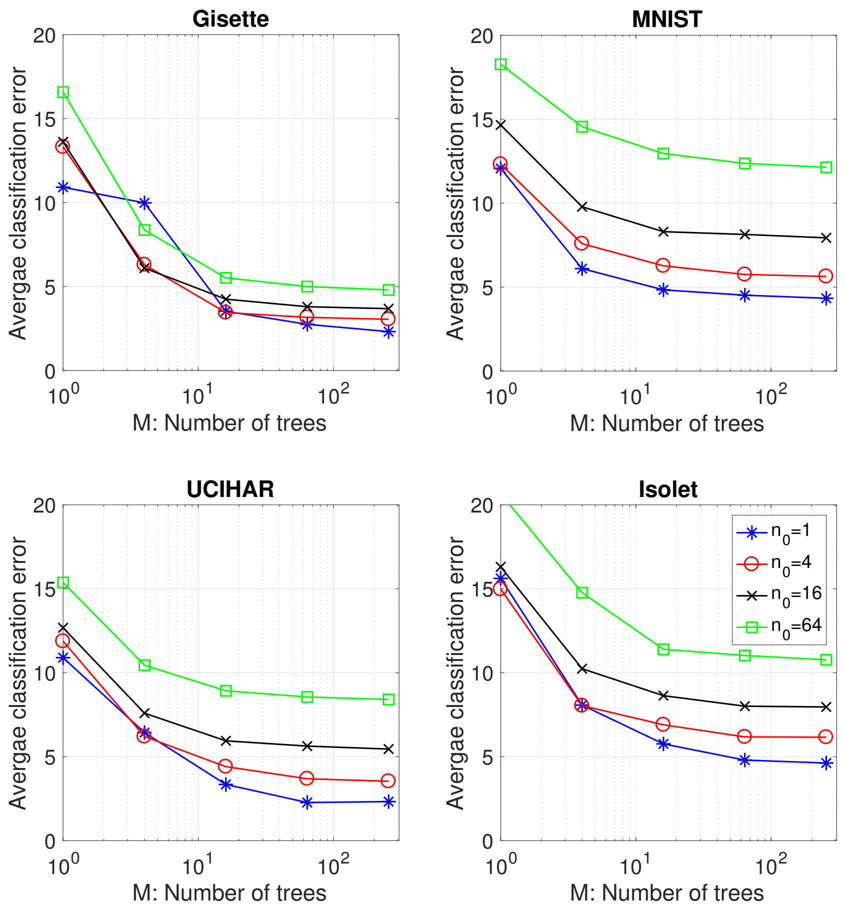


Figure 3.10: Average classification error of the CompRF algorithm on classification datasets. X-Axis shows the number of trees used in the forest. The title denotes the dataset and each curve corresponds to a fixed value of n_0.

Classification

We use four classification datasets. MNIST (Lecun and Cortes, 1998) and Gisette are handwritten digit datasets. Isolet and UCIHAR are speech recognition and human activity recognition datasets respectively (Lichman, 2013). Details of the datasets are shown in the first three rows of Table 3.1.

Parameters of CompRF: We examine the behaviour of the CompRF with respect to the choice of the leaf size n_0 and the number of trees M. We perform 10-fold cross-validation over $n_0 \in \{1, 4, 16, 64\}$ and $M \in \{1, 4, 16, 64, 256\}$. In Figure 3.10 we report the resulting cross validation error. Similar to the recommendation for CART forests (Biau and Scornet, 2016), we achieve the best performance when the leaf size is small, that is $(n_0 = 1)$. Moreover, there is no significant improvement for M greater than 100.

Table 3.1: Average and standard deviation of classification error for the CompRF vs. other methods. The first three rows describe datasets.

	MNIST	Gisette	UCIHAR	Isolet
Dataset Size	70000	7000	10229	6238
Variables	728	5000	561	617
Classes	10	2	5	26
KNN	2.91	3.50	12.15	8.27
CART RF	2.90 ($\pm$ 0.05)	3.04 ($\pm$ 0.26)	7.47 ($\pm$ 0.32)	5.48 ($\pm$ 0.27)
CompRF (U)	4.21 ($\pm$ 0.05)	3.28 ($\pm$ 0.19)	8.70 ($\pm$ 0.32)	6.65 ($\pm$ 0.14)
CompRF	**2.50** ($\pm$ 0.05)	**2.48** ($\pm$ 0.13)	**6.54** ($\pm$ 0.11)	**4.43** ($\pm$ 0.26)

Comparison between CompRF, CART and KNN: Table 3.1 shows the average and standard deviation of classification error for 10 independent runs of CompRF, CART forest and KNN. We also report the performance of the unsupervised CompRF algorithm denoted by CompRF(U) in Table 3.1. Training and test sets are given in the respective datasets. The parameters n_0 and M of CompRF and CART, and k of KNN are chosen by 10-fold cross validation on the training set. Note that KNN is not randomized, thus there is no standard deviation to report.
The results show that, surprisingly, CompRF can slightly outperform the CART forests for classification tasks even though it uses considerably less information. The reason might be that the CompRF splits are better adapted to the geometry of the data than the CART splits. While the CART criterion for selecting the exact splitting point can be very informative for regression (see below), for classification it seems that a simple data dependent splitting criterion as in the supervised CompRF can be as efficient. Conversely, we see that unsupervised splitting as in the unsupervised CompRF is clearly worse than supervised splitting.

Regression

Next we consider regression tasks on four datasets. Online news popularity (ONP) is a dataset of articles with the popularity of the article as target (Fernandes *et al.*, 2015). Boston is a dataset of properties with the estimated value as target variable. ForestFire is a dataset meant to predict the burned area of forest fires, in the northeast region of Portugal (Cortez and Morais, 2007). WhiteWine (Wine) is a subset of wine quality dataset (Cortez *et al.*, 2009). Details of the datasets are shown in the first two rows of Table 3.2.
Since the regression datasets have no separate training and test set, we assign 90% of the items to the training and the remaining 10% to the test set. In order to remove

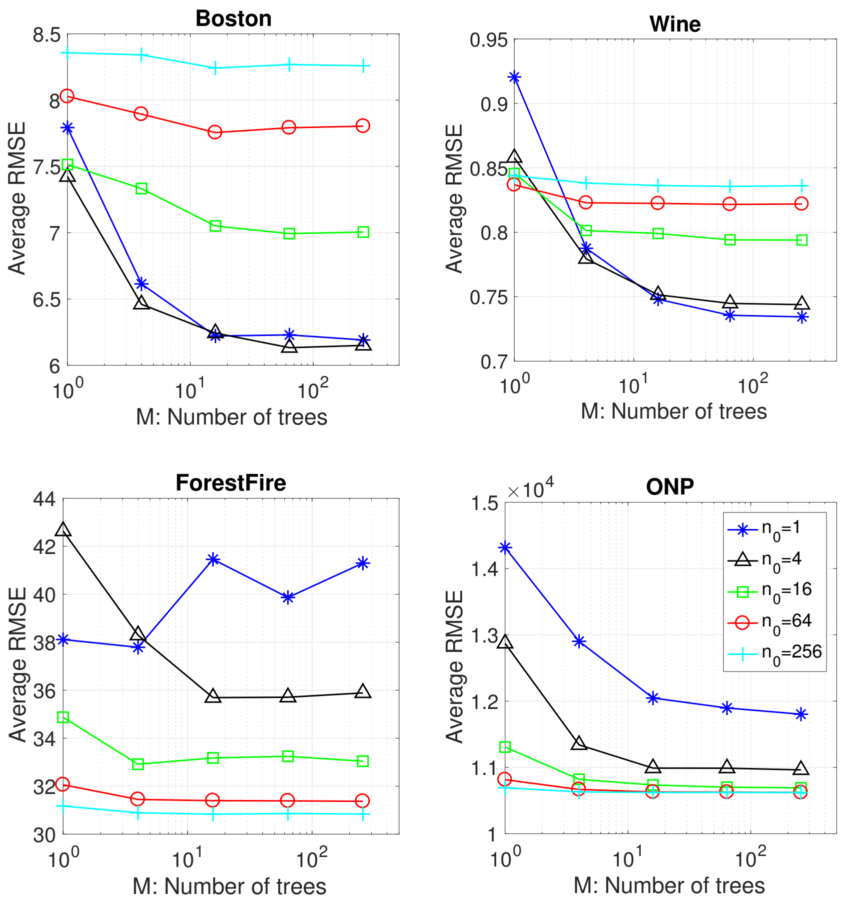


Figure 3.11: Average RMSE of the CompRF algorithm on regression datasets. X-Axis shows the number of trees used in the forest. The title denotes dataset and each curve corresponds to a fixed value of n_0.

the effect of the fixed partitioning, we repeat the experiments 10 times with random training/test set assignments. Note that we use CompRF with unsupervised tree construction for regression.

Parameters of CompRF: We report the behaviour of the CompRF with respect to the parameters n_0 and M. We perform 10-fold cross-validation with the same range of parameters as in the previous subsection. Figure 3.11 shows the average root mean squared error (RMSE) over the 10 folds. The cross-validation is performed for 10 random training/test set assignments. The figure corresponds to the first assignment out of 10 (the behaviour for the other training/test set assignments is similar). The CompRF algorithm shows the best performance with $n_0 = 1$ for the Boston and ForestFire datasets, however larger values of n_0 lead to better performance for other datasets. We believe that the main reason for this variance is the unsupervised tree

Table 3.2: Average and standard deviation of the RMSE for the CompRF vs. CART regression forest.

	ONP	Boston	ForestFire	WhiteWine
Dataset Size	39644	506	517	4898
Variables	58	13	12	11
CART RF	**1.04** ($\pm$ 0.50) $\cdot 10^4$	**3.02** ($\pm$ 0.95)	**45.32** ($\pm$ 4.89)	**59.00** ($\pm$ 2.94)$\cdot 10^{-2}$
CompRF	1.05 ($\pm$ 0.50) $\cdot 10^4$	6.16 ($\pm$ 1.00)	45.37 ($\pm$ 4.69)	72.46 ($\pm$ 3.16) $\cdot 10^{-2}$

construction in the CompRF algorithm for regression.

Comparison between CompRF and CART: Table 3.2 shows the average and standard deviation of the RMSE for the CompRF and CART forests over the 10 runs with random training/test set assignment. For each combination of training and test sets we tuned parameters independently by cross validation. CompRF is constructed with unsupervised splitting, while the CART forests are built using a supervised criterion. We can see that on the Boston and Wine datasets, the performance of the CART forest is substantially better than the CompRF. In this case, ignoring the Euclidean representation of the data and just relying on the comparison-based trees leads to a significant decrease in performance. However the performance of our method on the other two datasets is quite the same as CART forests. We can conclude that in some cases the CART criterion can be essential for regression. However, note that if we are just given a comparison-based setting—without actual vector space representation—it is hardly possible to propose an efficient supervised criterion for splitting.

CompRF and subsampling

Here, we investigate the role of subsampling in the performance of the CompRF. To construct each tree of the CompRF, we randomly pick a subsample of $r|S|$ points among the set of training points (S) without replacement and make the tree only based on the subsample. We use the following range for the subsampling ratio: $r \in \{0.1, 0.2, 0.4, 1\}$. The left panel of Figure 3.12 shows the average classification error of the CompRF for various values of r. The right plot in this figure shows the normalized average MSE of the CompRF for regression datasets. Note that the range of MSE depends on the dataset. To make a unified figure, for each dataset, we divided all average values of the MSE by the maximum value of the MSE on that particular dataset.

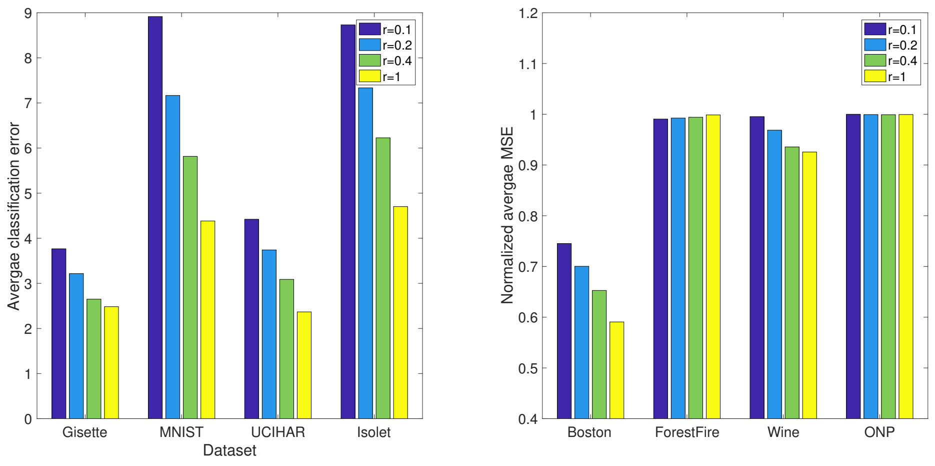


Figure 3.12: (Left) Average classification error of the CompRF algorithm on four classification datasets and various subsampling ratios (r). (Right) Normalized average MSE of the CompRF algorithm on four regression datasets and various subsampling ratios (r). The X-axis denotes the datasets. Note that for each dataset we divided all MSE values by the maximum value of the dataset. In this way bars can be plotted together.

Our results hardly show any significant positive effect of subsampling. On the contrary, in classification tasks we see a significant decrease in error when the whole dataset is used. Only in case of ForestFire dataset do we see some slight improvement.

3.3.2 Metric, non-Euclidean setting

In this set of experiments we aim to demonstrate the performance of the CompRF in general metric spaces. We choose graph-structured data for this experiment. Each data-point is a graph, and as a distance between graphs we use graph-based kernel functions. In particular, the Weisfeiler-Lehman graph kernels are a family of graph kernels that have promising results on various graph datasets (Shervashidze *et al.*, 2011). We compute the WL-subtree and WL-edge kernels on four of the datasets reported in Shervashidze *et al.* (2011): MUTAG, ENZYMES, NCI1 and NCI109. In order to evaluate triplet comparisons based on the graph kernels, we first convert the kernel matrix to a distance matrix in the standard way (expressing the Gram matrix in terms of distances).
We compare supervised and unsupervised CompRF with the Kernel SVM and KNN classifier in Table 3.3. Note that in this setting, CART forests are not applicable as they would require an explicit vector space representation. Parameters of the

Table 3.3: Average and standard deviation of the classification error for the CompRF in comparison with kernelSVM on graph datasets with two graph kernels: WL-subtree and WL-edge.

	MUTAG	ENZYMES	NCI1	NCI109
Train Size	188	600	4110	4127
Classes	2	6	2	2
WL-subtree kernel				
Kernel SVM	17.77 (± 7.31)	47.16 (± 5.72)	**15.96** (± 1.56)	**15.55** (± 1.40)
KNN	14.00 (± 8.78)	48.17 (± 4.48)	18.13 (± 2.27)	18.74 (± 1.97)
CompRF unsupervised	14.44 (± 7.94)	**39.33** (± 6.49)	17.96 (± 1.85)	19.10 (± 2.22)
CompRF supervised	**13.89** (± 7.97)	39.83 (± 5.00)	17.35 (± 1.98)	18.71 (± 2.61)
WL-edge kernel				
Kernel SVM	15.55 (± 6.30)	53.67 (± 6.52)	**15.13** (± 1.44)	**15.38** (± 1.69)
KNN	12.78 (± 7.80)	51.00 (± 4.86)	18.56 (± 1.36)	18.30 (± 1.82)
CompRF unsupervised	11.67 (± 7.15)	38.50 (± 4.19)	17.91 (± 1.42)	19.56 (± 1.61)
CompRF supervised	**11.11** (± 8.28)	**38.17** (± 5.35)	18.05 (± 1.63)	18.40 (± 2.27)

Kernel SVM and k of the KNN classifier are adjusted with 10-fold cross-validation on training sets.
We set the parameters of the CompRF to $n_0 = 1$ and $M = 200$, as it shows acceptable performance in the Euclidean setting. We assign 90% of the items as training and the remaining 10% as the test set. The experiment is repeated 10 times with random training/test assignments. The average and standard deviation of classification error is reported in Table 3.3. The CompRF algorithm outperforms the kernel SVM on the MUTAG and ENZYMES datasets. However, it has slightly lower performance on the other two datasets. However, note that the kernel SVM requires a lot of background knowledge (one has to construct a kernel in the first place, which can be difficult), whereas our CompRF algorithm neither uses the explicit distance values nor requires them to satisfy the axioms of a kernel function.

3.3.3 Comparison-based setting

Now we assume that the distance metric is unknown and inaccessible directly, but we can actively ask for triplet comparisons. In this setting, the major competitors to comparison-based random forests are indirect methods that first use ordinal embedding to a Euclidean space, and then classify the data in the Euclidean space. As practical active ordinal embedding methods do not really exist (with the exception of (Jamieson and Nowak, 2011), but this algorithm is mostly of theoretical interest and unsuitable in practice), we settle for a batch setting in this case. After embedding, we use CART forests and the KNN classifier in the Euclidean space.

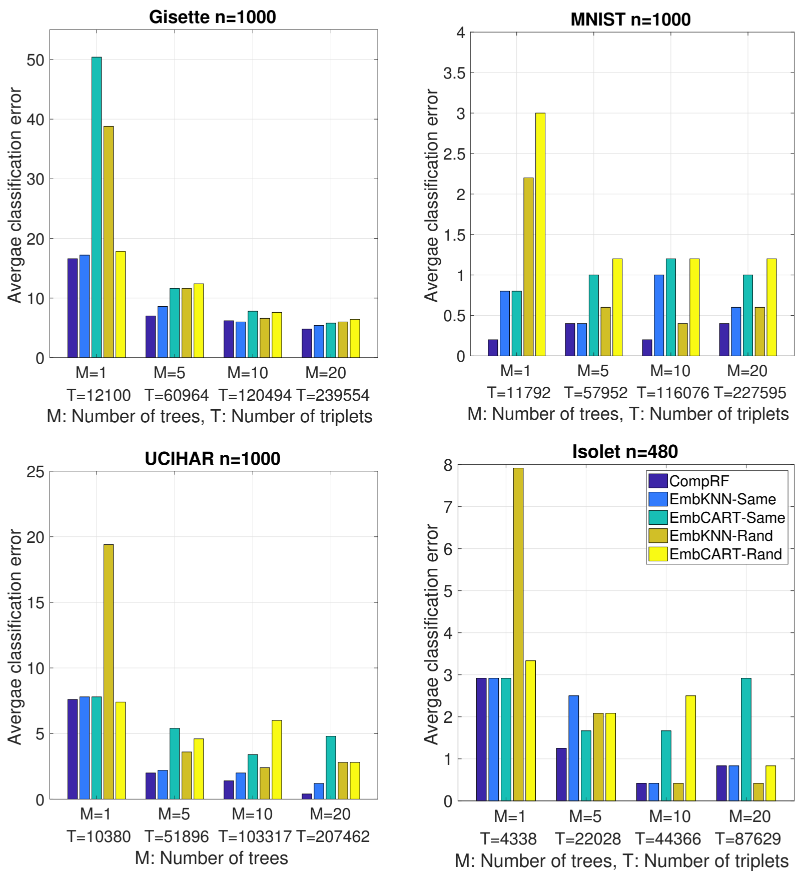


Figure 3.13: Average classification error of the CompRF in comparison with embedding approach on classification datasets with less than 1000 items. *EmbKNN-Same* (resp. *EmbCART-Same*) denotes the TSTE+KNN using the same triplets as CompRF, while *EmbKNN-Rand* (resp. *EmbCART-Rand*) stands for using TSTE with the same number of random triplets. X-Axis show the number of trees (M) used for the CompRF and the corresponding number of triplets (T) for the embedding. Each set of bars corresponds to a fixed M. Note that by increasing M, the number of triplets used by CompRF will be increased, as it appears in the X-Axis.

