EPFL

Thèse n° 8224

Stochastic Models for Comparison-based Search

Présentée le 27 octobre 2023

Faculté informatique et communications
Laboratoire de la dynamique de l’information et des réseaux 1
Programme doctoral en informatique et communications

pour l’obtention du grade de Docteur ès Sciences

par

Daniyar CHUMBALOV

Acceptée sur proposition du jury

Prof. B. Faltings, président du jury
Prof. M. Grossglauser, directeur de thèse
Prof. C. Mascolo, rapporteuse
Prof. D. Figueiredo, rapporteur
Prof. R. West, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

“ "You miss 100% of the shots you don’t take.
— Wayne Gretzky"
— Michael Scott ”
— Daniyar Chumbalov

To my parents

Acknowledgements

I wish to thank my advisor, Matthias Grossglauser, for his mentorship. You have been an endless source of inspiration for me throughout these years and I have always truly admired your intuition and your level of mathematical culture.

I would like to thank the members of my jury committee: Prof. Robert West, Prof. Cecilia Mascolo, and Prof. Daniel Figueiredo. I greatly appreciated your feedback and comments on my thesis.

I wish to thank Andrei Romashchenko for introducing me to the world of scientific research back in Moscow when I was a second year undergraduate student.

It was a pleasure being a part of our INDY lab where I got to know and had a chance to work with clever people from whom I have learned a lot not only about math and computer science, but also about life.

I want to thank my close friends, Tsioma, Lev, Makar, Philipp, and especially Sanyok. You have always been there for me when I was going through my hard times and have been inspiring me for new achievements.

Finally, I want to thank my parents for their love, support and trust in me. This work is dedicated to you.

Lausanne, 25 September 2023 D. C.

Abstract

Every day, people search for information in databases helped by the convenience and efficiency of recent search-algorithms. These algorithms work extremely well in situations where the target search object is explicitly described in a query provided, as input, to a search engine. However, it is not always possible to formulate an explicit query. Searching on the Web for a specific image that was seen a few days before could be difficult; because describing it in terms of keywords might poorly match its annotation, assuming that the picture was annotated, which is also not guaranteed. This problem is addressed in a line of research called *searching with relevance feedback*, where the main idea is for the user to interact with the system in several rounds, until the target is found. In each round the user observes search results suggested by the system and provides his feedback on them. Based on this feedback, the search results are then refined and their updated version is presented to the user in the next round. This process repeats until the system outputs the user's search target. In this thesis, we study a special, and very natural for humans, form of such feedback, that is expressed through comparisons: "Which object among $(i_1, i_2, \dots, i_K)$ is more similar to your target t?". We derive new probabilistic comparison models that better capture users' choices, and we develop comparison-based search algorithms, based on these models, that offer *theoretical guarantees* and that are *efficient* in both computational complexity and in the number of questions posed to the user.

In particular, we study two new choice models. The first is a *Probit* model for triplet comparisons $(i, j; t)$, i.e. when the user chooses between two objects (i, j); this model is naturally based on the distance to the decision hyperplane between two alternatives i and j. The second model is a *scale-free* γ-CKL model that handles both pairs and K-tuples of alternatives; it is a generalization of the Crowd Kernel Learning (CKL) model proposed by Tamuz et. al. [47]. We show that both models are able to fit real-world comparison data better than the current state-of-the-art choice models, as the objects' embeddings learned by maximizing the corresponding log-likelihoods better predict human-generated triplets. We then describe how to learn a variational embedding of the objects and show that it benefits the search algorithm in the early stages of large uncertainty about relationships among objects, when only a few searches have been run and the system is short on comparison data.

Under the Probit model, we build GAUSSSEARCH – a Bayesian search algorithm with Gaussian belief for the target object's location. The special form of the Probit model enables us (1) to characterize the most informative query at each step of the search, which leads to a low query complexity, (2) to efficiently perform closed-form Bayesian updates of the posterior, and (3) to prove the convergence of the algorithm to the target in the "dense" case when $n \to \infty$. In a series of synthetic experiments, we show that GAUSSSEARCH has lower query complexity, similar to what is achieved by the state-of-the-art methods, but has much lower computational complexity.

Under the γ-CKL model, in the case when the number of objects n in the database is infinite, we use the scale-free property of the model to propose a search scheme with provably exponential rate of convergence to the target. This scheme uses a backtracking mechanism that keeps the search procedure from "losing the target". Then we propose a heuristic implementation of this scheme and through a number of experiments show its convergence to the target at an exponential rate in practice. Finally, when n is moderately small, we propose an efficient Bayesian heuristic γ-CKLSEARCH that at each step of the search approximately finds most informative queries to show to the user under the γ-CKL model.

To validate our proposed methods, we perform extensive experiments on real-world data. First, we demonstrate the advantages of using, in the cold-start regime, the proposed variational embedding over the point-vector embedding in a search algorithm. Second, we perform an experiment with real users in a task of searching for a movie actor; we show the efficacy of the GAUSSSEARCH algorithm when no prior comparison data is available. Finally, in another experiment with real users, we compare GAUSSSEARCH and γ-CKLSEARCH, and we confirm the superior performance of the latter.