Comparing various ordinal embedding algorithms, such as GNMDS (Agarwal *et al.*, 2007), LOE (Terada and Luxburg, 2014) and TSTE (van der Maaten and Weinberger, 2012) shows that the TSTE in combination with a classifier consistently outperforms the others. Therefore, we here only report the comparison with the TSTE embedding algorithm. The dimension of embedding is a crucial parameter for embedding methods. Since the running time also grows with dimension it is infeasible to embed

in very high dimensions. Moreover in a general comparison-based setting, we do not know the proper Euclidean dimension that items should lie in. We choose the embedding dimension by 2-fold cross-validation in the range of $d \in \{10, 20, 30, 40, 50\}$ (embedding in more than 50 dimensions is impossible in practice due to the running time of the TSTE). We also adjust k of the KNN classifier in the cross-validation process.
We design a comparison-based scenario based on Euclidean datasets. First, we let CompRF choose the desired triplets to construct the forest and classify the test points. The embedding methods are used in two different scenarios: once with exactly the same triplets as in the CompRF algorithm, and once with a completely random set of triplets of the same size as the one used by CompRF.
The size of our datasets by far exceeds the number of points that embedding algorithms, particularly TSTE, can handle. To reduce the size of the datasets, we choose the first two classes, then we subsample 1000 items. Isolet has already less than 1000 items in first two classes. We assign half of the dataset as training and the other half as test set. Bar plots in Figure 3.13 show the classification error of the CompRF in comparison with embedding methods with various numbers of trees in the forests (M). We set $n_0 = 1$ for the CompRF.
In each set of bars, which corresponds to a restricted comparison-based regime, CompRF outperforms embedding methods or has the same performance. Another significant advantage of CompRF in comparison with the embedding is the low computation cost. A simple demonstration is provided in the following subsection.

Running time of CompRF vs. embedding method

Here we report the running time of CompRF in comparison with TSTE embedding combined with KNN. Note that if we apply CART forest after embedding, it can be even more time consuming. In addition, the running time of embedding does not change significantly if we apply the same triplets as the CompRF or a random subsample of triplets, therefore we report the running time based on the same triplets as the CompRF.
We use the subsample of Gisette dataset with $n = 1000$ point, similar to the previous subsection. We perform the embedding with $d = 10$ and $d = 50$ dimensions and fixed $k = 5$. Table 3.4 shows the running time of the experiments. Since the running time of embedding can change significantly based on the initial conditions, we run embedding algorithms five times and we report the average running time. The algorithms are implemented on a single core CPU and the running times are reported in seconds.
The required running time for the embedding algorithm is orders of magnitude longer than the CompRF. Moreover, the embedding algorithms need a cross-validation step to adjust the number of dimensions and other parameters of the classifier.

Table 3.4: Comparison of computation time between CompRF and TSTE+KNN. The reported values are in seconds.

Number of trees (M)	1	5	10	20
CompRF	1	4	8	16
TSTE+KNN (d=10)	148	236	350	595
TSTE+KNN (d=50)	185	654	1214	2398

3.4 Conclusion and future work

We propose comparison-based random forests for classification and regression tasks. This method only requires comparisons of distances as input. From a practical point of view, it works surprisingly well in all kinds of circumstances (Euclidean spaces, metric spaces, comparison-based setting) and is much simpler and more efficient than some of its competitors such as ordinal embeddings.
We have proven consistency in a simplified setting. As future work, this analysis should be extended to more realistic situations, namely tree construction depending on the sample; forests with inconsistent trees, but the forest is still consistent; and finally the supervised splits. In addition, it would be interesting to propose a comparison-based supervised tree construction for the regression tasks.

Chapter 4

Estimation of perceptual scales using ordinal embedding

4.1 Psychophysical scaling

The quantitative study of human behavior dates back to 1860, when the experimental physicist Gustav Theodor Fechner published *Die Elemente der Psychophysik* (Fechner, 1860). Fechner not only founded the scientific discipline of psychophysics, but his work is widely regarded as the beginning of the quantitative, scientific study of psychology. Since the seminal work of Fechner, the “measurement of sensation magnitude” has been at the very heart of psychophysics (Gescheider, 1988). This problem is called “psychophysical scaling” later in the literature, which is formally defined as the problem of quantifying the magnitude of sensation induced by a physical stimulus (Marks and Gescheider, 2002; Krantz *et al.*, 2007).

Here, we focus on a general psychophysical scaling problem. More precisely, there exists a physical quantity as the stimulus, which we can objectively measure. The perception (or sensation) of the stimulus is usually hard to measure and quantify. The (difference) scaling problem refers to experiments and methods designed to find the functional relation between the perceived magnitude and the stimulus. An example of a psychophysical scaling function is outlined in Figure 4.1. For brevity, throughout the rest of the thesis, we refer to this function as scaling function. In Figure 4.1, the physical stimulus S and its perceived counterpart ψ are denoted on the X and Y axes respectively. The scaling function, also called psychometric function, relates the perceived value and the raw stimulus values as follows:

$$\psi = f(S)$$

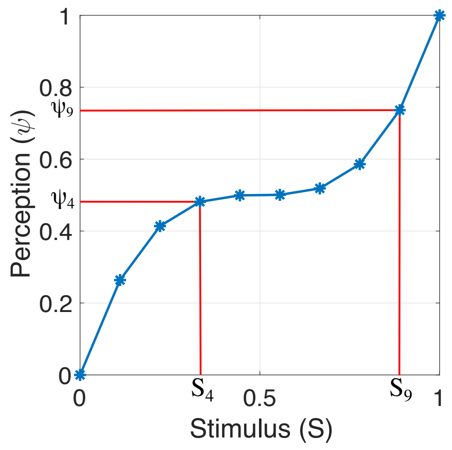


Figure 4.1: An example of psychopysics scaling function. The X-axis shows the physical stimulus values (S) with 10 discrete steps. Y-axis denotes the perceived value (ψ)

The approach of Fechner, which led to the Fechner's law, is based on experiments with measurements of the just-noticeable-difference (JND). One JND is the smallest amount of change in the stimulus level which is noticeable by the human observer. Assuming that each JND corresponds to one unit of the perceptual scale, one can reconstruct the scaling function (Fechner, 1860; Luce and Edwards, 1958). Even though the Fechner's law holds for many cases of sensory stimuli, there exists a great deal of criticism to the method (Norris and Oliver, 1898; Stevens, 1957; Gescheider, 1988).

Thurstonian scaling is another common approach for the scaling problem (Thurstone, 1927). Thurstonian scaling is based on discrimination of stimuli pairs. In simple words, the perceptual distance of two stimuli is determined by the probability that a human observer can discriminate the stimuli pair. The other alternative to scaling is the direct approach of magnitude estimation (Stevens, 1957). In this approach, a human observer is asked to provide perceived intensity values corresponding to physical stimuli in a way that ratios of given values represent the ratios of perception. Shepard pointed out a possible flaw of the direct approach later in Shepard (1981). According to Shepard, there exists an unknown and undesirable *response transformation function*, which in the direct magnitude estimation method is neglected. The intervening response transformation function causes a bias on direct responses. For a detailed overview of the psychophysical scaling methods see Gescheider (1988). In the following, we discuss the method of triads for psychophysical scaling.

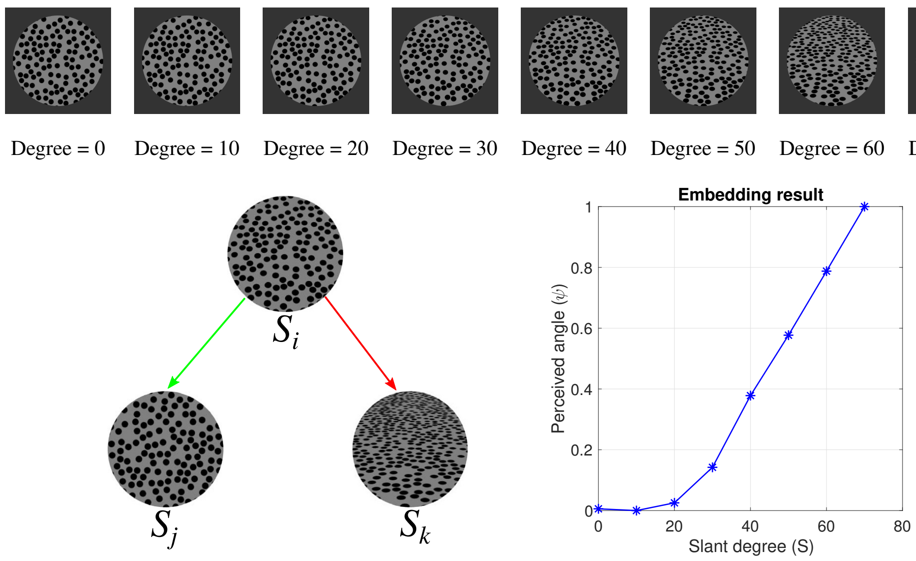


Figure 4.2: Top: Eight stimuli used in the slant-from-texture experiment (Aguilar *et al.*, 2017). Bottom-left: An example of a triplet question used for the slant-from-texture experiment. The triplet question is: "Which of the bottom images –S_j or S_k– is more similar to the top image S_i?" Bottom-right: The scaling function gathered by the comparison-based approach.

4.1.1 Scaling and the method of triads

An alternative approach to data acquisition in vision science—and cognitive science in general—is based on triplet comparisons (Torgerson, 1958). This approach is traditionally referred to as *method of triads*, in the psychophysics literature. There exists a fixed discretization of the physical stimulus, say $S_1, \ldots, S_n$. The method of triads, proposed in Torgerson (1958), asks participants triplet questions in the form of Equation (1.1). Rather than attempting accurate quantitative measurements of a particular phenomenon, triplet questions aim for qualitative observations. The obvious potential of such an approach is that the statements are not dependent on the response transformation function of the observers, thus the issue of scaling answers across many observers becomes easier.

The approach is to collect the answers to a set of triplet questions from the observers and then perform a method to estimate the scaling function from triplet comparisons. Suppose that $\psi_1, \ldots \psi_n$ denote the perceived values of stimuli $S_1, \ldots, S_n$ (see Figure 4.1). The problem boils down to estimating the magnitudes of perception $\hat{\psi}_1, \ldots \hat{\psi}_n$, which are consistent with the answers to the queried triplet questions.

We give an example to clarify the procedure of scaling using the triplet questions. The psychophysics experiment, called "slant-from-texture" experiment, is designed to find the functional relation of perceived angle with the true angle of the tilted flat plane with a dotted texture. (see Aguilar *et al.* (2017) for details). Figure 4.2 (Top) shows the various stimuli used in the experiment. The bottom-left image depicts an example of a triplet question designed for this task. The participant is asked "which of the two bottom images — S_j or S_k — is more similar to the top image S_i?"; see Figure 4.2 (Bottom-left). Having the answers to a set of triplet questions, one can reconstruct the scaling function. The function, shown in Figure 4.2 (Bottom-right), describes the relation of perceived angle ψ and the slant degree S (actual stimulus).

The approach of triplet comparisons—the method of triads—is not new to psychophysics; there has been a very long tradition in psychology to explore methods to estimate perceptual (difference) scales from clearly visible *supra-threshold* differences in stimulus appearance (Torgerson, 1958; Coombs *et al.*, 1970; Marks and Gescheider, 2002). The earlier approaches are based on inferring a similarity matrix using the triplet comparisons. Recently a more generic approach based on the triplet comparisons, called the "maximum likelihood difference scaling (MLDS)", has become popular in vision science for estimation of perceptual (difference) scales (Maloney and Yang, 2003; Knoblauch and Maloney, 2010). There have been reports that both naive, as well as seasoned observers, find the method of triads with supra-threshold stimuli intuitive and fast, requiring less training (Aguilar *et al.*, 2017).

However, the MLDS method comes with major obstacles which severely limit its use in psychophysics: First, it makes a strong model assumption. It assumes that the scaling function is monotonic with respect to the stimulus. Secondly, the MLDS method is capable of finding only one-dimensional scaling functions. Thus, it cannot deal with the cases that the perception is intrinsically multi-dimensional (e.g. color). Both issues are of high relevance in a general psychophysics scaling problem.

On the other hand, the evaluation of comparison-based data (data gathered from triplet comparisons) has been an active field of research in computer sciences and machine learning (Schultz and Joachims, 2003; Agarwal *et al.*, 2007; Tamuz *et al.*, 2011; Ailon, 2011; Jamieson and Nowak, 2011; van der Maaten and Weinberger, 2012; Kleindessner and von Luxburg, 2014; Terada and Luxburg, 2014; Ukkonen *et al.*, 2015; Arias-Castro *et al.*, 2017; Jain *et al.*, 2016; Haghiri *et al.*, 2017). The core question of the studies is to find a Euclidean representation of the items based on the answers to triplet questions. This problem is systematically studied in the machine learning literature under the name of **ordinal embedding**. There are a number of fast and accurate algorithms developed to solve the ordinal embedding problem (Agarwal *et al.*, 2007; van der Maaten and Weinberger, 2012; Terada and Luxburg, 2014). As we will show in this chapter, these algorithms can be extremely useful in the field of psychophysics as well.

In this chapter, we demonstrate and discuss how the new machine learning methods can be applied to the field of psychophysics, and how and when they can be of advantage. The rest of this chapter is organized as follows: In section 4.2, we review two traditional embedding methods of psychophysics which are closely related to the approach of triplet comparisons. In addition, we formally introduce the ordinal embedding problem in machine learning and discuss its advantages in comparison to the traditional embedding methods of psychophysics. Section 4.3 is dedicated to extensive simulation scenarios which compare the performance of ordinal embedding against the applicable competitors in the psychophysics. In section 4.4, we apply the comparison-based approach and ordinal embedding methods on a real psychophysics experiment. In the next section, we provide instructions on how to use the comparison-based approach and the ordinal embedding algorithms in psychophysics experiments. In the last section, we conclude the chapter by discussing the advantages of the ordinal embedding for scaling problem and mentioning the open problems.

4.2 Embedding methods

In this section, we discuss three embedding methods which are either proposed to estimate a scaling function with triplet comparisons, or related to the method of triads in psychophysics. First, we review the traditional non-metric multi-dimensional scaling (NMDS) (Shepard, 1962; Kruskal, 1964b). Even though this method is not based on triplet comparisons, it uses the rank order of dissimilarities which makes it relevant to the method of triads. Secondly, we explain the more recent method of maximum likelihood difference scaling (MLDS) (Maloney and Yang, 2003). This method works with triplet comparisons, however, it can deal with a very restricted set of scaling functions. Finally, we introduce the problem of ordinal embedding in machine-learning. We show that the ordinal embedding can tackle the scaling problem of psychophysics in a broad sense. We also discuss one of the most successful algorithms to solve the ordinal embedding problem.

4.2.1 Non-metric multi-dimensional scaling (NMDS)

The non-metric multi-dimensional scaling (NMDS) by Shepard and Kruskal is one of the well-established methods to analyze dissimilarity data (Shepard, 1962; Kruskal, 1964a,b). It is assumed that a matrix of dissimilarities (not necessarily a metric distance) between pairs of items is given. In the psychophysics studies, this matrix usually comes from human feedback. Shepard posed the problem of estimating a d-dimensional Euclidean representation of items, say $y_1, y_2, \dots y_n \in \mathbb{R}^d$, such that the pairwise distances of estimates are consistent to a monotonic transform of the given dissimilarities. The monotonic transform takes only the rank order of dissimilarities into consideration. As in many psychophysics experiments, the magnitude of dissimilarity (or similarity) cannot be quantitatively measured, considering the rank order of distances is more plausible.

If δ_{ij} denotes the dissimilarity of items i and j, given as the input, and $d_{ij} = \|y_i - y_j\|$ is the Euclidean distance of embedded items y_i and y_j in $\mathbb{R}^d$, then the goodness of a Euclidean representation is measured by a quantity called **stress** (Kruskal, 1964b):

$$\text{stress} = \frac{\sum_{ij}(d_{ij} - f(\delta_{ij}))^2}{\sum_{ij} {d_{ij}}^2}.$$

A smaller value of stress means a better fit of the Euclidean representation. The nominator, which is the squared loss between the input dissimilarities and the Euclidean distances, makes the distances as close as possible to the dissimilarities. The denominator is added to remove the degenerate solution (d_{ij} and $f(\delta_{ij})$ can become infinitesimal together). Finding the Euclidean representation of items which minimizes the stress function is very troublesome, as the function $f(.)$ can be chosen from the set of all monotonic transform functions.

The approach proposed in Kruskal (1964a) finds an estimation of the optimal solution through a two-step optimization procedure. In the first step, a configuration of embedding points $y_1, y_2, \ldots y_n$ is fixed; this means that distance values d_{ij} are also fixed. Then he suggests a greedy algorithm (called isotonic regression) to find the closest monotonic function $f(.)$ that minimizes the stress function. In the second step of optimization, the values of $f(\delta_{ij})$ are fixed and the embedding points $y_1, y_2, \ldots y_n$ are adjusted by the gradient descent algorithm to minimize the stress. The two steps are repeated consecutively until the stress value shows no further improvement or it becomes smaller than a certain threshold.

The NMDS algorithm has been used extensively in psychology (Reed, 1972; Smith and Ellsworth, 1985; Barsalou, 2014), neuroscience (Op de Beeck *et al.*, 2001; Kayaert *et al.*, 2005; Kaneshiro *et al.*, 2015) and broader fields (Liberti *et al.*, 2014; Machado *et al.*, 2015). The non-parametric flavor of the method makes it a general purpose algorithm that is easy to apply. In addition, it can find representations in multi-dimensional spaces. However, there are two major drawbacks. First, the proposed optimization algorithm does not consider all monotonic transforms of the dissimilarity values. In other words, separating the two steps can lead to a local minimum solution of the proposed optimization. Moreover, fixing the function $f(.)$ leaves a non-convex optimization algorithm. This means that again in one step of the algorithm the solution is a local minimum. The second issue with the NMDS is that it requires the full rank order of pairwise distances as input. If we assume that the order of distances comes from triplet questions, then we require in order of $O\left(n^2 \log n\right)$ triplet questions to sort all pairwise distances. This property makes it infeasible in practice, as the number of required triplets grows very fast with the number of items (stimuli).