Key words: comparisons, probabilistic modeling, interactive search, human computer interaction, recommender systems, machine learning

Résumé

Chaque jour, des personnes recherchent des informations dans des bases de données grâce à la commodité et à l'efficacité des récents algorithmes de recherche. Ces algorithmes fonctionnent très bien dans les situations où l'objet de la recherche est explicitement décrit dans une requête fournie en entrée à un moteur de recherche. Cependant, il n'est pas toujours possible de formuler une requête explicite. La recherche sur le web d'une image spécifique vue quelques jours auparavant peut s'avérer difficile, car sa description en termes de mots-clés peut mal correspondre à son annotation, en supposant que l'image ait été annotée, ce qui n'est pas non plus garanti. Ce problème est abordé dans une ligne de recherche appelée *recherche avec retour d'information sur la pertinence*, où l'idée principale est que l'utilisateur interagisse avec le système en plusieurs tours, jusqu'à ce que la cible soit trouvée. À chaque tour, l'utilisateur observe les résultats de recherche suggérés par le système et donne son avis sur ceux-ci. Sur la base de ces commentaires, les résultats de recherche sont ensuite affinés et leur version actualisée est présentée à l'utilisateur lors du tour suivant. Ce processus se répète jusqu'à ce que le système produise la cible de recherche de l'utilisateur. Dans cette thèse, nous étudions une forme particulière, et très naturelle pour les humains, de ce retour d'information, qui s'exprime par des comparaisons : "Quel objet parmi $(i_1, i_2, \ldots, i_K)$ est le plus similaire à votre cible t" ? Nous dérivons de nouveaux modèles probabilistes de comparaison qui rendent mieux compte des choix des utilisateurs, et nous développons des algorithmes de recherche basés sur les comparaisons, fondés sur ces modèles, qui offrent des *garanties théoriques* et qui sont *efficaces* en termes de complexité de calcul et de nombre de questions posées à l'utilisateur.

En particulier, nous étudions deux nouveaux modèles de choix. Le premier est un modèle *Probit* pour les comparaisons de triplets $(i, j; t)$, c'est-à-dire lorsque l'utilisateur choisit entre deux objets (i, j) ; ce modèle est naturellement basé sur la distance à l'hyperplan de décision entre deux alternatives i et j. Le second modèle est un modèle *scale-free* γ-CKL qui traite à la fois les paires et les K-tuples d'alternatives; il s'agit d'une généralisation du modèle Crowd Kernel Learning (CKL) proposé par Tamuz et. al. [47]. Nous montrons que les deux modèles sont capables de s'adapter aux données de comparaison du monde réel mieux que les modèles de choix actuels de pointe, car les enchâssements d'objets appris en maximisant les log-vraisemblances correspondantes prédisent mieux les triplets générés par l'homme. Nous décrivons ensuite comment apprendre un encastrement variationnel des objets et montrons que cela profite à l'algorithme de recherche dans les premiers stades d'une grande incertitude sur les relations entre les objets, lorsque seules quelques recherches ont été effectuées et que le système manque de données de comparaison.

Dans le cadre du modèle Probit, nous construisons GAUSSSEARCH – un algorithme de recherche bayésien avec une croyance gaussienne pour la localisation de l'objet cible. La forme spéciale du modèle Probit nous permet (1) de caractériser la requête la plus informative à chaque étape de la recherche, ce qui conduit à une faible complexité de la requête, (2) d'effectuer efficacement des mises à jour bayésiennes de forme fermée du postérieur, et (3) de prouver la convergence de l'algorithme vers la cible dans le cas "dense" lorsque $n \to \infty$. Dans une série d'expériences synthétiques, nous montrons que GAUSSSEARCH a une complexité d'interrogation plus faible, similaire à celle obtenue par les méthodes de pointe, mais a une complexité de calcul beaucoup plus faible.

Sous le modèle γ-CKL, dans le cas où le nombre d'objets n dans la base de données est infini, nous utilisons la propriété scale-free du modèle pour proposer un schéma de recherche avec un taux de convergence exponentiel vers la cible. Ce schéma utilise un mécanisme de retour en arrière qui empêche la procédure de recherche de "perdre la cible". Nous proposons ensuite une implémentation heuristique de ce schéma et, à travers un certain nombre d'expériences, nous montrons sa convergence vers la cible à un taux exponentiel dans la pratique. Enfin, lorsque n est modérément petit, nous proposons une heuristique bayésienne efficace γ-CKLSEARCH qui, à chaque étape de la recherche, trouve approximativement les requêtes les plus informatives à montrer à l'utilisateur selon le modèle γ-CKL.

Pour valider les méthodes proposées, nous réalisons des expériences approfondies sur des données réelles. Tout d'abord, nous démontrons les avantages de l'utilisation, dans le régime de démarrage à froid, de l'intégration variationnelle proposée par rapport à l'intégration de vecteurs ponctuels dans un algorithme de recherche. Deuxièmement, nous réalisons une expérience avec des utilisateurs réels dans une tâche de recherche d'un acteur de cinéma ; nous montrons l'efficacité de l'algorithme GAUSSSEARCH lorsqu'aucune donnée de comparaison préalable n'est disponible. Enfin, dans une autre expérience avec des utilisateurs réels, nous comparons GAUSSSEARCH et γ-CKLSEARCH, et nous confirmons la performance supérieure de ce dernier.