4.2.2 Maximum likelihood difference scaling (MLDS)

Decades after the introduction of NMDS, the authors of Knoblauch *et al.* (1998) introduced the maximum likelihood difference scaling (MLDS) method to solve a specific instance of difference scaling problem (Krantz *et al.*, 2007). Later the full description and analysis of the MLDS method appeared in Maloney and Yang (2003). The MLDS method asks quadruplet questions which involve four stimulus levels. If we denote the perceptual scale of four stimuli S_i, S_j, S_k, S_l by $\psi_i, \psi_j, \psi_k, \psi_l$, then a quadruplet question asks whether the difference in perception $|\psi_i - \psi_j|$ is smaller or larger than the difference of perception $|\psi_k - \psi_l|$ (see the definition in (1.2)).
There are two main assumptions in the MLDS model. First, it assumes that the perceived value is a scalar (one-dimensional) denoted by ψ. Secondly, the MLDS method assumes the monotonicity of the perceptual scale with respect to the stimulus. More precisely, it assumes that the order of two stimuli in the physical space implies the same order in the perceptual scale: $S_i < S_j \Rightarrow \psi_i < \psi_j$.
In contrast to the NMDS, the MLDS method considers a parametric model on the feedback of subjects. For a quadruplet of stimulus levels S_i, S_j, S_k, S_l, which be briefly write it as $(i, j; k, l)$), a decision random variable is defined as follows:

$$D(i, j; k, l) = |\psi_i - \psi_j| - |\psi_k - \psi_l| + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian noise with standard deviation $\sigma > 0$. The observer would respond the pair (i, j) has a larger difference if $D(i, j; k, l) > 0$. In this case the response to the quadruplet $q = (i, j; k, l)$ is set to $R_q = 1$, otherwise the response is $R_q = 0$. The goal of MLDS is to estimate the perception scale (ψ) which maximizes the likelihood of queried quadruplet questions. Assuming that $R_1, R_2, \ldots, R_m \in \{0, 1\}$ denote independent responses to m quadruplet questions, the likelihood of perception scales will be:

$$\mathcal{L}(R_1, \ldots, R_m | \psi_2, \ldots, \psi_{n-1}, \sigma) = \prod_{q=1}^{m} \Phi(\Delta_q)^{R_q} [1 - \Phi(\Delta_q)]^{1-R_q},$$

where $\Phi(.)$ denotes the cumulative distribution function of $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $\Delta_q = |\psi_i - \psi_j| - |\psi_k - \psi_l|$ for the quadruplet q. Note that we set $\psi_1 = 0, \psi_n = 1$ to remove degenerate solutions. The likelihood is not convex with respect to the perceptual scale values. Thus, the proposed numerical methods can lead to local maxima.
There MDLS method has a couple of advantages from the theoretical and practical point of view. The maximum likelihood estimator is unbiased and has minimum variance among the unbiased estimators. As a practical advantage, it is empirically shown that a small subset of quadruplets is enough for the convergence of the algorithm. Finally, it is shown empirically that the MLDS method produces low variance output embeddings with respect to the input noise level (Maloney and Yang, 2003).

In spite of the above-mentioned benefits, the MLDS method has a couple of drawbacks. First of all, the algorithm is proposed solely for one-dimensional perception scales. In some cases (see the examples of color and pitch perception in Figure 4.3) the perception scale needs more than one dimension. Secondly, even in the one-dimensional case, the monotonic function assumption is very restrictive and may not hold for many psychophysics experiments. Finally, the unbiased and minimum variance properties of the MLDS holds for the global maximum. However, there is no guarantee that the optimization method reaches the global solution.

4.2.3 Ordinal embedding

Instead of the stimulus levels, in machine learning we deal with a set of abstract items, say $x_1, x_2, \dots, x_n \in \mathcal{X}$. We assume that no representation is available for the items, instead, it is assumed that a metric dissimilarity function $\delta(.,.)$ exists to describe the dissimilarity of the items. The metric δ is not accessible, yet we have access to an oracle which responds to a triplet question $t = (i, j, k)$, based on the metric. The triplet question will be "Is item x_i more similar to item x_j or item x_k"? We denote this triplet question by $t = (i, j, k)$. The response to the triplet is denoted by R_t and stored as the following:

$$R_t = \begin{cases} 1, & \text{if the oracle responds } x_j \text{ is more similar to } x_i \\ -1, & \text{if the oracle responds } x_k \text{ is more similar to } x_i \end{cases} \tag{4.1}$$

Assume that the answers to a subset of triplet questions $T \subset \{(i, j, k) | x_i, x_j, x_k \in \mathcal{X}\}$ are collected from the oracle. Given an embedding dimension d and the answers to the triplet questions T, the ordinal embedding task is to find points $y_1, y_2, \dots y_n \in \mathbb{R}^d$ in a d-dimensional Euclidean space which is consistent with the answers of the queried triplet questions. The consistency of an embedding with a triplet $t = (i, j, k)$ can be determined as the following:

$$R_t \cdot \text{sgn}(\|y_i - y_j\|^2 - \|y_i - y_k\|^2) = \begin{cases} 1, & \text{The embedding is consistent with } R_t \\ -1, & \text{The embedding is not consistent with } R_t \end{cases}$$

The function sgn(.) returns the sign of a real value and $\|y_i - y_j\|$ denotes the Euclidean distance between the points y_i and y_j. The **ordinal embedding** attempts to find an embedding which maximizes the number of consistent triplets. It can be formally written as the following:

$$\max_{y_1,\ldots,y_n \in \mathbb{R}^d} \sum_{t=(i,j,k)\in T} R_t \cdot \operatorname{sgn}(\|y_i - y_j\|^2 - \|y_i - y_k\|^2), \tag{4.2}$$

It is not always possible to find a d-dimensional embedding for an arbitrary dissimilarity metric $\delta(.,.)$. Moreover, in a practical setting the answers to the triplets might be noisy. Therefore, the optimal solution is not necessarily consistent with the full set of triplets T. An extension to this problem, is the case that the embedding dimension (d) is also unknown.

Connection with the scaling problem

Ordinal embedding problem can actually solve the scaling problem of psychophysics in a broad setting. The different stimuli play the same role as the abstract items in the ordinal embedding problem. Indeed, the ordinal embedding does not care about the physical parameters of the stimulus. In other words, all various stimuli in the experiment are treated the same. The advantage is that all sorts of stimulus, possibly with different modalities, can be used with this approach. In addition, ordinal embedding methods can find a multi-dimensional embedding that describes the perceptual space of humans.
The slant-from-texture problem (discussed earlier, see Figure 4.2) can be approached by one-dimensional ordinal embedding. Since the perceived angle is a scalar, the ordinal embedding with dimension $d = 1$ can find the proper solution to the scaling problem. In this example we apply one of the ordinal embedding algorithms with the answers to queried triplet questions. The output embedding is a one-dimensional scalar. We plot the functional relation of stimulus and the perceptual scale in Figure 4.2 (Bottom-right). Note that the perception is only measured through the triplet questions from the participant and no similarity or distance value is asked. More details on this experiment is provided in Section 4.4.
Before moving to the particular methods for ordinal embedding we briefly discuss two examples in the psychophysics to demonstrate the benefits of the multi-dimensional ordinal embedding. The color perception is usually described by more than one perceptual dimension. Figure 4.3 (Left) shows the two-dimensional color circle proposed in Shepard (1962); Ekman (1954). The color circle is reconstructed with the NMDS method based on the original 14*14 similarity judgment matrix. The wavelength is written at the right side of each colored dot. Surprisingly, the violet dots are similar to the red dots for a human observer. This fact suggests a circular perceptual curve which is only possible in two dimensions. The second example is the the pitch perception, which is thought to be perceived on a two-dimensional helix, even though the frequency is one-dimensional (Shepard, 1982; Houtsma, 1995).

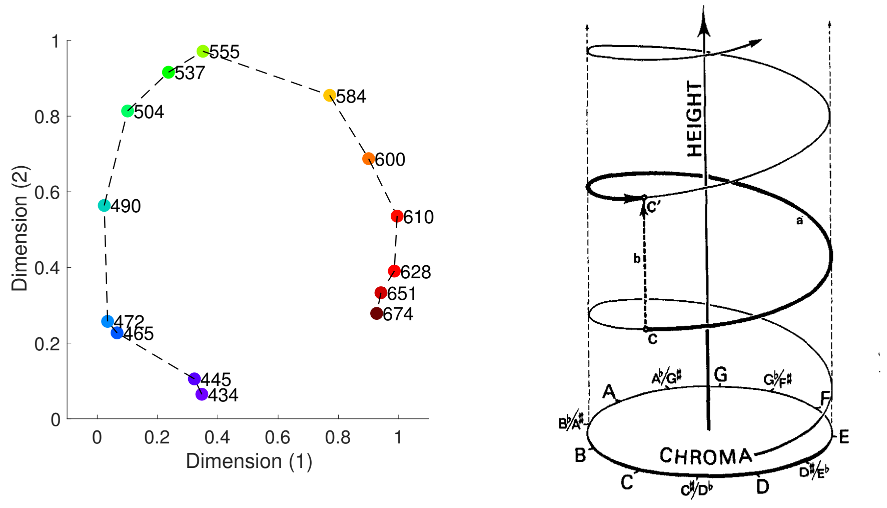


Figure 4.3: Left : The two-dimensional circle of color perception gathered by similarity measurements between 14 colors (Shepard, 1962). The *wavelength* of each color is written on the right side of the colored dot. Right: The helix proposed by Shepard for the pitch perception. The *chroma* of the pitch varies along the spiral path of the curve and the two-dimensional space describes the perception (Shepard, 1982).

Figure 4.3 (Right) shows the proposed perception space by Shepard. In both cases, pitch and color, the multi-dimensional ordinal embedding can enable the researcher to find perceived values in a two-dimensional (or higher) Euclidean spaces that properly describes the perception.

Stochastic triplet embedding

In recent years, there has been a surge of methods to address the ordinal embedding problem in the machine learning community, for example generalized non-metric multidimensional scaling (GNMDS) (Agarwal *et al.*, 2007), crowd-median kernel (Tamuz *et al.*, 2011), stochastic triplet embedding (STE) (van der Maaten and Weinberger, 2012) and local ordinal embedding(LOE) (Terada and Luxburg, 2014). In general, the focus of the machine learning community is to build methods that require only a small number of triplets to embed a large number of items, make as few assumptions as possible, and to be robust towards noise in the data.
In the following we focus on the stochastic triplet embedding (STE) and its variation, t-distributed stochastic triplet embedding (t-STE), because in our experience they work the best and also have a simple model that might also be plausible in a psychophysics setting. The STE method introduces a probabilistic model to solve the ordinal embedding problem defined in equation 4.2. Assume that $y_1, \dots, y_n \in \mathbb{R}^d$

are the correct representations of our objects. The model assumes if a participant is being asked if y_i is closer to y_j than to y_k, then he gives a positive answer according to the following probability:

$$p_{ijk} = \frac{\exp(-\|y_i - y_j\|^2)}{\exp(-\|y_i - y_j\|^2) + \exp(-\|y_i - y_k\|^2)}.$$

Intuitively, easy triplet questions (where $\|y_i - y_j\|$ is relatively smaller than $\|y_i - y_k\|$) will be answered correctly in most of the cases, whereas difficult triplet questions (where $\|y_i - y_j\|$ is about as large as $\|y_i - y_k\|$) can be mixed up. Given the answers to a set of triplets, the STE algorithm attempts to maximize the likelihood of the embedding (point configuration) with respect to the answered triplets. The answer to a triplet $t = (i, j, k)$ is given according to Equation 4.1. Assuming that triplets are answered independently, one can form the likelihood of an embedding given the answers to a set of triplets T, as the following:

$$\mathcal{L}\left(y_1, \ldots, y_n | R_1, \ldots, R_{|T|}\right) = \prod_{t=(i,j,k)\in T, R_t=1} p_{ijk} \cdot \prod_{t=(i,j,k)\in T, R_t=-1} (1 - p_{ijk}).$$

The log-likelihood is maximized to find the solution of ordinal embedding. In the above formulation, the probability of satisfying a triplet goes rapidly to zero even if a triplet is slightly violated. As a result a severe and a slight violation of triplets are penalized almost the same. To make the statistic more robust, the authors proposed using a t-Student kernel with a heavy tail instead of the Gaussian kernel (van der Maaten and Weinberger, 2012). Note that the kernel function measures similarities of estimated outputs. The modified method is called t-distributed STE (t-STE).
The STE and t-STE algorithms can deal with a large number of items (stimulus levels) and reasonable number of triplets. Moreover, the heavy-tail distribution of t-STE makes it robust to noise, which is an important characteristic dealing with psychophysics data. Unlike MLDS, the algorithm is capable of embedding in higher dimensional Euclidean spaces. Finally, the functional relation is not restricted to monotonic functions, which is the case for the MLDS method. In spite of all advantages, the proposed optimization is not convex, which makes it vulnerable to inappropriate local optima.

4.2.4 Summary of embedding methods

Here we sum up the properties of ordinal embedding and compare it with the related work in psychophysics, namely NMDS and MLDS. Table 4.1 summarizes the methods based on various criteria. Ordinal embedding methods can produce high quality results with a small partial set of triplet answers. This property makes them

superior to the traditional NMDS that requires the full order of distances. On the other hand, the embedding methods are not limited to the case of one-dimensional monotonic functions (as it is assumed in MLDS).
As the number of items, and consequently the number of triplets, grows, ordinal embedding algorithms become drastically slow. This is however, more a concern for machine learning purposes that deal with thousands of items and hundreds of thousands of triplets. The algorithms (particularly STE and t-STE) have a quite acceptable running time for the psychophysics experiments.

4.3 Simulations

In this section, we aim to compare the empirical performance of ordinal embedding methods against the traditional embedding approaches in psychophysics (NMDS and MLDS) with diverse simulations. We consider one-dimensional and two-dimensional perceptual spaces. First, we explain the setup of the simulations for both one-dimensional and two-dimensional scenarios in detail. Then, we present the simulation results of the one-dimensional and two-dimensional perceptual spaces in separate subsections.

4.3.1 Simulation setup

Stimulus and perceptual scale: We assume that the stimulus and perception are measured on a scale from 0 to 1, and the true relation between the physical stimulus and the perception is encoded by a function $f : (0,1) \to (0,1)^d$. The parameter d denotes the dimension of the perceptual spaces. We consider two cases: $d = 1$, and $d = 2$. We consider n uniformly chosen steps for the stimulus levels, denoted by $S = \{S_1, S_2, \dots, S_n\}$. These steps are used to evaluate the perception of the observer through triplet questions. The true perceptual scale for the stimulus S_i is expressed by $y_i = f(S_i)$.
Generating subsets of triplet questions: The assumptions of the MLDS affects the set of valid triplets for this method. The MLDS method assumes that the percep-

Method	Data required	Statistical model	Multi-dimensional
NMDS	Order of distances	No	Yes
MLDS	Partial quadruplet set	Yes	No
t-STE	Partial triplet set	Yes	Yes

Table 4.1: The comparison of ordinal embedding methods. Each row corresponds to one method, while the properties are listed in the columns.

tual function is monotonic. In this way, if we sample three stimuli S_i, S_j, S_k such that $i < j < k$, then the only non-obvious triplet question for the MLDS method is $t = (j, i, k)$. This triplet asks whether the distance $\delta(i, j)$ is smaller or larger than the distance $\delta(j, k)$. By the assumption of MLDS the distance $\delta(i, k)$ is the largest distance among the three distances. Thus the other two triplet questions with the three items are already known, and cannot be fed into the algorithm. However, ordinal embedding methods do not have the monotonic assumption, thus they can deal with three triplet questions based on the three items. In this way, by any combination of three stimuli we can make **one** non-obvious (valid) triplet question for the MLDS method and **three** non-obvious (valid) questions for the ordinal embedding methods. Having n stimuli steps, leads to $\binom{n}{3}$ valid triplets for the MLDS method and $3\binom{n}{3}$ valid triplets for the ordinal embedding methods.
In order to have a fair comparison, we always feed the same number of triplets to all embedding algorithms. A random subset of triplets are chosen without replacement from the set of all valid triplets for each algorithm. The size of the random subset is chosen in the range $r \cdot \binom{n}{3} | r \in \{0.2, 0.4, 0.6, 0.8, 1\}$. The value $r = 1$ is equivalent to the whole set of valid triplets for the MLDS method.
Underlying model to generate triplet answers: In order to generate answers to the triplet questions, we construct a model that resembles a typical observer of the psychophysics experiment. We assume that the simulated observer answers a triplet question based on a noisy version of the perceptual scale, denoted by $\tilde{y}_i = f(S_i) + \varepsilon$. In this notation $\varepsilon \sim \mathcal{N}(0, \Sigma)$ is a zero-mean Gaussian noise in d-dimensional space. The parameter $\Sigma = \sigma \cdot \mathbf{I}_d$ is a diagonal matrix with entries equal to σ on the diagonal. The simulated observer produces the answer to the queried triplet question $t = (i, j, k)$ as follows:

$$R_t = \begin{cases} 1, & \text{if } \|\tilde{y}_i - \tilde{y}_j\| < \|\tilde{y}_i - \tilde{y}_k\|, \\ -1, & \text{Otherwise.} \end{cases}$$

Here, $\|.\|$ denotes the Euclidean distance, which in one-dimensional space equals to the absolute difference. The above formulation means that the model samples three noisy versions of the perceptual function to answer each triplet question.
We use the range $\sigma \in \{0.01, 0.05, 0.1, 0.5\}$ for various noise regimes in our simulations. Note that the perceptual scale (y) plays the same role as the perceptual scale (ψ) in the psychophysics notation. We use a different notation to emphasize the fact that perceptual space can be multi-dimensional, and to make a clear separation to scalar values of ψ.
The above-mentioned model produces answers to the triplet questions. The simulated answers to the triplets cannot be used to run NMDS though. In order to run NMDS one needs measured dissimilarities (exact values of δ_{ij}) between the pairs of items. To have a comparison with the NMDS method we propose an alternative

model to generate the dissimilarity values. We generate one set of noisy perceptual values for n stimuli levels as before, $\tilde{y}_i = f(S_i) + \varepsilon$. We then generate the dissimilarity values $\delta_{i,j} = \|\tilde{y}_i - \tilde{y}_j\|$ to feed into the NMDS algorithm. The information is equivalent to having the answers to all triplet questions. Therefore, we compare the NMDS with the embedding methods when $r = 1$.
Embedding methods: Given the answers to the set of queried triplets, we then apply various embedding algorithms to generate embeddings of the simulated data. The MATLAB implementation by van der Maaten and Weinberger (2012) is used for the STE and t-STE methods [1]. We use the default optimization parameters of both methods. The degree of freedom for the t-Student kernel is set to $\alpha = 1$ for the t-STE method. We use the R-package, available on CRAN repository, for the LOE and MLDS algorithms[2]. Again, we run both algorithms with default optimization parameters. We set the embedding dimension according to the dimension of true perceptual function $d = 1$.
All embedding methods solve a non-convex optimization problem and thus are prone to find inaccurate local optima. To reduce this effect, we run all the algorithms 10 times with random initiations. Among the 10 embedding outputs we choose the one that has the least triplet error (defined below). Note that, this procedure is valid since we only use the set of input triplets and the true perceptual function $f(.)$ is not involved.
Independent of the above repetition (which is done to remove the effect of local minima), each embedding method is executed 10 times. This repetition is meant to analyze the average behaviour and the variances of the algorithms. We finally report the average performance of the methods over the 10 repetitions. The standard deviations are reported in the supplementary material, Section A.1.
Evaluating the results: We consider two approaches to evaluate the performance of various methods:

1. Mean-squaed-error (MSE): For one-dimensional perceptual spaces we can compute the Mean-Squared-Error (MSE) between the estimated scales ($\hat{y}$) and the true function values (y). Since the embedding result is unique only up to scaling, rotation and translation, we need two steps of **normalization** before computing the MSE. First, we re-scale the output of embedding to be in the range of $(0, 1)$ as our functions are defined in this range. This will solve translation and scaling issues. Secondly, If we get the output $\hat{y}$ as a result of embedding, this answer is not unique due to the rotation possibility. More precisely, $-\hat{y}$ can also be considered as an answer without violating any answered triplet of the input set. Therefore, we choose between $\hat{y}$ and $-\hat{y}$, the

[1] `https://lvdmaaten.github.io/ste/Stochastic_Triplet_Embedding.html`
[2] `https://cran.r-project.org/web/packages/loe`, `https://cran.r-project.org/package=MLDS`

output which shows a smaller MSE. In this way we consider the best rotation of the output.

2. Triplet error: The MSE criterion can be only used for one-dimensional embedding outputs. In more than one dimension there will be infinite ways to rotate the embedding output without violating the input triplets. Each of the rotations is a valid answer and should be considered to compute the MSE, which in practice is impossible. As an alternative, we measure the performance of the estimated embedding to evaluate new triplet questions. To this end, we compute a quantity called the **triplet error** for each embedding output. Let assume the embedding algorithm produces the set of embedding $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ given the set of input triplets T. The triplet error is defined based on the a new set of triplets T' as the following:

$$\text{triplet error} = \frac{1}{|T'|} \sum_{t=(i,j,k)\in T'} \mathbf{1}\left[R_t \cdot \operatorname{sgn}(\|\hat{y}_i - \hat{y}_j\|^2 > \|\hat{y}_i - \hat{y}_k\|^2) = -1\right]. \tag{4.3}$$

where $\mathbf{1}(.)$ equals to one, if the inside expression is true and it is zero otherwise. Note that $R_t \cdot \operatorname{sgn}(\|\hat{y}_i - \hat{y}_j\|^2 > \|\hat{y}_i - \hat{y}_k\|^2) = -1$ if the estimated embedding is not consistent with new triplet t. Thus, we count the ratio of non-consistent triplets in this way. In practice we are provided with only *one* set of answered triplets, which we denote by T. We suggest two ways to make the new triplet set T', which we call the **validation set**. First we can assign $T' = T$, meaning that we feed the same set of input triplets to measure the triplet error. In a second way, we perform k-fold cross-validation to avoid overfitting. We partition the set of input triplets (T) into k non-intersecting folds. We perform the embedding and the evaluation k times. In each iteration we pick one of the folds as the validation set (T') and the rest of the folds as the training set (the input to the embedding algorithm). The final triplet error is the average triplet error over k validation sets. Throughout the rest of the thesis, we refer to the latter approach as **cross-validated triplet error**, while the first approach is called the **triplet error**.

4.3.2 One-dimensional perceptual space

We start with one-dimensional scenarios where both the stimulus and the perception scale are one-dimensional, $d = 1$. We present the simulations results in two parts concerning the monotonic and non-monotonic scales.

Simulations with monotonic scales

Our first simulation involves a typical monotonic function as it occurs in many psychophysics experiments. The true perceptual function $f(.)$ (a Sigmoid function) is shown in Figure 4.5 (a). Figure 4.4 (b) and (c) show the output embedding of the MLDS and STE algorithms for 10 iterations respectively. The other ordinal embedding methods, which have a similar performance, are reported in supplementary material (Section A.1). The average (over 10 repetitions) MSE and triplet error of various embedding algorithm are depicted in Figure 4.5 (d) and (e) respectively.
Considering the embedding output, the MSE and the triplet errors, the MLDS method performs better than ordinal embedding algorithms. The main reason is the assumptions that MLDS makes. As MLDS seeks a monotonic function, it has much more chance to find a function shape close to the original function. The ordinal embedding algorithms also show an acceptable performance. In particular, when we provide more triplet answers ($r = 1$) the average MSE and triplet error of MLDS and ordinal embedding algorithm tends to be the same.
A detailed result of this simulation including the four ordinal embedding outputs and the performance of algorithm with other values of σ is appeared in supplementary material; see Figure A.1. We also examine another monotonic function in Figure A.2 of the supplementary material, however the results are consistent with the first function.

Simulations with non-monotonic scales

We perform the same experiment on a non-monotonic function as well. A second-degree polynomial function is chosen as the true perceptual function $f(.)$; see Figure 4.5 (a). Figure 4.5 (b) and (c) show the output embedding of MLDS and STE algorithms for 10 iterations respectively. The embedding output of LOE and t-STE are quite similar to the STE (see supplementary material). The average (over 10 repetitions) MSE and triplet error of various embedding algorithm are depicted in Figure 4.5 (d) and (e) respectively.
The function shapes depicted in Figure 4.5 (b) show the poor performance of the MLDS method for the non-monotonic function. The embedding result is indeed not surprising as MLDS tries to fit the most consistent monotonic function with the given triplet answers. The average MSE and triplet error are also significantly larger for MLDS. Note that in this regime, the ordinal embedding algorithms can correctly estimate the true function shapes. Notably, the ordinal embedding methods are capable of discovering broader scaling functions (non-monotonic scales) with the same number of triplets as we used for the monotonic scales.
Similar to the monotonic functions we report the full details of the simulation in supplementary material; see Figure A.4. We also perform the simulation on a Sinu-

soid function. The results are demonstrated in the Figure A.3 of the supplementary material.

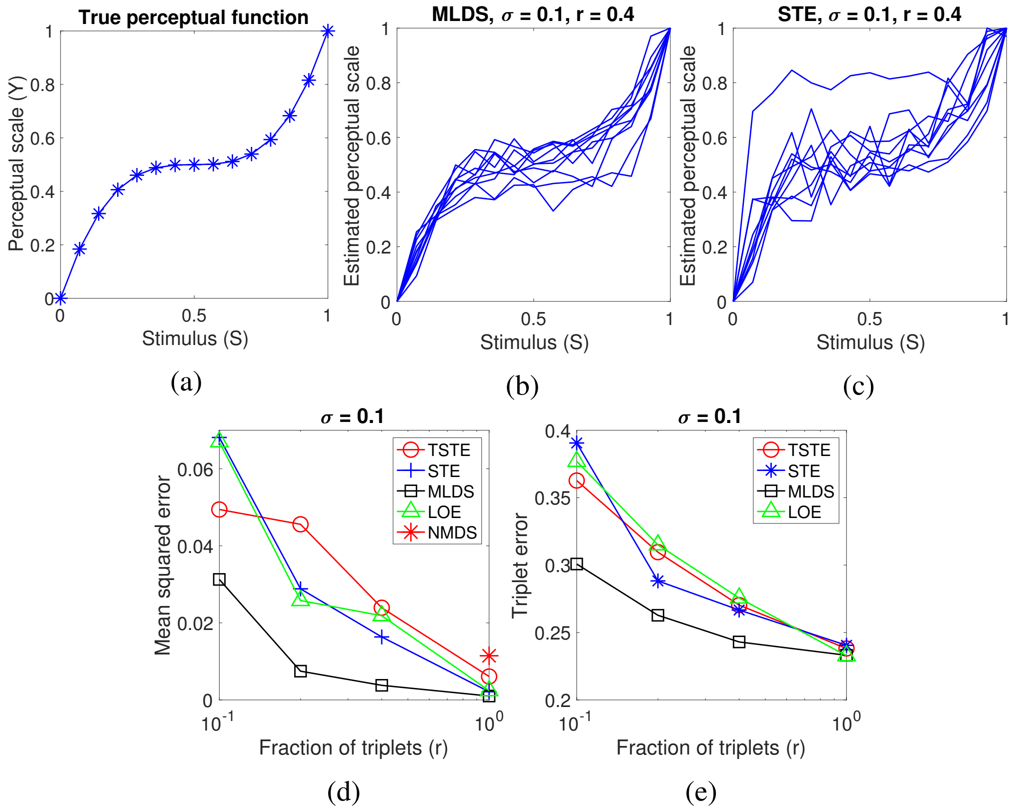


Figure 4.4: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional embedding methods in psychophysics (MLDS and NMDS) for a monotonic one-dimensional perceptual function (Sigmoid). (a) The true perceptual function (y). (b) Ten embedding results ($\hat{y}$) of the MLDS method for a fixed value of standard deviation (σ) and triplet fraction (r). (c) Ten embedding results ($\hat{y}$) of the STE method for a fixed value of standard deviation (σ) and triplet fraction (r). (d) The average MSE of embedding methods. (e) The average triplet error of embedding methods.

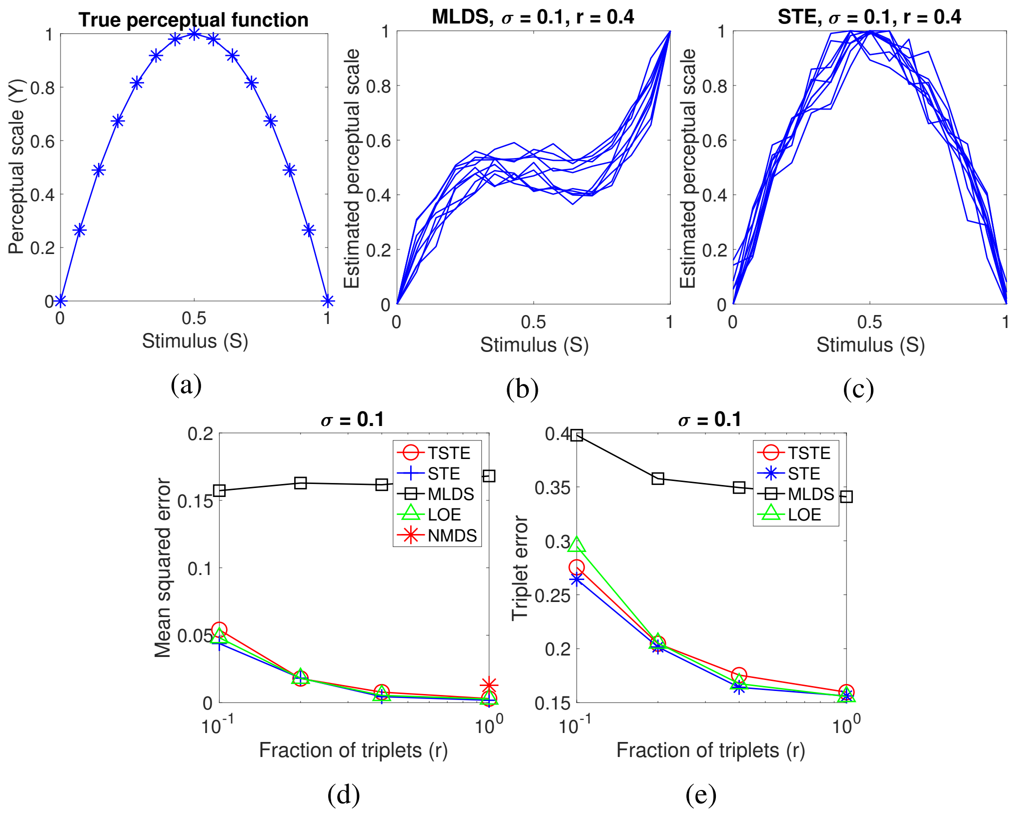


Figure 4.5: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional embedding methods in psychophysics (MLDS and NMDS) for a non-monotonic one-dimensional perceptual function (second degree polynomial). (a) The true perceptual function (y). (b) Ten embedding results ($\hat{y}$) of the MLDS method for a fixed value of standard deviation (σ) and triplet fraction (r). (c) Ten embedding results ($\hat{y}$) of the STE method for a fixed value of standard deviation (σ) and triplet fraction (r). (d) The average MSE of embedding methods. (e) The average triplet error of embedding methods.

4.3.3 Multi-dimensional perceptual space

So far, we considered simulations in which the perception could be represented in a one-dimensional Euclidean space. However, in some cases such as the examples of color and pitch perception in Figure 4.3, more than one dimension is required to represent the perception. It seems quite possible that a physical stimulus is perceived in higher dimensions than the psychical quantity. Here, we perform a simulation with a function mapping from one-dimensional stimulus space into two-dimensional perception space.

As a classic question in visual perception, we choose the color perception. We use the color similarity data presented in Ekman (1954) and the NMDS embedding as a ground truth for our simulations. Figure 4.6 (a) shows the embedding of the similarity data [3] in two dimensions by the NMDS method. The wavelength of each color is also denoted beside the color. The X-Y axes of the plot correspond to the two perceptual dimensions of the color. Note that the NMDS embedding is used as a ground truth in our simulations. In other words, we fix the NMDS embedding, depicted in Figure 4.6 (a), then we generate noisy answers to the triplet questions based on that.

Note that the stimulus is color, represented by the wavelength. We use the wavelength values in our illustration, however, one can rescale the wavelengths to the range of $S \in (0,1)$ to be consistent with the underlying model we defined earlier. In this setting, the true perceptual function $y_i = f(S_i)$, the noisy perceptual scale $\tilde{y}_i$ and the estimated perceptual scale $\hat{y}_i$ are all two-dimensional vectors.

We fix the embedding dimension to $d = 2$ for the following embedding methods: NMDS, LOE, STE and t-STE. However, the MLDS is only capable of embedding in one dimension. Thus, we perform MLDS with $d = 1$. Figure 4.6 (b) and (c) show the two-dimensional embedding output of the NMDS and STE algorithms respectively. The embedding outputs are shown for the parameter values $\sigma = 0.1$ and $r = 1$. The average triplet error of various embedding methods is shown in Figure 4.6 (d) for the parameter value $\sigma = 0.1$.

The comparison of Figure 4.6 (b) and (c) reveals the performance of NMDS and ordinal embedding methods in presence of noise. The STE produces a circle of colors fairly similar to the true perceptual function, while the colors are mixed up in the embedding obtained with the NMDS. The triplet error also shows that ordinal embedding algorithms outperform the NMDS method significantly. It is also worth to mention that MLDS produces large triplet error as it finds an embedding in one dimension. Detailed results of the experiment has appeared in the supplementary material; see Figure A.5.

[3]Data is taken from `https://faculty.sites.uci.edu/mdlee/similarity-data/`

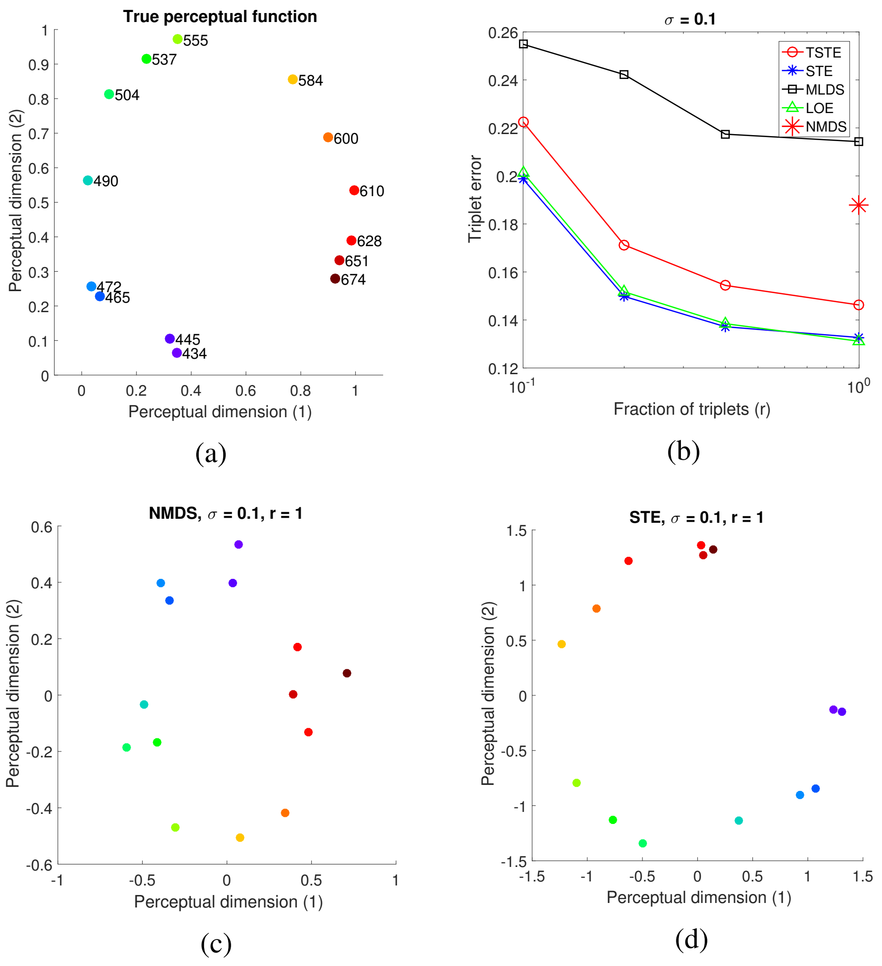


Figure 4.6: The comparison of ordinal embedding methods (MLDS, STE and TSTE) against the traditional NMDS method of psychophysics for the two-dimensional color perception function. (a) The perceptual function in two dimensions. The stimulus value, color wavelength, is written in front of each colored dot. The two-dimensional vector space represents the perceptual space. (b) The average triplet error of various ordinal embedding methods in comparison with the NMDS method. (c) The embedding result of the NMDS method depicted in two dimensions for a fixed value of standard deviation (σ) and triplet fraction (r). (d) The embedding result of the STE method depicted in two dimensions for a fixed value of standard deviation (σ) and triplet fraction (r).

4.4 Experiments

In addition to the extensive simulations, we apply the comparison-based approach and ordinal embedding methods on a real world experiment of the psychophysics. In this experiment we examine the perception of angle for a tilted plane with a dotted pattern on the surface. The experiment is called “slant-from-Texture”.