Mots clefs : comparaisons, modélisation probabiliste, recherche interactive, interaction homme-machine, systèmes de recommandation, apprentissage automatique

Contents

Acknowledgements **i**

Abstract / Résumé **iii**

Notation Table **ix**

1 Introduction **1**
- 1.1 Searching with Relevance Feedback 1
- 1.2 Probabilistic Comparison Models 2
- 1.3 Latent Setting and LEARN2SEARCH 4
- 1.4 Related Work 5
  - 1.4.1 Learning Embedding from Comparisons 5
  - 1.4.2 Comparison-based Search with Known Features 6
  - 1.4.3 Comparison-based Search with Hidden Features 8
- 1.5 Outline 8

2 Triplet Comparison Models **9**
- 2.1 Formal Definition and Examples of the Models 9
- 2.2 New Models 13
  - 2.2.1 Probit Model 13
  - 2.2.2 γ-CKL Model 15
- 2.3 Proofs 20
  - 2.3.1 Theorem 1 20
  - 2.3.2 Proposition 1 24

3 Technical Background for Bayesian Inference **27**
- 3.1 Moment-Matching for Probit Model 28
- 3.2 Variational Inference 32

4 Learning Object Embeddings from Triplet Comparisons **35**
- 4.1 Learning Embedding via Maximum Likelihood Estimator 35
- 4.2 Models Performance 36

4.3 Estimating d via SVD 41
4.4 Embedding Visualization 42
4.5 Variational Embedding 44

5 Comparison-based Search 47
5.1 Probit Model: GAUSSSEARCH 48
  5.1.1 Choosing the Next Query 48
  5.1.2 Updating the Model 50
  5.1.3 Convergence of GAUSSSEARCH 52
  5.1.4 Experimental Results: Non-latent Setting 53
  5.1.5 Experimental Results: Search in Latent Setting 56
5.2 γ-CKL model: Exponential Convergence and γ-CKLSearch 60
  5.2.1 Exponential Convergence: a Scheme with Backtracking 60
  5.2.2 Implementation of the Scheme with Backtracking 71
  5.2.3 γ-CKLSEARCH for Moderate n 73
5.3 Proofs 75
  5.3.1 Theorem 2 75
  5.3.2 Theorem 3 76
  5.3.3 Lemma 2, Lemma 3, and Theorem 4 80
  5.3.4 Lemma 4 and Theorem 5 84

6 Experiments with Human Oracles 87
6.1 LEARN2SEARCH with Probit Model 90
6.2 γ-CKLSEARCH vs GAUSSSEARCH 92

7 Conclusion 95

Bibliography 102

Curriculum Vitae 103

Notation Table

i, j, k	generic object indices
Q	set of object indices, query
n	total number of objects in the database
$[n]$	objects $\{1, 2, \ldots, n\}$
d	embedding dimension
m	step of the algorithm
t	target index
$\boldsymbol{x}_i$	embedding of object i
$\boldsymbol{x}_t$	target embedding
$\mathcal{X}, \boldsymbol{X}$	embedding of the objects $[n]$
(y, Q) or $(\boldsymbol{x}, Q)$	choice data
$p(i, Q; t)$ or $p(\boldsymbol{x}_i, \mathcal{X}_Q; \boldsymbol{x}_t)$	probability of the k-triplet outcome "i is the closest to t among alternatives Q"
$p(i, j; t)$ or $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$	probability of the triplet outcome "i is closer to t than j"
$\{i, j\}$ or $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$	a query of size 2

1 Introduction

1.1 Searching with Relevance Feedback

Over the past few decades, search engines have become the go-to tool for seeking information. Modern information-retrieval techniques very well identify relevant objects to the query entered by a user, especially when the search query describes sufficiently precisely the subject of the search. Indeed, by typing partial lyrics of a song into Google or by searching for a specific question on Stack Exchange, a user can immediately find the desired music track or the answer to the question, because these types of queries explicitly match the search target. However, in many other situations, such queries could be difficult to express: searching for a new co-worker to add on a social-media platform when you remember only his face can be problematic, as most people would struggle to draw the face of a person accurately enough that it could be used as a query image. Moreover, even if the user could provide the initial search query, the output of the search engine most likely would be not sufficient to complete the search. The next thing the search engine could do is to request additional relevance feedback from the user to the engine's initial output in order to refine the set of potential search target candidates and then reiterate on this, until the target is eventually identified. An example of such process in real life can be well illustrated by the joint work of a forensic artist and a witness of a crime, where the latter has to give feedback on a sequence of images to gradually arrive at a faithful approximation of the suspect's face.

In the scope of a search task in a database of objects with relevance feedback, the user interacts with the search system in multiple rounds. The form of such relevance feedback plays a fundamental role in designing the underlying search algorithm. As suggested in the psychological literature (see e.g. [45]), it is usually cognitively easier for humans to express their relevance feedback as a preference choice between two objects, rather than to give a numerical score when assessing the "proximity" of the candidates suggested by the system to the user's actual target. In this dissertation, we study search systems that interact with a user via questions in the following form: “Which object among $\mathcal{Q}$ is the most similar to your target t?”, where $\mathcal{I}$ is a subset of candidate objects from the database chosen at each round by the system and t is the user’s target object. The choice made by the user thus take the form of *tuple* comparisons.