Slant-from-texture experiment

The experiment is intended to find the functional relation between the perceived angle of the slant with a dotted surface and the actual physical degree of slant. The dataset is originally used in Aguilar *et al.* (2017) (see the main paper for more information on the experiment settings). Figure 4.2 (Top) shows the eight stimuli used in this experiment. The degree of slant is varied from 0 to 70 degrees in steps of 10 degrees, making 8 stimulus levels. The experiments is initially performed with the assumption of a monotonic relation of slant degree and the perception. In this way, for each combination of three stimuli $S_i < S_j < S_k$(three degrees of tilting) only one triplet question is asked, whether $\delta(S_j,S_i) < \delta(S_j,S_k)$. There are $\binom{8}{3} = 56$ possible triplet questions considering 8 levels of the stimulus. Each subject has answered the whole set of possible triplets multiple times to remove the effect of noisy response as much as possible. Subjects $\{1,6,8\}$ have answered 420 triplet question in total, while the other subjects answered 840.

Since the ground truth embedding is unknown, we can only rely on the triplet error for evaluation of the output embedding. To avoid overfitting we use 10-fold cross-validation to compute the *cross-validated triplet error* (See the definition in Simulation setup). Figure 4.7 (Top) shows the average and standard deviation of the cross-validated triplet error for 8 subjects and four embedding methods, including: MLDS, STE, t-STE and LOE. All ordinal embedding algorithm have very similar performance to the MLDS. We can notice that t-STE slightly works better than other algorithm. Thus, we suggest this method as the main candidate for psychophysics experiments.

In addition to the triplet error, we also show the embedding outputs of MLDS and t-STE for 8 subjects in Figure 4.7 (Bottom). Note that the plots are gathered with the full set of triplets, not the only the training folds. The resulting functions are very similar, unless that the t-STE embeddings tend to be non-monotonic in some cases. Even-though the t-STE predicts non-monotonic outputs it still has the same triplet error as the MLDS. This shows that for a couple of subjects a non-monotonic function, which is totally ignored by the MLDS, can be a better fit.

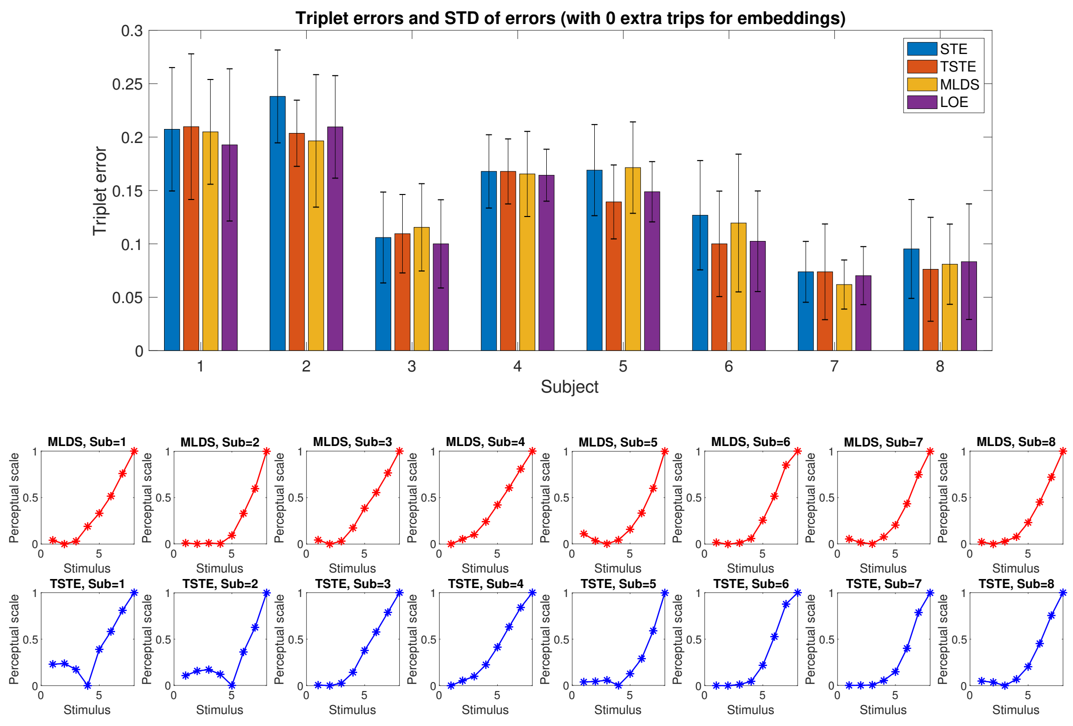


Figure 4.7: (Top) Average and standard deviation of cross-validated triplet error for 8 subjects of the slant-from-texture experiment. Each group of bar shows the error for one subject, as each bar in the group corresponds to one of the embedding methods shown with different colors. (Bottom) The embedding outputs for 8 subjects with two embedding methods: MLDS and t-STE. The MLDS method is depicted at the top row while the t-STE is shown at the bottom.

4.5 How to apply ordinal embedding methods in psychophysics

We introduced ordinal embedding and its capabilities in psychophysics. Here we present advice as rules of thumb to make it more applicable for a researcher who is unfamiliar with the methods. We describe how to perform a typical psychophysical scaling task using triplet questions and ordinal embedding methods. We organize this section by answering the questions arises in each step of applying the method.

4.5.1 How to make the subset of triplets?

Initially, we are given a set of n stimuli. At the fist step, one needs to consider the whole set of possible triplet questions. As we mentioned earlier in simulations, every combination of three items from the stimuli set gives rise to **three** triplet questions. Therefore, the complete set of possible triplet questions contains $3\binom{n}{3}$ triplets. The set of all possible triplet might be huge, thus a small subset of triplets needs to be subsampled. A natural question is ”which of the triplet questions among the whole set of possible questions should be chosen? Which subset is more informative?”. In chapter 5, we answer this question with various simulations and one experiment. We tried various strategies to sample a subset of triplets form the full set of triplets. However, in our experience a **random selection strategy** works best, meaning that it leads to the best performance of embedding algorithms with the fewest triplet questions possible.

4.5.2 How many triplets?

A small subset of triplets already contains enough information for the ordinal embedding. Formally, if the required embedding dimension is d, it is proven that $\mathcal{O}(dn\log(n))$ triplet answers are sufficient to reconstruct the true embedding of n items (stimulus levels) (Jain *et al.*, 2016). According to this result, we suggest to start with a subset of size $dn\log(n)$ and perform the ordinal embedding. If the embedding algorithm gives a desirable cross-validated triplet error, it means that we already fed enough triplets to the algorithm. However, in case of a high error, we increase the number of triplets until the embedding shows acceptable performance.

4.5.3 How to evaluate the quality of embedding?

We reported the MSE in our simulations; however, the true perceptual scale is not available in a real experiment. The general approach that we suggest for the evaluation of ordinal embedding is through the **cross-validated triplet error** (see Equation 4.3). The chosen subset of triplets needs to be partitioned into training and validation sets. The embedding algorithm finds a Euclidean embedding for the perceptual scales, given the training set of triplet as input. Having the estimated embedding, we calculate the cross-validated triplet error based on the validation set. This procedure is preferable as it can avoid the overfitting of the embedding.

4.5.4 How to choose the embedding dimension?

In our ordinal embedding formulation, we always assumed that the embedding dimension is given as input. However, in practice this information might be not provided. Note that the embedding dimension refers to the perceptual dimension which is in general unknown to the psychophysicist. We suggest to run the embedding algorithms in various dimensions and choose the smallest dimension which shows an acceptable cross-validated triplet error. Indeed, increasing the dimension can always produce less triplet error. However, we should only count significant improvements of the error to choose the proper embedding dimension.

4.5.5 Which algorithm, which implementation?

Considering the results of the various algorithms on the many tasks, we believe that t-STE is the general recommended method to be used in psychophysical scaling experiments based on the triplet questions. The original implementation of the authors is available at `https://lvdmaaten.github.io/ste/Stochastic_Triplet_Embedding.html` implemented in MATLAB. In addition, there exists a Python implementation at `https://github.com/gcr/tste-theano`.

4.6 Discussion

In this chapter, we introduced the ordinal embedding methods, proposed in the machine learning literature, as a powerful approach to perform psychophysical scaling tasks. The recommended ordinal embedding methods require a small subset of triplet comparison. This property makes them preferable to the traditional NMDS which needs the total rank order of the pairwise distances. On the other hand, ordinal embedding methods are capable of embedding in multi-dimensional Euclidean spaces without restrictions on the scaling function. Hence, they have the advantage over the MLDS method of psychophysics. In spite of above mentioned benefits, there are a few open issues regarding the usage of ordinal embedding methods that we mention in the following.

4.6.1 Open issues

Confidence intervals: There has been considerable efforts to propose algorithms for the ordinal embedding problem. However, there exists no particular study which provides confidence intervals for the estimated embeddings. Although this issue is not taken very seriously in machine learning, for the psychophysics studies it is of high importance.
Interpreting the embedding: A challenging yet important step is to interpret the embedding results. After gathering an embedding (possibly in a multi-dimensional perceptual space), and a mapping of stimuli in this space, there are a couple of natural questions arising. What does each perceptual dimension mean? How are the perceptual dimensions related to the variations of the physical stimulus? These are essential questions which can lead to better understanding of human perception.
Conjoint measurement: In addition to the general scaling problem, we believe that the ordinal embedding is a very good candidate to tackle conjoint measurement problems of psychophysics. In a conjoint measurement experiment the sensory stimulus consists of more than one modality. The ordinal embedding treats all the sensory stimuli as abstract items. Therefore, one can possibly apply the ordinal embedding methods with less restrictions, such as the independence or additivity assumptions, which are common assumptions for methods dealing with the conjoint measurement problem.

Chapter 5

Psychophysical scaling using crowd-sourcing platforms

5.1 Introduction

Since the earliest studies of psychophysics, precise stimulus control has been considered as the *"first commandment of psychophysics"* (Geisler, 1987, p. 30), and the quantitative study of human behavior preferred accuracy over quick (and painless) data acquisition: *"The search for short-hand methods in technology is laudable enough, but it is entirely out of place in science, where new trails are being blazed and attempts are made to reduce and to eliminate all of the errors of observation. Ease and convenience are poor experimental guides."* (Dallenbach, 1966, p. 656).
Nonetheless, crowdsourcing experiments are becoming increasingly popular, and there have been positive discussions of using Amazon Mechanical Turk (MTurk) in behavioral experiments (e.g. Chandler *et al.*, 2014; Marder and Fritz, 2015), and suggestions that crowdsourcing may, for some purposes, even be better than traditional laboratory experiments in terms of broader demographics (c.f. Henrich *et al.*, 2010). However, the worry whether "behavioral facts" could ever be obtained without (much) control over hardware, observer concentration, attention, viewing distance, language competence, etc., or even distractions by peers or children and multi-tasking while doing online experiments, are very serious indeed.
It seems quite clear that a naive approach of simply re-implementing traditional psychophysics experiments on MTurk is likely to fail: the traditional tools such as the just noticeable difference (JND) framework have been developed with the intention to be used in well-controlled lab environments, as they crucially rely on the fine control of the stimuli, and on highly-trained observers. From a purely methodological point of view the reliance on JND-style experiments is reasonable: JND-style data, particularly using the method of forced-choice, are the most reliable and robust estimates of human behavior (Blackwell, 1952; Jäkel and Wichmann, 2006). It would not come as a surprise if such methods failed in crowdsourcing setups.

For this reason we believe it misleading to ask *whether* it is possible to perform psychophysics in a crowdsourcing setup; one would have to ask *how the basic methods in psychophysics need to be changed in order to enable crowdsourcing as a new tool for psychophysics*.
In this chapter, we hypothesize that data acquisition by means of triplet comparisons, combined with the ordinal embedding, is a suitable approach for performing psychophysics tasks on crowdsourcing platforms. The immediate benefit comparison-based setting is that the statements may be less dependent on the fine details of the experimental setup, and that the issue of scaling answers across many diverse participants becomes easier. However, there are two major points that we address in this chapter:

1. **Subsampling approach for triplets:** In previous chapter, Section 4.5.1, we mentioned that random triplet subsampling is empirically shown to be the best choice. Here, we examine this point through simulations and one experiment. We use different subsampling strategies and different ordinal embedding algorithms and compare them in three different setups (simulations, lab, MTurk).

2. **Lab versus crowdsourcing:** We run the same psychophysics task in a well-controlled lab environment (few participants, each participant measured on a number of different days, and without distractions) and on MTurk (many participants, none of them highly motivated, and each participant only measured once) and compare the results.

As the main contribution of this chapter, we provide empirical results that suggest the feasibility of crowdsourcing with the comparison-based approach for psychophysics experiments as an easier alternative to well-controlled lab experiments. We open doors towards using the comparison-based approach in crowdsourcing platforms. In addition we perform the simulations, as well as lab and MTurk experiments, using various triplet subsampling strategies. We then compare the performance of subsampling strategies, in order to find out the best strategy. Before moving to the simulation and experiments, we discuss different triplet subsampling strategies used in this chapter.

5.2 Triplet subsampling strategies

In our simulations and experiments, we use three different *subsampling strategies* to generate a subset of *m* triplets to be used as input to ordinal embedding algorithms:

1. **Random:** We sample *m* triplets uniformly at random from the set of all possible triplets.

2. **l-repeated-random:** We sample m/l triplets uniformly at random from the set of all possible triplets, and ask each triplet l times. Answers to triplets that are systematically repeated l times are aggregated via majority vote, i.e., the answer to a triplet that we provide as input to the embedding algorithm is the same that is given in more than $\frac{l}{2}$ of the l repetitions of that same triplet (note that we always choose l to be odd).

3. **Landmark:** We fix a small number of landmark objects $\ell_1, ..., \ell_k$, then ask all triplet questions of the form (x, ℓ_i, ℓ_j). The number of landmark points, k, is chosen in a way that it leads to approximately m triplet questions.

5.3 Simulations

5.3.1 Simulation setup

Since the aim of this section is to investigate the subsampling, we choose a very simple and generic dataset. The dataset consists of n items, sampled uniformly at random in $[0,1]^3$. The procedure to generate the triplet questions and the answers to them is as follows: we first sample a set of m triplets over the items of the dataset (with different subsampling strategies). The three subsampling strategies are used in separate experiments. The answers to the set of subsampled triplets are generated according to the actual Euclidean distances of points combined with a simple noise model. The noise model perturbs the correct triplet answers (as given by the Euclidean distance) as follows: for a fixed noise probability $p < 0.5$, we flip the answers to triplet questions with probability p, for each triplet independently.
The noisy triplet answers are given as input to different ordinal embedding methods. We tried various ordinal embedding methods, including: Generalized Non-Metric Multidimensional Scaling (GNMDS, Agarwal *et al.*, 2007), (t-)Stochastic Triplet Embedding (STE/t-STE, van der Maaten and Weinberger, 2012) and Local Ordinal Embedding (LOE, Terada and Luxburg, 2014). In case of our dataset, the LOE method outperforms the other algorithms. Thus, we only report the comparison of different strategies performed with the LOE method.
The efficiency of triplet subsampling strategies is measured by means of *triplet prediction accuracy*. Triplet prediction accuracy has a similar definition to the cross-validation triplet error, defined in Equation (4.3). In addition to the set of m triplet answers, which is used for ordinal embedding, an independent validation set is also constructed. The ratio of triplet answers in the validation set which are consistent with the ordinal embedding output is defined as the triplet prediction accuracy. In this way: triplet prediction accuracy $\approx 1 -$ cross-validated error. The relation is approximate as in the cross-validation procedure the folds are fixed. However, to calculate the prediction accuracy, we randomly assign triplets to the validation set.

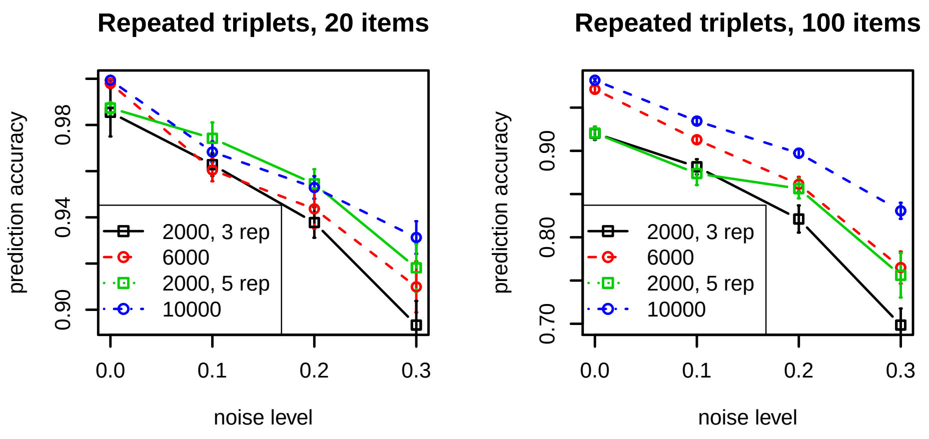


Figure 5.1: The comparison of the l-repeated-random triplet subsampling strategy against the random strategy. The number of total items in the dataset is 20 and 100, for the left and right plot respectively. The x-axis corresponds to varying noise parameters p, while the y-axis denotes the prediction accuracy. Each curve in the plot corresponds to a subsampling strategy and a fixed value of l (repetition of each triplet)

5.3.2 Result

We present the comparison of the three subsampling strategies in two parts. First we compare the performance of random subsampling against the l-repeated-random strategy. Secondly, we report the results for the comparison of the random strategy against the landmark strategy.

Random vs. l-repeated-random

We vary the number of items n, the number l of repetitions per triplet and the noise level p. We run the experiments with two values $n \in \{20, 100\}$. For the l-repeated-random strategy, we use 2000 triplets, each of them get answered $l \in \{3, 5\}$ times. In this way the total number of triplet question is $m = 2000 \cdot l$. In order to have a fair comparison, we use the same number of total triplets for the random triplet subsampling strategy.
In Figure 5.1, we show the results of triplet prediction with the l-repeated-random sampling and the random subsampling, for different values of l, n and the noise level p. The left plot corresponds to the experiment with $n = 20$ items. The noise level is denoted on the x-axis, and each curve shows the prediction accuracy of one strategy with a fixed value of l. In this plot, we see that repeated sampling and employing majority vote may slightly improve prediction accuracy if the noise level

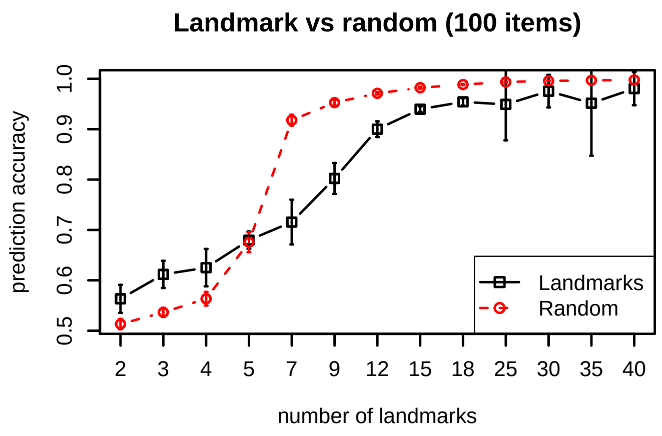


Figure 5.2: The comparison of the landmark subsampling strategy against the random strategy for $n = 100$ items. The x-axis corresponds to the number of landmarks k, used for the landmark strategy. Each curve in the plot corresponds to a subsampling strategy.

is moderate (0.1 or 0.2) and the number of triplets is relatively high in relation to the number of items. For a higher noise level, random sampling outperforms repeated sampling, which is in accordance with results by Lin *et al.* (2014) that repeated answers with majority voting may be beneficial especially for moderate noise levels. In a situation that we consider more interesting, namely if the number of given triplets m is low in comparison to the number of items n, relative to the number of items (right plot), repeated triplets lead to worse prediction accuracy than the same number of distinct triplets.