The purpose of this thesis is to propose an efficient way of searching using comparisons in a database of objects. The key decision to be made is how to model humans choices. For this, we present two new probabilistic choice models. For each model, we develop search algorithms that are *scalable* in the size of the database of objects and *efficient* in both computational complexity and in the number of interaction rounds.

1.2 Probabilistic Comparison Models

The concept of perceived similarity between different objects has been a long-studied topic in the psychological literature of the past century. Usually, it is assumed that the objects have some representations in a latent feature space Ω, and that when a person compares two objects, his choice is viewed as an outcome of a perceptual judgement on the similarity between objects based on these representations. The latent space is usually considered to be low-dimensional, even though the raw objects might be high-dimensional (such as images, videos, and sequences of musical notes). For example, a human face, for the purposes of a similarity comparison, could be quite accurately described by a few tens of features, capturing head shape, eye color, fullness of lips, gender, age, and the like [11]. The similarity $s(i, j)$ between two objects i and j is then defined as a function $f(\boldsymbol{x}_i, \boldsymbol{x}_j)$ of the objects representations $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$.

One of the most influential psychological theories assumes that the latent space Ω is a metric space with a distance function $d(\cdot,\cdot)$ and that the perceived similarity between two objects is inversely related to the distance between these objects in Ω ([44, 4]):

$$s(i, j) = g[d(\boldsymbol{x}_i, \boldsymbol{x}_j)],$$

where g is some monotonically non-increasing function. This theory gave rise to a powerful class of methods called multi-dimensional scaling (MDS) that produces latent embeddings of the objects from similarity estimations. The MDS was initially developed to deal with explicit numerical scores for pairwise similarities [49, 7] and, later, it was generalized to non-metric MDS that can handle ordinal data [32, 43, 1]. Despite the intuitiveness of the model and its practical appeal, the original idea of using distance as a measure of (dis)similarity had some drawbacks, and its main criticism was in the violations of some of the metric axioms in practice [51, 52]. Tversky in [51] empirically demonstrated a violation of some of the distance axioms through a number of experiments. He proposed, as an alternative, to represent an object i as a set of qualitative features X_i (rather than a continuous point in a metric space), and he designed a feature-contrast similarity model based on set operations:

$$s(i,j) = F(X_i \cap X_j, X_i \setminus X_j, X_j \setminus X_i).$$

Nevertheless, probabilistic models for human choices, that are based on the distance-based similarity measures [42, 34, 37] and were developed to account for the noise in the judgements, have been the most successful in predictive tasks [4] and have become dominant in modern machine learning research.

In this thesis, we follow the probabilistic view on the comparison outcomes; we assume that the decision outcomes are sampled from a probability distribution that depends on the object representations in a latent metric space. We assume, in particular, that the objects from the database $[n] = \{1, 2, \ldots, n\}$ have associated latent feature vectors $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ in $\Omega = \mathbb{R}^d$ that reflect their individual properties and that are intuitively used by users when answering the system's queries. We define $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ to be the feature matrix of the objects in $[n]$, where the rows of $\boldsymbol{X}$ are the corresponding feature vectors of the objects. We use the Euclidean distance $\|\cdot\|$ between the objects' vectors in $\mathbb{R}^d$ in order to quantify similarities between them:

$$\text{“}i \text{ is the most similar object in } Q \text{ to } t\text{”} \quad \Longleftrightarrow \quad i = \arg\min_{j \in Q} \|\boldsymbol{x}_j - \boldsymbol{x}_t\|.$$

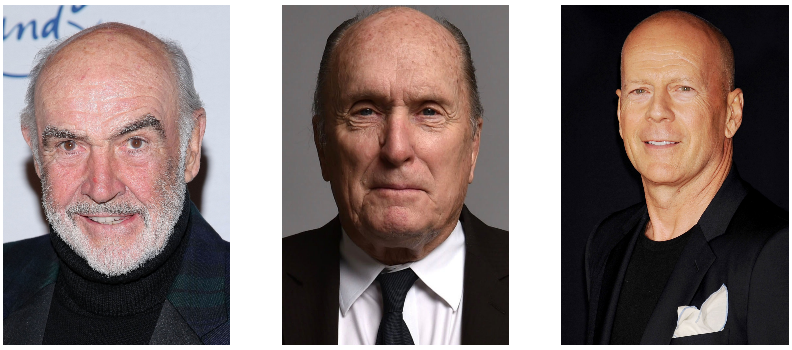

Actor 1 Actor 2 Target actor

Figure 1.1: An example of inconsistency across user answers: 60% of respondents chose Actor 1 as more similar to the Target actor than Actor 2 [14].

As different users might have different perceptions of similarity and, even for a single user, it might be difficult to answer every comparison confidently, we expect to observe inconsistencies across the collected replies (see Fig. 1.1).