Random vs landmark

We make a dataset of $n = 100$ items similar to the previous part. Then, k landmarks $\ell_1, \ldots, \ell_k$ are selected uniformly at random from all $n = 100$ items. Next, we evaluate $t(x, \ell_i, \ell_j)$ for all $1 \leq i < j \leq k, x \in S \setminus \{\ell_i, \ell_j\}$ to produce input triplets for embedding. We use the same number of triplets, i.e. $\binom{k}{2}(n-2)$, chosen uniformly at random to construct another embedding for comparison. Similar to the repeated strategy, we only show LOE results in Figure 5.2, as it exhibits the best performance.

The results show that using the landmark strategy for triplet subsampling leads to a worse triplet prediction accuracy than random triplet sampling, except in a regime when very few triplets are known. In this regime, with very few triplets, the triplet prediction accuracy is hardly better than random guessing.

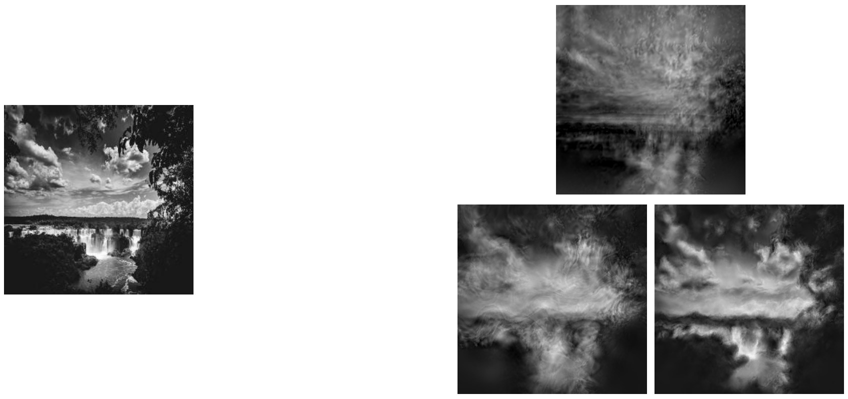


Figure 5.3: Left: the original (basis) waterfall image used for the eidolon experiment. Right: An example triplet question — *"Which of the bottom two images is more similar to the top image?"*

5.4 The eidolon experiment: Lab and MTurk

As a representative psychophysics task we choose a comparison task between images. To generate the images, we use the Eidolon Factory by Koenderink *et al.* (2017)—more specifically, its partially_coherent_disarray() function. In this toolbox, a given basis image can be distorted systematically, using three different parameters called *reach*, *grain* and *coherence*. An eidolon of a basis image then corresponds to a parametrically altered version of this image. Reach controls the strength of a distortion (the higher the value, the stronger the amplification), grain modifies how fine-grained the distortion is (low values correspond to 'highly fine-grained'), whereas a parameter value close to 1.0 for coherence indicates that "local image structure [is retained] even when the global image structure is destroyed" (Koenderink *et al.*, 2017, p. 10). From a psychophysics point of view, we want to know which and to what degree the image modifications influence the percept. Starting with a black and white image of a natural landscape as basis image (see Figure 5.3, left), we generate 100 altered images, using reach and grain in $\{5, 12, 26, 61, 128\}$ and coherence in $\{0.0, 0.33, 0.67, 1.0\}$. All possible combinations of these parameter values result in $5 \cdot 5 \cdot 4 = 100$ different images.

5.4.1 Lab experiment setup

In our psychophysical lab, we ask eight participants (4 male, 4 female, aged 19 to 25, mean 21 years) to answer triplet questions. See Figure 5.3 (right) for an example triplet question. For this purpose, participants use a standard computer mouse to click on one of the two bottom images that they deem more similar to the top image. Stimuli are presented on a 1920×1200 pixels (484×302 mm) VIEWPixx LCD monitor (VPixx Technologies, Saint-Bruno, Canada) at a refresh rate of 120 Hz in an otherwise dark chamber. Viewing distance is 100 cm, corresponding to 3.66×3.66 degrees of visual angle for a single 256×256 pixels image. The surround of the screen is set to a grey value of 0.32 in the $[0,1]$ range, the mean value of all experimental images. The experiment is programmed in MATLAB (Release 2016a, The MathWorks, Inc., Natick, Massachusetts, United States) using the Psychophysics Toolbox extensions version 3.0.12 (Brainard, 1997; Kleiner *et al.*, 2007) along with the iShow library (`http://dx.doi.org/10.5281/zenodo.34217`).

Answers have to be given within 4.5 seconds after a triplet presentation onset, otherwise the triplet will be registered as unanswered and the experiment will proceed to the next triplet. (this occurred in 0.013% of all cases only). The experiment is self-paced, i.e., once a participant answers a question, the next one will appear directly after a short fixation time of 0.3 seconds, during which only a white 20×20 pixels fixation rectangle at the center of the screen is shown. Before the experiment starts, all test subjects are given instructions by a lab assistant and practice on 100 triplets to gain familiarity with the task. The set of practice triplets is disjoint from the set of experimental triplets. Participants are free to take a break every 200 triplet questions. They give their written consent prior to the experiment and are either compensated with €10 per hour for their time or gain course credit towards their degree. All test subjects were students and reported normal or corrected-to-normal vision.

We implement three triplet subsampling strategies: random, 3-repeated-random and landmark. Every participant answers 6000 triplet questions in three approximately 75-min sessions of 2000 triplets each. In the landmark subsampling experiment we have a fourth session during which participants answer 1500 triplet questions as a test set. Subjects 1, 2 and 3 each answer 6000 triplets, which are sampled uniformly at random (with repetitions). Subjects 4, 5 and 6 each answer 2000 triplets which are repeated 3 times. The triplets for the first sessions of subjects 1 and 4 are identical and presented in the same order. In their second and third session, subject 4 answers the same triplets as in their first session, but the order is randomized in each session. In the same way, subject 5 answers a subset of subject 2's triplets and subject 6 answers a subset of subject 3's triplets.

The triplets for subjects 7 and 8 are selected according to the landmark strategy. We select 12 landmarks, giving rise to $\binom{12}{2}(100-2) = 6468$ triplets, from which we subsample 6000 triplets that are presented to the subjects over three sessions. The landmarks are picked by hand, such that they provide a good covering of the parameter space. The triplets are identical for both subjects. Subjects 7 and 8 also answer 1500 triplets sampled uniformly at random, as a test set.

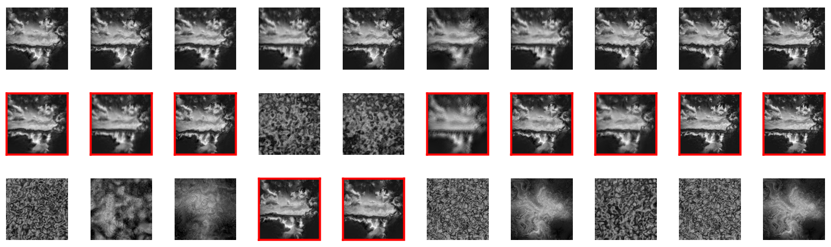


Figure 5.4: 10 out of 20 triplet questions used as Gold Standards in one of the MTurk experiments. Each column corresponds to one triplet question. The first row is the query image. The second and third row show the options of the participant. The obvious answers are denoted by red frames

5.4.2 MTurk setup

In order to achieve a fair comparison between the lab and crowdsourcing settings, we design our crowdsourcing experiment on MTurk to be as similar as possible to the lab experiment. To this end, we create a survey page on our local webservers to which all participants get redirected. The page layout, background color and stimuli presentation times are chosen as in the lab experiment.

Similar to the lab experiment, we generate sessions consisting of 2000 (2500 for the landmark setting) triplet questions. In the case of experiments with test sets we increased the task size to 2500 questions. The allocated time for sessions is 4 hours, however a session is done in less than 2 hours on average. To obtain datasets of 6000 triplets, one needs three consecutive sessions from each participant, taking more than 6 hours. In the lab, these sessions were held on separate days. Conducting many repeated sessions with the same participant and precise time schedules is the strength of the lab, which we sacrifice by using the crowd on MTurk. On MTurk, we do not conduct long tasks of 6000 triplets.

To filter out participants that may just give random answers to our triplet questions, we implement a sanity check (validation procedure). We pick 20 triplet questions with very obvious correct answers in every session and mark them as our *Gold Standard Questions* (see Figure 5.4 for an example set of 10 gold standard questions). We excluded participants (and their corresponding session) from the evaluation if their error rate on those questions was larger than 20%. This was the case for 7 out of 60 sessions (12%); thus it is as well to note that in a crowdsourcing setting the exclusion of—for whatever reason—poorly performing participants is an issue that typically does not, or to a much lesser extent, affect lab experiments. This is a potentially thorny issue, as one should not, of course, exclude participants because their behavior does not align well with one's hypothesis. Sanity checks for crowdsourcing experiments certainly deserve more thought in the future.

We run three sets of experiments, corresponding to our three strategies of triplet subsampling. In the first set of experiments, we ask 2000 uniformly random triplets from 30 participants. Out of those, 23 participants passed the sanity check in the end. In the second set of experiments we implement the 3-repeated-random subsampling strategy in 15 sessions each consisting of 2000 triplets. We fix 5 sets of 2000 random triplets and ask each of them three times. All these participants were accepted in the end. The third set of experiments is the landmark setting. Here we use exactly the same triplets as the lab experiments. We divide the 6000 landmark triplets and 1500 test triplets each into 3 sets. In this way, each participant has to perform a session consisting of 2000 landmark triplets and 500 test triplets. We have 15 sessions for this task, and all of them were validated.

5.4.3 Results

For the eidolon experiment, we do not have ground truth information about the distances. Therefore, we can only measure how well we can predict answers to a set of test triplets using a set of training triplets (triplet prediction accuracy), in the knowledge that the answers to the test triplets may be noisy. The 3-repeated-random setting provides some information about that noise level: For lab subjects 4, 5 and 6, for each triplet we can take the majority vote of the answers for that triplet. The agreements between triplet answers and majority votes over these subjects are 90.8%, 90.8% and 89.0%, respectively. As a result, the highest accuracy that we can expect for predicting participants' triplet answers is about 90%.

We have made two observations that apply to all of the following results. First, the t-STE method consistently outperforms other methods for triplet prediction. Secondly, perhaps surprisingly, prediction accuracy is the best for embedding in 2 dimensions, even though images are generated using 3 parameters. This may be due to interaction of the parameters. At the end of this section, the results of embedding with various methods and in different dimensions, ranging from 1 to 6, are reported. The rest of experiments are conducted with t-STE and embedding in 2 dimensions.

In the following we first report the comparison of lab and MTurk data. In a separate subsection, we examine the performance of various subsampling strategies using the lab data. Then, we present cross-subject analysis of triplets in the random subsampling strategy. Next, we examine if pooling (accumulating) the triplet answers of various subjects leads to better triplet prediction accuracy. Finally, the results of embedding in various Euclidean dimensions are reported

Lab vs. MTurk

Random: The two left boxes in Figure 5.5 (a) show the triplet prediction accuracy for the lab and MTurk experiments. We use 1500 random triplets of each participant for the embedding and 250 triplets for evaluation. For the MTurk experiments we report the performance based on 23 participants who passed the sanity check.

The results for each participant, for the random subsampling scenario in both lab and MTurk setup, are reported in Figure 5.6. Each point corresponds to one session of 2000 triplets. The MTurk participants are sorted based on the performance of t-STE method on the horizontal axis. The last 7 subjects are exactly the same subject who did not pass the sanity check. Most of the other MTurk participants have acceptable accuracy. This figure is the expansion of the two left box plots presented in Figure 5.5 (a).

Repeated-Random: For the training set we merge 1500 triplets from each session which we ask 3 times. In this way the training set contains 4500 triplets. As the test set we choose 250 random triplets from the remaining triplets of the 3 repeated sessions. The two middle boxes in Figure 5.5 (a) show the triplet prediction accuracy for both MTurk and lab experiments in this setting.

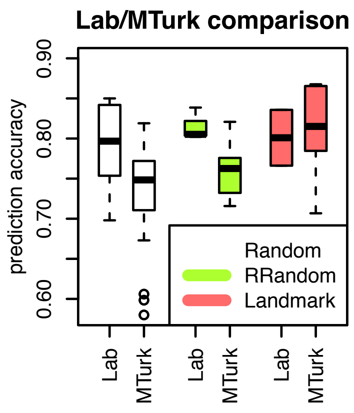
Landmark: The training set contains 6000 triplets of 3 sessions. The test set contains 1500 triplets. The two right boxes in Figure 5.5 (a) show the triplet prediction accuracy in this setting.

Figure 5.5 (a) shows that in most cases the MTurk participants' responses can be predicted with about 5 percent less accuracy than those of the lab participants. Furthermore, MTurk data have a higher variance. Clearly, lab data are of higher quality than MTurk data; however, the difference is certainly not dramatic, and one can imagine scenarios where the ease of data acquisition and the much larger sample of participants outweights the (slightly) lower data fidelity.

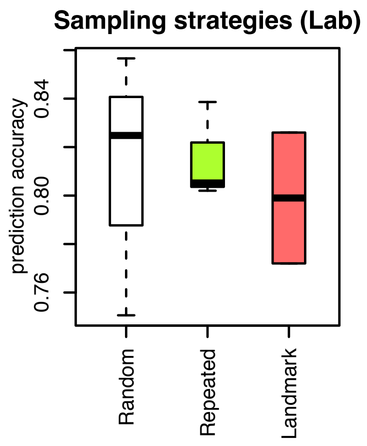
Comparison of subsampling strategies

We also report the prediction accuracy of various subsampling strategies for the lab experiments in a comparable setting in Figure 5.5 (b). We use 4500 triplets for training, selected according to the respective strategy, i.e., random or 1500 triplets three times each, or 4500 triplets from the set of landmark triplets, and test prediction accuracy on a test set of 250 random triplets. Note that this analysis is done based on the lab experiments.

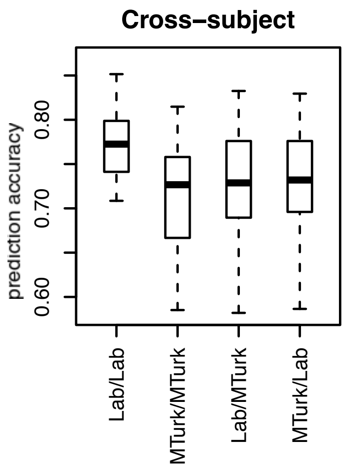
Similar to the results of simulation, the random subsampling strategy outperforms the other two strategies. The variance of prediction accuracy is relatively small for all three strategies.



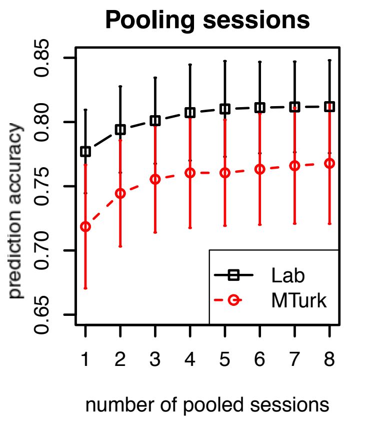
(a)



(b)

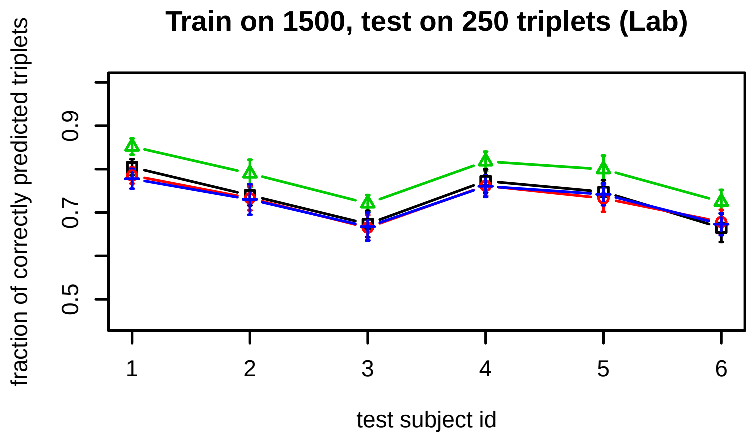


(c)



(d)

Figure 5.5: (a) Comparison of triplet prediction accuracy for MTurk vs. lab in 3 subsampling strategies. Note that boxes with different colors are not comparable since they have different training sizes. (b) Prediction accuracies for 3 various subsampling strategies from the lab experiment in a comparable setting. (c) Cross-subject triplet prediction accuracies. (d) Prediction accuracy with respect to pooling size.



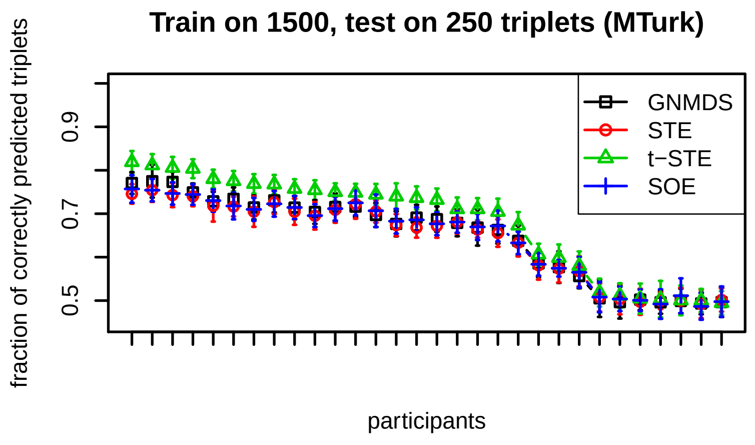


Figure 5.6: Triplet prediction accuracy for each subject of lab and MTurk experiments. We check how well the triplets of a single subject can predict the triplets of the same subject. From the set of 2000 triplets of a subject (we only use the first 2000 for lab experiment participants), we select 250 triplets at random as validation set, select 1500 triplets from the rest as training set for embedding and report the accuracy on the validation set. The accuracies are averaged over 20 repetitions. Top: Lab participants 1–6; Bottom: MTurk participants, sorted by t-STE triplet prediction accuracy.

Cross-subject analysis

Assuming a ground truth embedding over the eidolon images, we investigate the consistency of the triplets of one participant with the embedding gathered from another participant. We focus on the sessions with random triplets. We use 9 sessions from the first three subjects in the lab experiments and 23 validated sessions from the random strategy in the MTurk experiments. We perform this analysis in 4 parts: Lab/Lab, Lab/MTurk, MTurk/Lab and MTurk/MTurk. Lab/MTurk means that all combinations of sessions from MTurk and sessions from the lab are used for training and testing, respectively. Box plots in Figure 5.5 (c) show the prediction accuracies in four parts of the experiments.

The average accuracy in the first two parts (Lab/Lab and MTurk/MTurk) is only slightly worse than the experiments on the same subject (Figure 5.5 (c) two left boxes). This strongly suggests that participants—both in the Lab and on MTurk—perceive the similarities between images in a consistent manner. The prediction accuracies convinced us that visual perception in our experimental scenario can be captured in a Euclidean space with two dimensions. Considering this embedding space participants show very similar interpretation of the similarities. The difference between Lab and MTurk is likely due to the noise and uncontrolled setting of data gathering in MTurk. Again the MTurk experiments show a higher variance—this, clearly, is a drawback of crowdsourcing in comparison with lab experiments: Whilst the average difference in prediction accuracy is only around 5%, the difference is larger than 10% for the most difficult to predict observers.

Pooling triplets

In this section, we examine if pooling many triplets can improve the triplet prediction accuracy. We only include triplets from the random strategy, namely 9 sessions from subjects 1 to 3 in the lab and 23 sessions from the MTurk participants who pass the sanity check. We run this experiment on 20 trials. In each trial, we permute the list of sessions in both lab and MTurk session lists. The first session of the permuted list is then used as the test set and the remaining ones are added to the pool of training set one by one. We report the average and standard deviation of prediction accuracy over 20 trials in Figure 5.5 (d) with respect to the size of training pool. We see a similar pattern in pooling of the sessions for both lab and MTurk experiments. We can roughly gain about 5 percent in accuracy with 4 sessions. However, adding more sessions can not help to improve. We again see about 5 percent difference in the accuracy of MTurk pooled sessions in comparison to the lab pooled sessions.

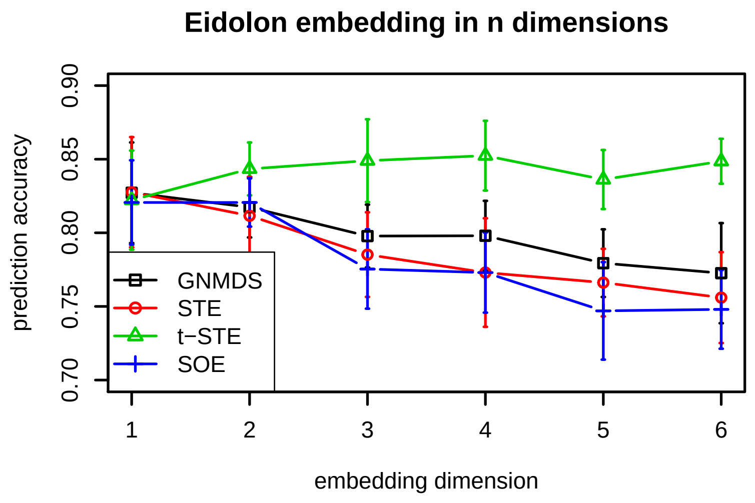


Figure 5.7: Triplet prediction accuracy of subject 1 in the lab experiment, using different embedding methods and embedding dimensions. 1500 triplets from lab subject 1 are used for training, 250 for testing (as in several of our evaluations).

Embedding dimension

Here, we report the prediction accuracy of various embedding methods with different embedding dimensions. Figure 5.7 shows the prediction result of embedding for subject 1 of the lab experiment with 1500 triplets as training set and 250 triplets as test set. Increasing the dimension from 1 to 2 improves the prediction accuracy significantly, however more than 2 dimensions cannot lead to a considerable improvement. We also observe that t-STE outperforms the other methods in all dimensions. The results for other subjects have slight differences, however, in most cases the best accuracy is obtained with 2 dimensions.

Appendix A

Supplementary Material

A.1 Extended simulation results of Chapter 4

Monotonic scaling functions: Here we present extensive results of the simulations with the monotonic scaling functions. Figure A.1 shows the extended results of the embeddings for a Sigmoid function. Beside the MSE and triplet error we also show the standard deviation of these two criteria. We again see that MLDS works slightly better than other methods, as the scaling function meets the assumptions of MLDS. We also report the results for the same experiment with a different scaling function (See Figure A.2. The function is a conditional degree 3 polynomial. Again, the MLDS outperforms all other embeddings.

Non-monotonic scaling functions: Similar to the monotonic functions, we compare the performance of the embedding methods on two non-monotonic function. Figure A.4 shows the extra results for a degree two polynomial scaling function. The MSE and triplet error both show poor performance of MLDS in comparison with the embedding methods. The second scaling function is a Sinusoid. The results of comparison are shown in Figure A.3. The ordinal embedding methods perform clearly better than the MLDS, however the difference is not as big as the previous function. The reason might be that the Sinusoid is closer to a monotonic function than the second degree polynomial.

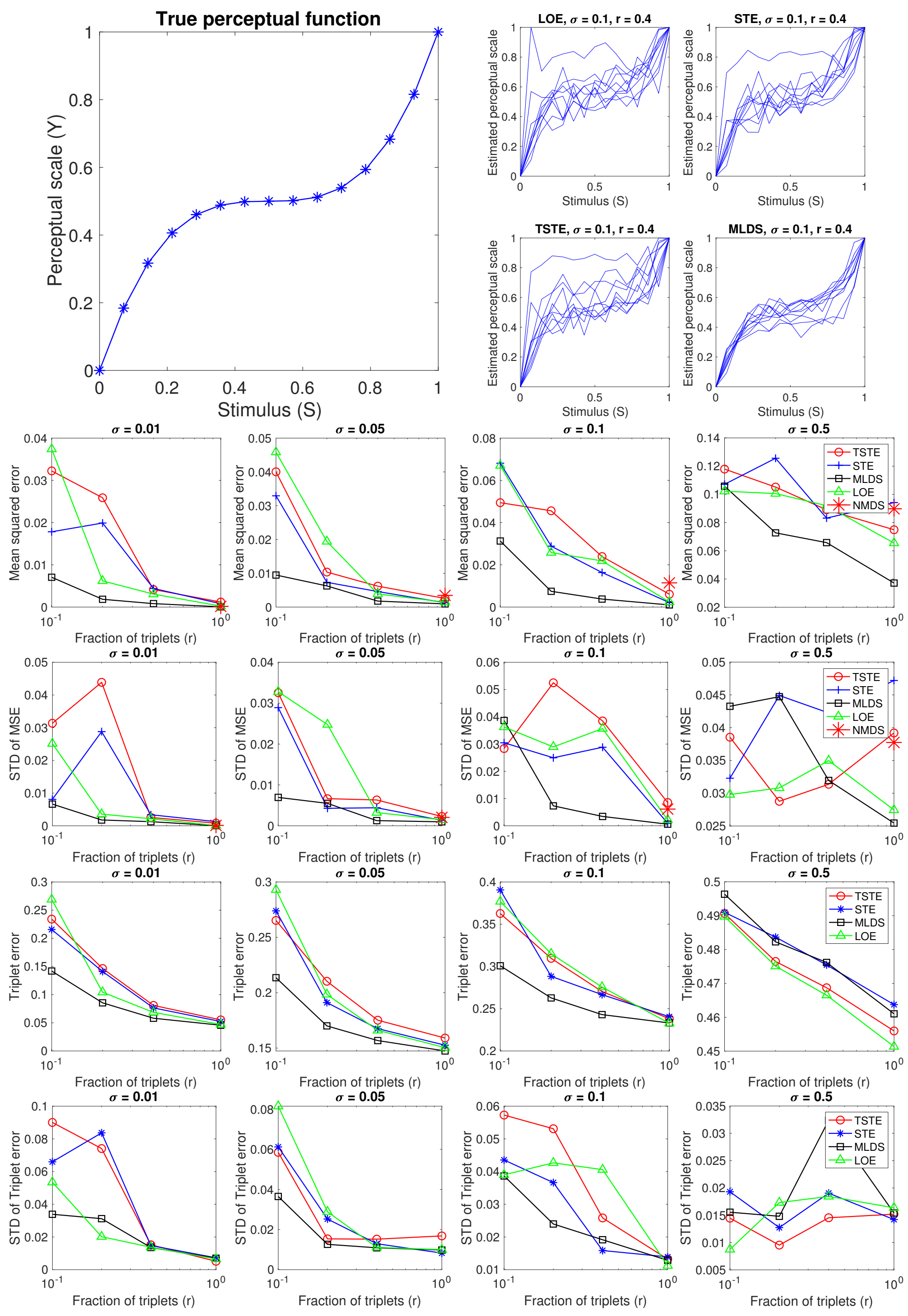


Figure A.1: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional methods in psychophysics (MLDS and NMDS) for a monotonic one-dimensional perceptual function. The true perceptual function (y) is appeared at the top left corner. Ten embedding results ($\hat{y}$) for a fixed value of noise standard deviation (σ) and triplet fraction (r) is shown on the top right corner. The second and third row depict the average MSE and the standard deviation of MSE for 10 runs of the algorithms. The fourth and fifth row show the average triplet error and the standard deviation of triplet error for 10 runs of the algorithms.

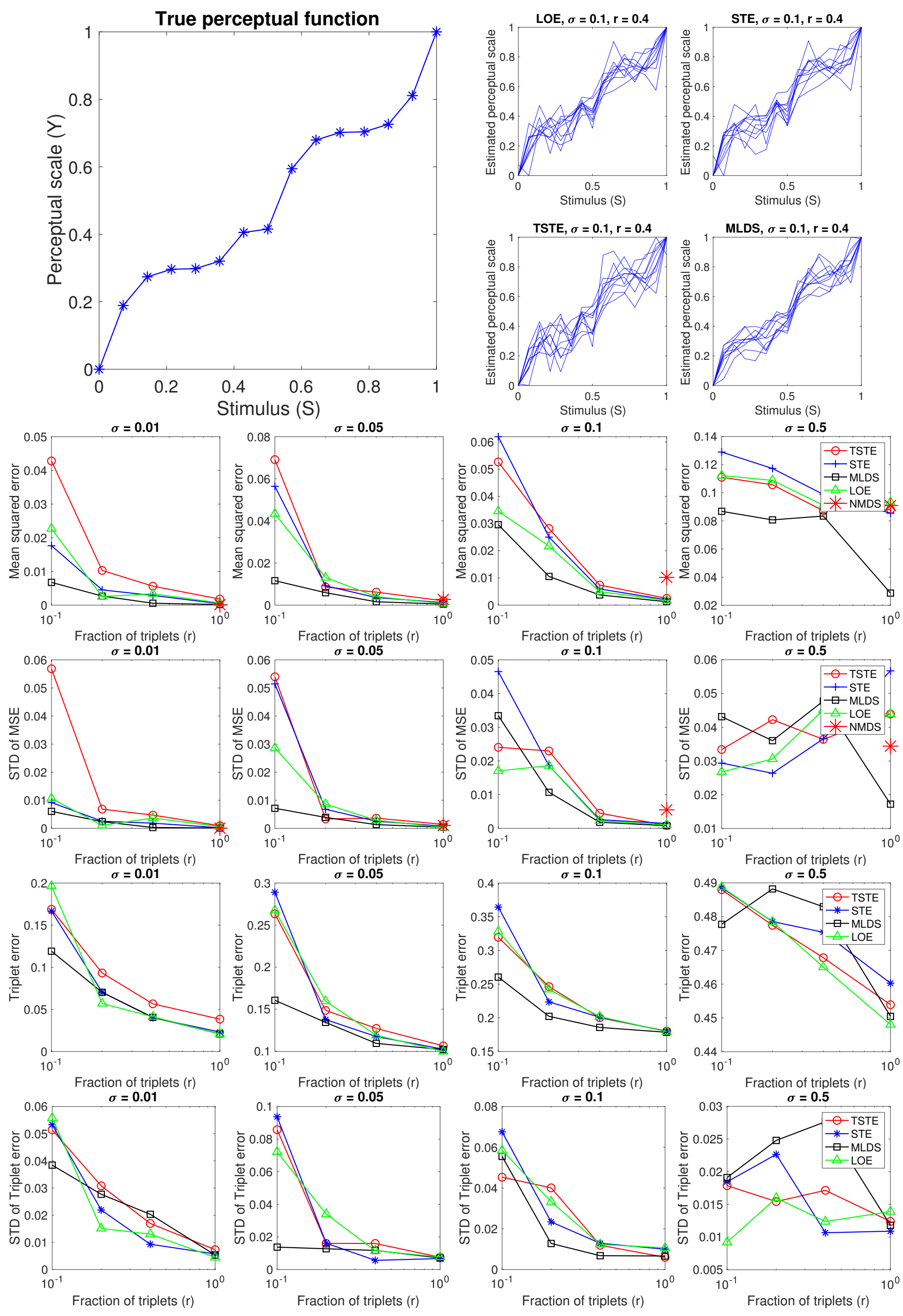


Figure A.2: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional methods in psychophysics (MLDS and NMDS) for a monotonic one-dimensional perceptual function. The true perceptual function (y) is appeared at the top left corner. Ten embedding results ($\hat{y}$) for a fixed value of noise standard deviation (σ) and triplet fraction (r) is shown on the top right corner. The second and third row depict the average MSE and the standard deviation of MSE for 10 runs of the algorithms. The fourth and fifth row show the average triplet error and the standard deviation of triplet error for 10 runs of the algorithms.

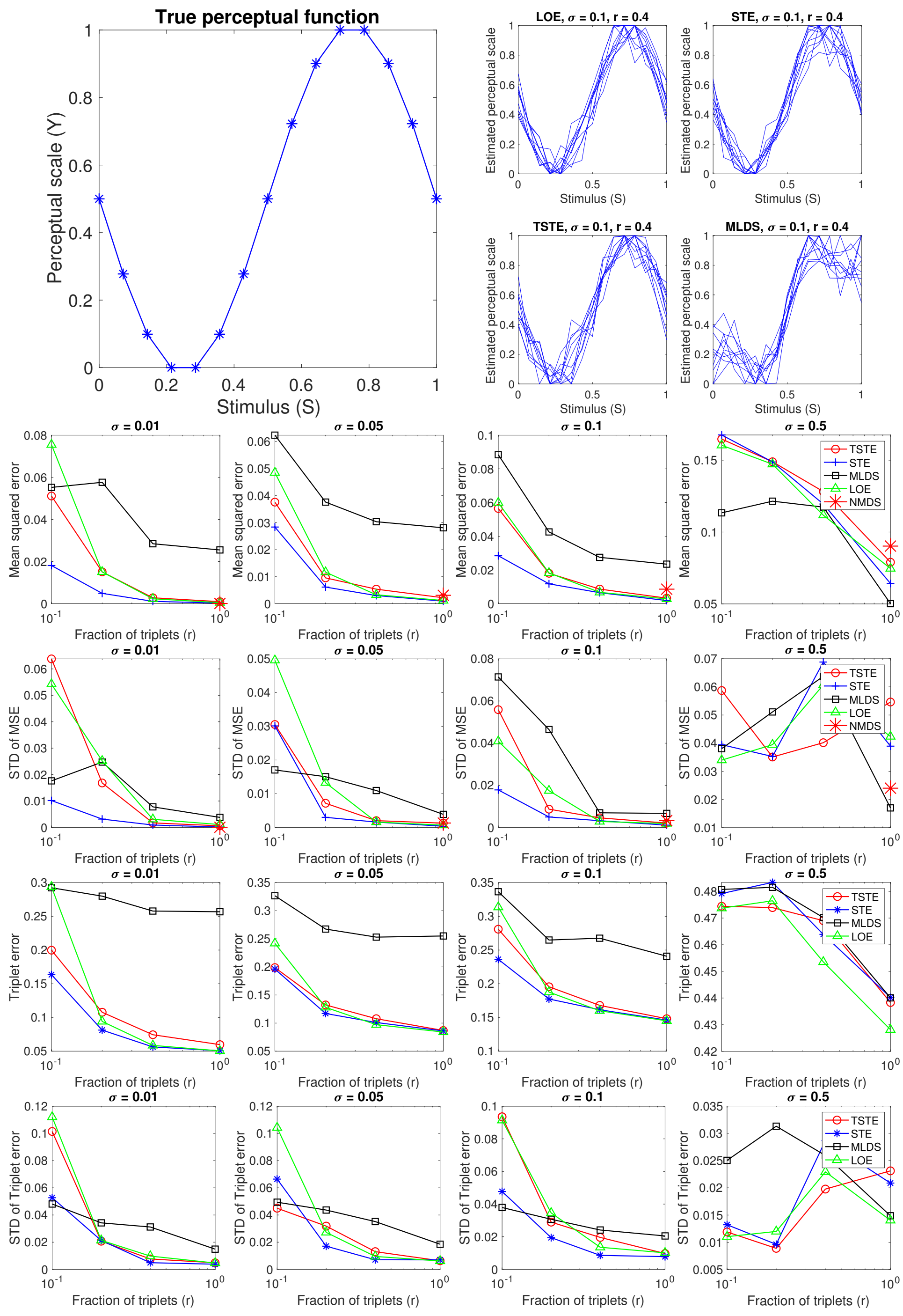


Figure A.3: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional methods in psychophysics (MLDS and NMDS) for a non-monotonic one-dimensional perceptual function (Sinusoid). The true perceptual function (y) is appeared at the top left corner. Ten embedding results ($\hat{y}$) for a fixed value of noise standard deviation (σ) and triplet fraction (r) is shown on the top right corner. The second and third row depict the average MSE and the standard deviation of MSE for 10 runs of the algorithms. The fourth and fifth row show the average triplet error and the standard deviation of triplet error for 10 runs of the algorithms.