Therefore, we need to introduce an appropriate probabilistic model to reflect the noise in comparison outcomes.

Formally, by asking a user "Which object in Q is the most similar to your target t?", we ask him a query Q that consists of a tuple of objects he has to choose from, based on the perceived proximity to his target t. When the user makes his choice, he gives us his answer $Y \in Q$. As previously mentioned, there could be an inconsistency in human similarity judgements across different individuals, and even the same user can change his opinion on his similarity choice for the same query due to factors that are outside of our control, hence we treat Y as a random variable. We denote the probability of receiving the outcome $Y = i$, i.e., observing the answer "i is the most similar object in Q to t", by

$$p(i, Q; t) := P(Y = i \mid Q, t). \tag{1.1}$$

This probability usually depends on the relationships between the the objects in the feature space $\mathbb{R}^d$. Hence, during the search procedure, a search algorithm relies on the latent features of the objects and on the probabilistic answer model when observing choice feedback from the user.

1.3 Latent Setting and LEARN2SEARCH

In the previous subsection, we make one important assumption – when the user compares objects from $[n]$, he utilizes their latent representations $\boldsymbol{X}$ when answering queries. We have discussed that in order to make progress, a search algorithm relies on probabilistic comparison models to compute probabilities of the observed query outcomes that the user provides. However, the algorithm can compute these probabilities only if it has access to these latent features $\boldsymbol{X}$. But in a real-world application, that involves using triplet models, it is quite possible that $\boldsymbol{X}$ will be hidden from the system, and possible workarounds, such as applying unsupervised learning methods to approximate $\boldsymbol{X}$ (for example, some pre-trained Convolutional Neural Networks to retrieve human facial features), could lead to representations that are arbitrarily inaccurate to compute triplet comparison probabilities on. We refer to this scenario as a *latent setting*, because the true feature vectors of the objects are hidden from the system and are available only to the user.

We address the problem of searching in a space of objects with unobserved features by proposing a general learning framework called LEARN2SEARCH. In this setting we assume that the only information we have access to is a set of indices $[n]$, representing objects, and results of triplet comparisons of objects in the form of triplets of indices $(i, j; t)$. Our framework operates with its own object representations that are used during the searches and that are learned and updated from the triplet comparisons obtained from past searches. LEARN2SEARCH consists of two main components: a search method (running on the estimated *embedding* $\hat{\boldsymbol{X}}$ of objects) and an embedding method (that learns and updates $\hat{\boldsymbol{X}}$), see the general scheme in Fig. 1.2.

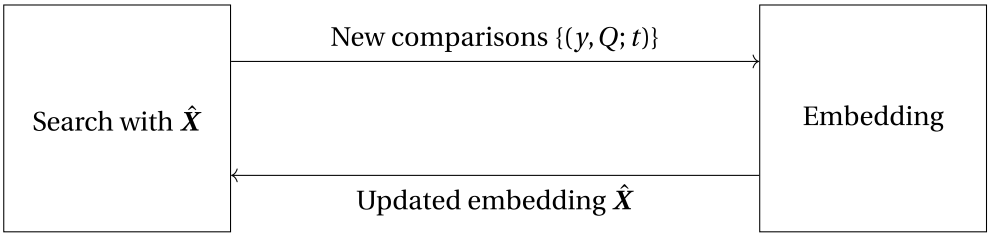

New comparisons $\{(y, Q; t)\}$
Search with $\hat{\boldsymbol{X}}$
Embedding
Updated embedding $\hat{\boldsymbol{X}}$

Figure 1.2: A general search framework LEARN2SEARCH.

In Chapters 5 and 6, we will experimentally demonstrate the efficacy of LEARN2SEARCH in the latent setting: The quality of the embedding $\hat{\boldsymbol{X}}$ improves as more searches complete, which in turn makes the future searches more efficient.

1.4 Related Work

1.4.1 Learning Embedding from Comparisons

The problem of estimating $\boldsymbol{X}$ from a set of triplet comparisons $(i, j; t)$ has been studied extensively in recent years. Jamieson and Nowak [27] study a problem of identification of the objects features matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ from a set of *noiseless* triplet comparisons between n objects in $\mathbb{R}^d$ via comparing their relative distances. They provide a lower bound $\Omega(dn\log n)$ on the number of triplets needed to uniquely determine the embedding $\boldsymbol{X}$ that satisfies all $O(n^3)$ possible relational triplet constraints and describe an adaptive scheme that builds an embedding by sequentially selecting triplets of objects to compare. In our work we assume noise in the triplet comparisons and cannot choose triplets to ask adaptively. The authors also do not discuss how to construct an embedding from a given *fixed* set $\mathcal{T}$ of similarity triplets, which is an essential problem arises in the latent setting.

More general algorithms for finding $\hat{\boldsymbol{X}}$ from a fixed set of triplet comparisons, under different triplet probability models, i.e. allowing noise in the triplet outcomes, are proposed in [1], [47] and [54], and involve solving an optimization problem of the form

$$\sum_{(i,j,;t)\in\mathcal{T}} f(\boldsymbol{X}, (i, j; t)) \to \min_{\boldsymbol{X}\in\mathbb{R}^{n\times d}}$$

subject to some constrains which is usually solved using gradient descent method. We discuss this approach in details in Chapter 4. Amid and Ukkonen [2] modify the formulation of the joint t-STE likelihood [54] to solve a problem of identification of multiple low-dimensional embeddings $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_d\}$, $\boldsymbol{X}_m \in \mathbb{R}^{n\times d_i}$, at once, each corresponding to a single latent feature $m \in \{1, 2, \ldots, d\}$ of an object, that are used by the oracle when answering triplet comparison queries. In particular, they assume that observing a triplet $(i, j; t)$ implies that the inequality

$$\|x_i^m - x_t^m\|^2 < \|x_j^m - x_t^m\|^2$$

is satisfied in $\boldsymbol{X}_m$ for some (possibly multiple) $m \in \{1, 2, \ldots, d\}$. In this work we assume a single latent embedding of the objects $\boldsymbol{X} \in \mathbb{R}^d$. Finally, theoretical properties for a general MLE with an assumed general probabilistic generative data model are studied by Jain et al. [26].

An alternative approach to learning ordinal data embedding is suggested in [31], where the authors explicitly construct kernel functions $k : [n] \times [n] \to \mathbb{R}$ to measure similarities between pairs of objects based on the set of triplet comparisons:

$$k(i, t) < k(j, t) \iff (i, j; t) \in \mathcal{T}.$$

This, e.g., allows to perform clustering on the set of objects by observing only triplet comparisons, however the authors do not explicitly assume the existence of an underlying feature space for the objects. Heim et al. [24] adapt the kernel version of [54] for an online setting, when the set of observable triplets is expanding over time. The authors use stochastic gradient descent to learn the kernel matrix $K = (k(i, j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$ and to exploit sparsity properties of the gradient. Although this work is closely related to our latent setting, i.e. periodically updating the embedding matrix with the new data collected from the previous searches as proposed in LEARN2SEARCH algorithm, the kernel decomposition, which is $O(n^3)$ in time and has to be performed after each completed search, would be too computationally expensive in an online real world application.

Finally, in [21] and [3] authors propose landmark-based triplet embedding methods that scale well for large values of n but assume direct access to the oracle in order to ask to compare adaptively selected triplets. Our scenario in the latent setting is different, as we only passively collect triplets from past searches.

1.4.2 Comparison-based Search with Known Features

In the general *active learning* problem, it is usually assumed that there exists a set of binary hypotheses $\mathcal{H}$ and a set of queries $\mathcal{X}$, where each hypothesis can be tested via a query from $\mathcal{X}$, i.e. by observing the hypotheses outcome. The goal is then to identify a target hypothesis by making queries which helps to shrink the set of possible target candidate hypotheses. The Generalized Binary Search (GBS) method [17, 38] is known to be near-optimal in the number of queries, if the observed outcomes are noiseless. The main idea of the GBS is to greedily find a query that shrinks the current set of candidate hypotheses the most.

In the noisy case, Nowak [39] and Golovin et al. [22] propose objectives that are greedily minimized over all possible queries and outcomes in a Bayesian framework with a full posterior distribution over the hypotheses. We adapt the method suggested in [22] to our setting in Chapter 5 and investigate its performance in comparison to our proposed search algorithm GAUSSSEARCH for our Probit oracle model.

Adaptive search through relevance feedback by using preprocessed features was studied in the context of image retrieval systems in [16, 19, 20, 46]. In these papers, the authors aim at building a heuristic system that assists users in finding target images in databases. In each round of a search, the system displays a set of images if size K and the user is asked to pick the one that is the closest to his desired target image. The system is based on a Bayesian framework with a full posterior on the database images and the next query (displayed set of images) is chosen via maximizing some information-theoretic criterion assuming users choices are drawn from a given probabilistic oracle model with $K > 2$ alternatives. In this work we mainly focus on the setup when the user needs to choose between two alternatives, and are interested in algorithms with certain theoretical guarantees.

Brochu et al. [9] study the comparison-based search problem in continuous space. Their approach relies on maximizing a surrogate GP model of the valuation function. In [13] and [25] the authors study a similar problem where the entire valuation function needs to be estimated. Comparison-based search under noise is also explored in [10]; the authors propose a Bayesian scheme that finds the next query via greedily maximizing the information gain using sampling from the posterior. In their work they independently proposed one of the oracle model that also depends on the distance from the target point $\boldsymbol{x}_t$ to the bisecting hyperplane defined by the query $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The main difference between that model and the Probit model proposed in this work is the activation function: the authors use the logistic function $f(x) = 1/(1+e^{-x})$ instead of the Gaussian CDF function, see Chapter 2.

The problem of the comparison-based search is closely related to the Reinforcement Learning (RL) field. Indeed, during a search, one can view a set of choices made by the user so far as a state, a set of all possible queries (which is of size $\binom{n}{k}$ for a query of size k) as a set of actions and somehow define a reward based on how close the procedure gets to the target point. The main difference is that in this work we model the target position as a distribution on $[n]$ or in $\mathbb{R}^d$, and the choice of the next query is usually based on the greedy maximization of some information-theoretic criterion. In contrast to maximizing cumulative reward, which is usually the objective in the RL literature, we seek for an action with an immediate reward that balances both exploration of the space and exploitation of the current belief.