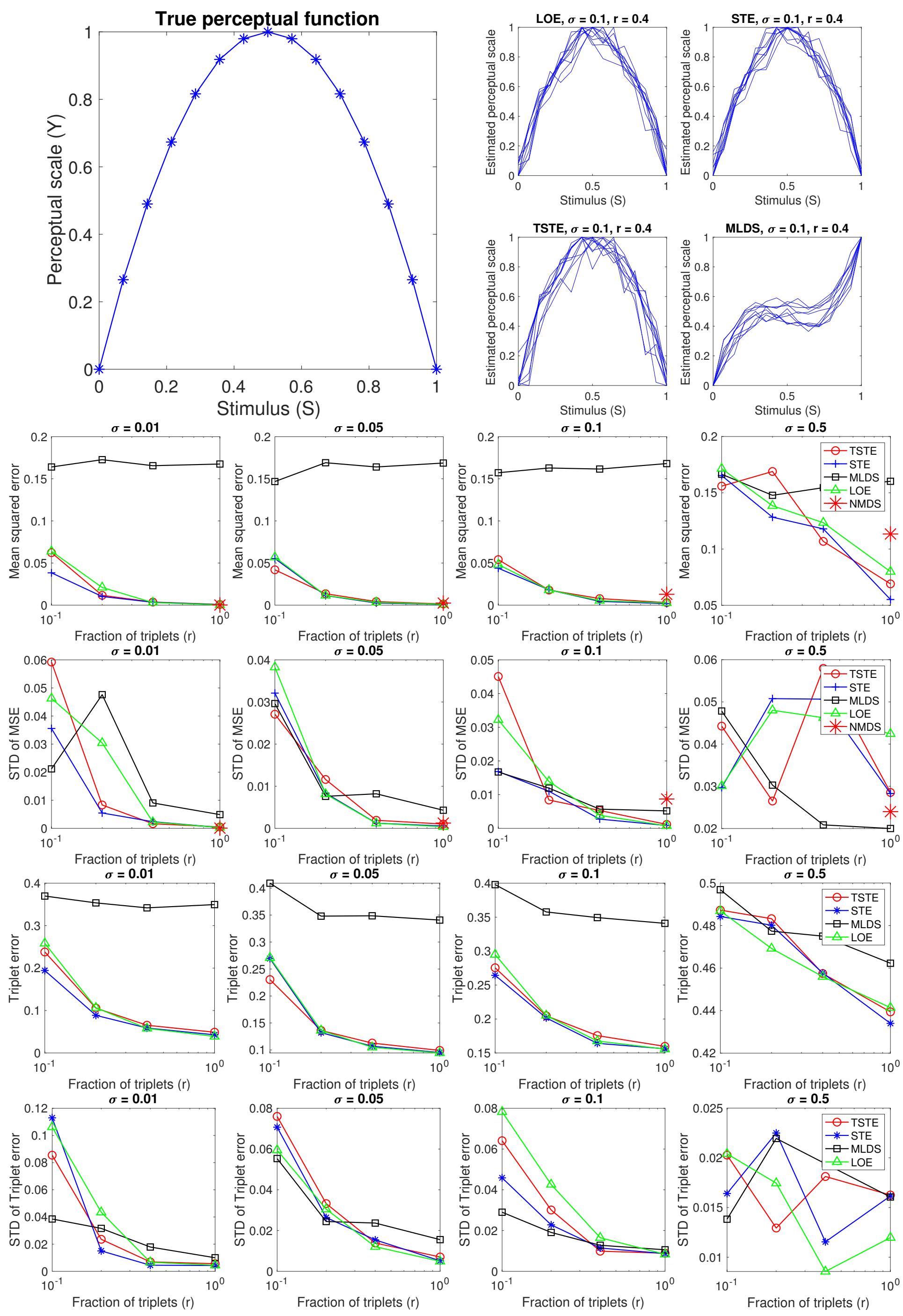


Figure A.4: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional methods in psychophysics (MLDS and NMDS) for a non-monotonic one-dimensional perceptual function. The true perceptual function (y) is appeared at the top left corner. Ten embedding results ($\hat{y}$) for a fixed value of noise standard deviation (σ) and triplet fraction (r) is shown on the top right corner. The second and third row depict the average MSE and the standard deviation of MSE for 10 runs of the algorithms. The fourth and fifth row show the average triplet error and the standard deviation of triplet error for 10 runs of the algorithms.

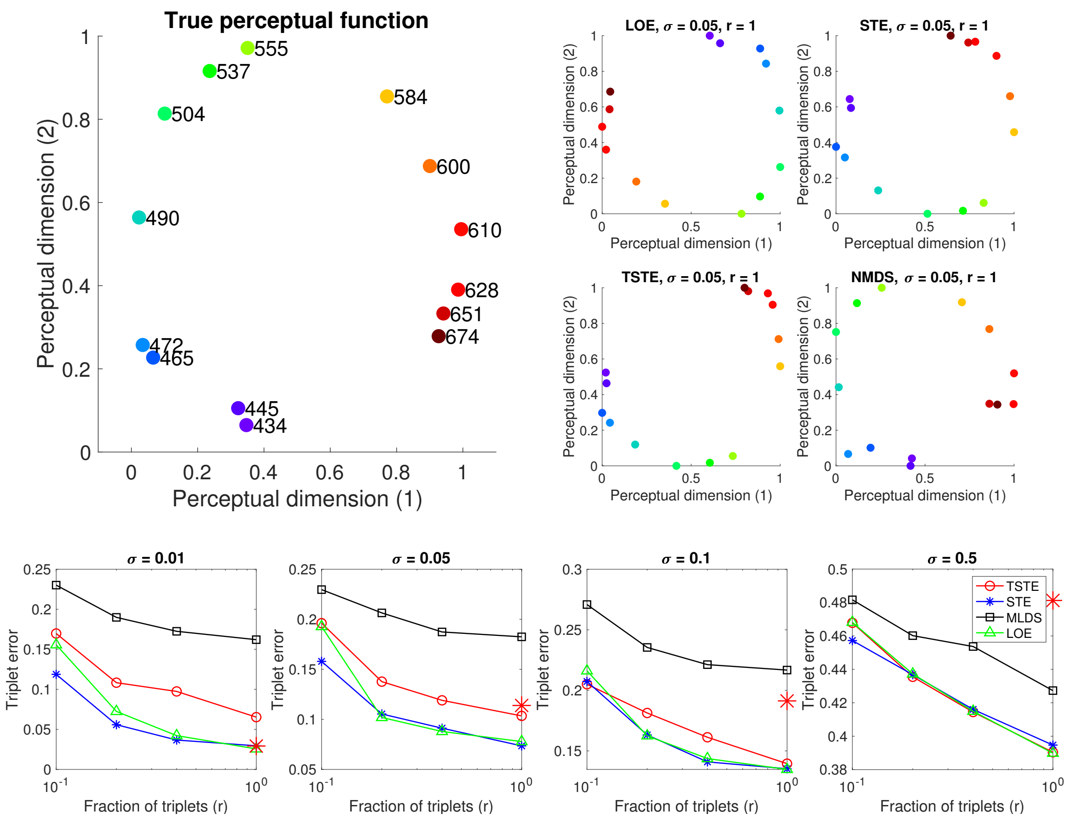


Figure A.5: The comparison of various ordinal embedding methods (LOE, STE, t-STE) against the traditional methods in psychophysics (MLDS and NMDS) for a two-dimensional perceptual function. The true perceptual function (y) is appeared at the top left corner. one of embedding results ($\hat{y}$) for a fixed value of noise standard deviation (σ) and triplet fraction (r) is shown on the top right corner. The second row shows the average triplet error over 10 repetitions of the algorithms.

Bibliography

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

Aguilar, G., Wichmann, F. A., and Maertens, M. (2017). Comparing sensitivity estimates from MLDS and forced-choice methods in a slant-from-texture experiment. *Journal of Vision*, **17**(1), 37, 1–18.

Ailon, N. (2011). Active learning ranking from pairwise preferences with almost optimal query complexity. In *Advances in Neural Information Processing Systems (NIPS)*.

Amid, E. and Ukkonen, A. (2015). Multiview triplet embedding: Learning attributes in multiple maps. In *International Conference on Machine Learning (ICML)*.

Arias-Castro, E. *et al.* (2017). Some theory for ordinal embedding.

Assouad, P. (1979). Étude dune dimension métrique liéea la possibilité de plongements dans rn. *CR Acad. Sci. Paris Sér. AB*, **288**(15), A731–A734.

Balcan, M., Vitercik, E., and White, C. (2016). Learning combinatorial functions from pairwise comparisons. In *Conference on Learning Theory (COLT)*.

Barsalou, L. W. (2014). *Cognitive psychology: An overview for cognitive scientists*. Psychology Press.

Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 509–517.

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *International Conference on Machine Learning (ICML)*.

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research (JMLR)*, **13**(4), 1063–1095.

Biau, G. and Scornet, E. (2016). A random forest guided tour. *Journal of the Spanish Society of Statistics and Operations Research (TEST)*, **25**(2), 197–227.

Biau, G., Devroye, L., and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research (JMLR)*, **9**(9), 2015–2033.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blackwell, H. R. (1952). Studies of psychophysical methods for measuring visual thresholds. *Journal of the Optical Society of America*, **42**, 606–616.

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision*, **10**, 433–436.

Breiman, L. (1996). Bagging predictors. *Machine learning*, **24**(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and regression trees*. CRC press.

Chandler, J., Mueller, P. A., and Paolacci, G. (2014). Nonnaivete among amazon mechanical turk workers: Consequences and solutions for behavioral researchers. *Behavior Research Methods*, **46**(1), 112–130.

Coombs, C. H., Dawes, R. M., and Tversky, A. (1970). *Mathematical Psychology*. Prentice-Hall, New Jersey.

Cortez, P. and Morais, A. (2007). A Data Mining Approach to Predict Forest Fires using Meteorological Data. In *Portuguese Conference on Artificial Intelligence*.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, **47**(4), 547–553.

Dallenbach, K. M. (1966). The staircase-method critically examined. *The American Journal of Psychology*, **79**(4), 654–656.

Dasgupta, S. and Freund, Y. (2008). Random projection trees and low dimensional manifolds. In *Symposium on the Theory of Computing (STOC)*, pages 537–546.

Dasgupta, S. and Sinha, K. (2015). Randomized partition trees for nearest neighbor search. *Algorithmica*, **72**(1), 237–263.

Davis, T. A. and Hu, Y. (2011). The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, **38**(1), 1:1–1:25.

Demiralp, Ç., Bernstein, M. S., and Heer, J. (2014). Learning perceptual kernels for visualization design. *IEEE transactions on visualization and computer graphics*, **20**(12), 1933–1942.

Denil, M., Matheson, D., and Freitas, N. (2013). Consistency of online random forests. In *International Conference on Machine Learning (ICML)*.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Springer.

Ekman, G. (1954). Dimensions of color vision. *The Journal of Psychology*, **38**(2), 467–474.

Ellis, D. P., Whitman, B., Berenzweig, A., and Lawrence, S. (2002). The quest for ground truth in musical artist similarity. In *Third International Conference on Music Information Retrieval*.

Fechner, G. T. (1860). *Elemente der Psychophysik*. Breitkopf und Hrtel, Leipzig.

Fernandes, K., Vinagre, P., and Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research (JMLR)*, **15**(1), 3133–3181.

Geisler, W. S. (1987). Ideal observer analysis of visual discrimination. In *Frontiers of Visual Science: Proceedings of the 1985 Symposium*, pages 17–31. National Academy Press, Washington, DC.

Gescheider, G. A. (1988). Psychophysical scaling. *Annual review of psychology*, **39**(1), 169–200.

Goyal, N., Lifshits, Y., and Schütze, H. (2008). Disorder inequality: a combinatorial approach to nearest neighbor search. In *International Conference on Web Search and Data Mining (WSDM)*.

Gupta, A., Krauthgamer, R., and Lee, J. R. (2003). Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 534–543.

Haghiri, S., Ghoshdastidar, D., and von Luxburg, U. (2017). Comparison-based nearest neighbor search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Haghiri, S., Garreau, D., and Luxburg, U. (2018). Comparison-based random forests. In *International Conference on Machine Learning (ICML)*.

Henrich, J., Heine, S. J., and Norenzayan, A. (2010). The weirdest people in the world? *Behavioral and Brain Sciences*, **33**, 61–135.

Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220.

Houle, M. and Nett, M. (2015). Rank-based similarity search: Reducing the dimensional dependence. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, **37**(1), 136–150.

Houtsma, A. J. M. (1995). Pitch perception. *Hearing*, **6**, 262.

Ishwaran, H. and Kogalur, U. B. (2010). Consistency of random survival forests. *Statistics & probability letters*, **80**(13), 1056–1064.

Jain, L., Jamieson, K. G., and Nowak, R. (2016). Finite sample prediction and recovery bounds for ordinal embedding. In *Advances in Neural Information Processing Systems (NIPS)*.

Jäkel, F. and Wichmann, F. A. (2006). Spatial four-alternative forced-choice method is the preferred psychophysical method for naive observers. *Journal of Vision*, **6**(11), 1307–1322.

Jamieson, K. G. and Nowak, R. D. (2011). Low-dimensional embedding using adaptively selected ordinal data. In *Annual Allerton Conference on Communication, Control, and Computing*.

Kaneshiro, B., Perreau Guimaraes, M., Kim, H.-S., Norcia, A. M., and Suppes, P. (2015). A Representational Similarity Analysis of the Dynamics of Object Processing Using Single-Trial EEG Classification. *PLoS One*, **10**(8), e0135697.

Karger, D. R. and Ruhl, M. (2002). Finding nearest neighbors in growth-restricted metrics. In *Symposium on the Theory of Computing (STOC)*, pages 741–750.

Kayaert, G., Biederman, I., Op de Beeck, H. P., and Vogels, R. (2005). Tuning for shape dimensions in macaque inferior temporal cortex. *European Journal of Neuroscience*, **22**(1), 212–224.

Kleindessner, M. and Luxburg, U. (2015). Dimensionality estimation without distances. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Kleindessner, M. and von Luxburg, U. (2014). Uniqueness of ordinal embedding. In *Conference on Learning Theory (COLT)*.

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

Kleiner, M., Brainard, D., Pelli, D., Ingling, A., Murray, R., and Broussard, C. (2007). Whats new in Psychtoolbox-3. *Perception*, **36**(14), 1.

Knoblauch, K. and Maloney, L. T. (2010). MLDS: Maximum likelihood difference scaling in R. *Journal of Statistical Software*, **25**, 1–26.

Knoblauch, K., Charrier, C., Cherifi, H., Yang, J., and Maloney, L. (1998). Difference scaling of image quality in compression-degraded images. *Perception ECVP abstract*, **27**.

Koenderink, J., Valsecchi, M., van Doorn, A., Wagemans, J., and Gegenfurtner, K. (2017). Eidolons: Novel stimuli for vision research. *Journal of Vision*, **17**(2), 7.

Krantz, D. H., Luce, R. D., Suppes, P., and Tversky, A. (2007). *Foundations of measurement: Geometrical, threshold, and probabilistic representations*, volume 1. Courier Corporation.

Krauthgamer, R. and Lee, J. (2004). Navigating nets: simple algorithms for proximity search. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, **29**(1), 1–27.

Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, **29**(2), 115–129.

Lawrence, J. (2013). *A catalog of special plane curves*. Courier Corporation.

Lecun, Y. and Cortes, C. (1998). The MNIST database of handwritten digits.

Li, L., Malave, V. L., Song, A., and Yu, A. (2016). Extracting human face similarity judgments: Pairs or triplets? In *CogSci*.

Liberti, L., Lavor, C., Maculan, N., and Mucherino, A. (2014). Euclidean distance geometry and applications. *Siam Review*, **56**(1), 3–69.

Lichman, M. (2013). UCI machine learning repository.

Lifshits, Y. and Zhang, S. (2009). Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Lin, C. H., Mausam, and Weld, D. S. (2014). To Re(label), or not to Re(label). In *Conference on Human Computation and Crowdsourcing (HCOMP)*.

Liu, T., Moore, A., Yang, K., and Gray, A. (2004). An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems (NIPS)*.

Luce, R. D. and Edwards, W. (1958). The derivation of subjective scales from just noticeable differences. *Psychological review*, **65**(4), 222.

Luukkainen, J. and Saksman, E. (1998). Every complete doubling metric space carries a doubling measure. *Proceedings of the American Mathematical Society*, **126**(2), 531–534.

Machado, J., Mata, M., and Lopes, A. (2015). Fractional State Space Analysis of Economic Systems. *Entropy*, **17**(8), 5402–5421.

Maloney, L. T. and Yang, J. N. (2003). Maximum likelihood difference scaling. *Journal of Vision*, **3**(8), 5.

Marder, J. and Fritz, M. (2015). The internet's hidden science factory. *PBS NewsHour*.

Marks, L. E. and Gescheider, G. A. (2002). Psychophysical scaling. *Stevens' handbook of experimental psychology*.

McNames, J. (2001). A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, **23**(9), 964–976.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, **63**(2), 81.

Norris, W. F. and Oliver, C. A. (1898). *System of Diseases of the Eye*, volume 3. JB Lippincott.

Op de Beeck, H., Wagemans, J., and Vogels, R. (2001). Inferotemporal neurons represent low-dimensional configurations of parameterized shapes. *Nature neuroscience*, **4**(12), 1244.

Ram, P. and Gray, A. (2013). Which space partitioning tree to use for search? In *Advances in Neural Information Processing Systems (NIPS)*.

Reed, S. K. (1972). Pattern recognition and categorization. *Cognitive psychology*, **3**(3), 382–407.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schultz, M. and Joachims, T. (2003). Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*.

Scornet, E. (2016). On the asymptotics of random forests. *Journal of Multivariate Analysis*, **146**, 72–83.

Scornet, E., Biau, G., and Vert, J.-P. (2015). Consistency of random forests. *The Annals of Statistics*, **43**(4), 1716–1741.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shepard, R. N. (1962). The analysis of proximities: multidimensional scaling with an unknown distance function. I. *Psychometrika*, **27**(2), 125–140.

Shepard, R. N. (1981). Psychological relations and psychophysical scales: On the status of direct psychophysical measurement. *Journal of Mathematical Psychology*, **24**(1), 21–57.

Shepard, R. N. (1982). Geometrical approximations to the structure of musical pitch. *Psychological review*, **89**(4), 305.

Shervashidze, N., Schweitzer, P., Leeuwen, E., Mehlhorn, K., and Borgwardt, K. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, **12**, 2539–2561.

Smith, C. A. and Ellsworth, P. C. (1985). Patterns of cognitive appraisal in emotion. *Journal of personality and social psychology*, **48**(4), 813.

Stevens, S. S. (1957). On the psychophysical law. *Psychological review*, **64**(3), 153.

Stewart, N., Brown, G. D., and Chater, N. (2005). Absolute identification by relative judgment. *Psychological review*, **112**(4), 881.

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

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

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological review*, **34**(4), 273.

Torgerson, W. S. (1958). *Theory and Methods of Scaling*. John Wiley and Sons, New York.

Tschopp, D., Diggavi, S., Delgosha, P., and Mohajer, S. (2011). Randomized algorithms for comparison-based search. In *Advances in Neural Information Processing Systems (NIPS)*.

Uhlmann, J. (1991). Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, **40**(4), 175–179.

Ukkonen, A., Derakhshan, B., and Heikinheimo, H. (2015). Crowdsourced non-parametric density estimation using relative distances. In *Conference on Human Computation and Crowdsourcing (HCOMP)*.

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

Wilber, M. J., Kwak, I. S., and Belongie, S. J. (2014). Cost-effective HITs for relative similarity comparisons. In *Conference on Human Computation and Crowdsourcing (HCOMP)*.