1.4.3 Comparison-based Search with Hidden Features

When the objects' features $\mathcal{X}$ are hidden, finding a target in a set of objects by making similarity queries is explored in [50] and [28] in the noiseless case. To deal with erroneous answers, Kazemi et al. [29] consider an augmented user model that allows a third outcome to the query, "?"; this answer can be chosen by a user to indicate that they do not know the answer confidently. With this model, they propose several new search techniques under different assumptions on the knowledge of relative object distances. Karbasi et. al. [28] also briefly discussed the case of a noisy oracle with a constant mistake probability and proposed to treat it with repeated queries. In [23] the authors consider the problem of nearest-neighbor search using comparisons in the noiseless setting.

1.5 Outline

In short, the outline of this thesis is as follows:

- Chapter 2 introduces two new comparison models, Probit and γ-CKL, and studies some of their theoretical properties.
- Chapter 3 discusses the general problem of Bayesian Inference and the approximate methods for solving it.
- Chapter 4 describes methods to learn object embedding from comparison data and compares the qualities of the learned embeddings using new models versus using state-of-the-art alternatives.
- Chapter 5 introduces two efficient search algorithms that are based on the Probit and the γ-CKL model respectively, and gives certain theoretical guarantees for these algorithms exploiting the properties of these models.
- Chapter 6 presents the results of the experiments with proposed search methods and models involving human oracles.
- Chapter 7 summarizes the results and findings of the thesis.

2 Triplet Comparison Models

In this Chapter, we first give a formal definition of a probabilistic comparison model, define their properties and explore some well-known models existing in the literature in light of these properties.

Then we introduce two new probabilistic comparison models. The first is a purely triplet model based on the decision hyperplane between two query points. The second is a generalization of the CKL model that preserves the scale-invariance property and helps avoid the curse of dimensionality the original model is suffering from.

2.1 Formal Definition and Examples of the Models

Following the discussion in the previous Chapter, we denote the probability of the choice "i is the most similar object in Q to t" by

$$p(i,Q;t) := P(Y = i \mid Q,t), \tag{2.1}$$

where i, t, and Q are object indices $[n]$.

Generally, for an arbitrary (finite) set of vectors $Z \subset \mathbb{R}^d$ and a target vector $\boldsymbol{x}_t \in \mathbb{R}^d$, we could also define the probability of vector $\boldsymbol{x} \in Z$ to be chosen as the closest in Z to $\boldsymbol{x}_t$ as

$$p(\boldsymbol{x},Z;\boldsymbol{x}_t) := P(Y = \boldsymbol{x} \mid Z,\boldsymbol{x}_t). \tag{2.2}$$

Since we assume that the user judgements are based on the hidden representations $\mathcal{X}$ of the objects, we will sometimes equivalently write (2.1) as

$$p(i,Q;t) = p(\boldsymbol{x}_i, \mathcal{X}_Q; \boldsymbol{x}_t), \quad \text{where} \tag{2.3}$$

$$\mathcal{X}_Q := \{\boldsymbol{x}_j \in \mathcal{X} \mid j \in Q\}. \tag{2.4}$$

When $|Q| = k$ we call (2.1) and (2.3) a *k-triplet model*. The minimum value of $k = 2$ corresponds to a special family of models called *triplet models*:

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) := P(Y = \boldsymbol{x}_i \mid Q = \{\boldsymbol{x}_i, \boldsymbol{x}_j\}, \boldsymbol{x}_t), \tag{2.5}$$

which will be the central type of models studied in this thesis. Generally, a triplet model (2.5) is required to satisfy at least the basic property of a *correct bias*: We expect the outcome "i is more similar to t than j", which we will further denote by $(i,j;t)$ or $(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$, to be the most probable if and only if the same is true in the latent representations $\mathcal{X}$

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) > \frac{1}{2} \iff \|\boldsymbol{x}_i - \boldsymbol{x}_t\| < \|\boldsymbol{x}_j - \boldsymbol{x}_t\|. \tag{2.6}$$

We define and explore the following properties of a triplet model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$:

(P1) **k-triplet generalization,** whether the triplet model can be naturally generalized to the case of more than two alternatives, i.e., $|Q| = k$, $k > 2$.

(P2) **Identical-query-point property,** whether the probability of observing the outcome $(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ is 1 when $\boldsymbol{x}_i = \boldsymbol{x}_t$ and $\boldsymbol{x}_j \neq \boldsymbol{x}_t$, i.e., $p(\boldsymbol{x}_t, \boldsymbol{x}_j; \boldsymbol{x}_t) = 1$.

(P3) **Distant-target behavior**, given query points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and the target $\boldsymbol{x}_t$: the query points positions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are fixed and the target point is moving away from them, i.e. $\lim_{m\to\infty} p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_{t_m})$ for $\boldsymbol{x}_{t_0} = \boldsymbol{x}_t$, $\|\boldsymbol{x}_i - \boldsymbol{x}_{t_m}\| \to \infty$ and $\|\boldsymbol{x}_j - \boldsymbol{x}_{t_m}\| \to \infty$ as $m \to \infty$, and $\|\boldsymbol{x}_i - \boldsymbol{x}_{t_m}\| \neq \|\boldsymbol{x}_j - \boldsymbol{x}_{t_m}\|$ for all $m \geq 0$.

(P4) **Narrow-query behavior**, given query points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and the target $\boldsymbol{x}_t$: the target $\boldsymbol{x}_t$ is fixed and the query points approaching their middle point, i.e., $\lim_{m\to\infty} p(\boldsymbol{x}_{i_m}, \boldsymbol{x}_{j_m}; \boldsymbol{x}_t)$ for $\boldsymbol{x}_{i_0} = \boldsymbol{x}_i$, $\boldsymbol{x}_{j_0} = \boldsymbol{x}_j$, $\bar{\boldsymbol{x}} := (\boldsymbol{x}_i + \boldsymbol{x}_j)/2$ and $\|\boldsymbol{x}_{i_m} - \bar{\boldsymbol{x}}\| \to 0$ and $\|\boldsymbol{x}_{j_m} - \bar{\boldsymbol{x}}\| \to 0$ as $m \to \infty$.

(P5) **Scale-free property**, whether the outcome probabilities remain unaltered if the objects features are scaled by a positive multiplicative constant, $p(c\cdot\boldsymbol{x}_i, c\cdot\boldsymbol{x}_j; c\cdot\boldsymbol{x}_t) = p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ for $0 < c \leq 1$.

Let us now introduce the two most popular triplet models, CKL and t-STE, and study them in light of properties above.

Crowd Kernel Learning Model. Tamuz et al. [47] defined the probability of the outcome "i is closer to t than j" by

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2}{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2 + \|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2}. \tag{2.7}$$

In their model, the probability of the outcome $(i, j; t)$ monotonically depends on the ratio of the distances $r = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2}{\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2}$ between the alternatives i and j and the target t:

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \frac{1}{1 + r}. \tag{2.8}$$

By fixing the distance $\|\boldsymbol{x}_i - \boldsymbol{x}_t\|$ and increasing $\|\boldsymbol{x}_j - \boldsymbol{x}_t\|$, i.e., making $r \to 0$, the probability of observing the outcome $(i, j; t)$ goes to 1, and vice versa, when we fix $\|\boldsymbol{x}_j - \boldsymbol{x}_t\|$ and increase $\|\boldsymbol{x}_i - \boldsymbol{x}_t\|$, i.e., when $r \to \infty$, the probability of choosing j over i with respect to t goes to 0. When one of the alternatives exactly match the target, the probability of choosing the closest object also becomes 1, hence the model satisfies the identical query point property, (P2). It also has a straightforward k-triplet generalization (P1):

$$p(\boldsymbol{x}_i, Q; \boldsymbol{x}_t) = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2}{\sum_{j \in Q} \|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2}.$$

For the property of the distant-target behavior (P3), the CKL model outputs $\frac{1}{2}$, i.e., the predicted outcomes are the perfect coin flips, when the target is infinitely far away from both query objects. The same occurs when the query is narrow (P4), $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) \to \frac{1}{2}$, as the query points become indistinguishable as they approach their middle point.

The central feature of this model is that it is scale-free (P5), because multiplying $\boldsymbol{X}$ by any positive constant does not change the probability $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$. Such a property is, in fact, desirable for a comparison model, as it reflects some of the psychological laws in perception [12, 33]. Later in this Chapter, we will discuss a set of axioms needed for scale-free models and will derive the general form of (2.7).

Stochastic Triplet Embedding Model. In [54] the authors proposed an alternative model based on the *logistic* function in the attempt to address some of the flaws of the CKL model when learning the object's embedding based on a set of triplets (we will discuss the problem of triplet embedding in Chapter 4):

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2)}{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2) + \exp(-\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2)}, \tag{2.9}$$

with a generalization for a k-triplet scenario

$$p(\boldsymbol{x}_i, Q; \boldsymbol{x}_t) = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2)}{\sum_{j\in Q}\exp(-\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2)}.$$

Inspired by the well-known algorithm for dimensionality reduction t-SNE [53], which uses heavy-tailed similarity kernels, they also introduced a t-distributed version of the original STE model:

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \frac{\left(1 + \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\left(1 + \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}} + \left(1 + \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}, \tag{2.10}$$

with α degrees of freedom, which is a hyperparameter. This model with heavy tails showed better performance in the experiments on modeling human generated triplets in their paper.

The STE models do not carry the identical query point property (P2): When the target is exactly at one of the query points, $\boldsymbol{x}_i = \boldsymbol{x}_t$, the probability $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) \neq 1$. The edge cases $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = 1$ or $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = 0$ are achieved only when exactly one of the query points is infinitely far from the target, i.e., when $\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^2 = \infty$ and $\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^2 < \infty$ or vice versa (P3). When $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \to 0$ and $\bar{\boldsymbol{x}} = (\boldsymbol{x}_i + \boldsymbol{x}_j)/2$ is fixed, the different between the distances $(\|\boldsymbol{x}_i - \boldsymbol{x}_t\| - \|\boldsymbol{x}_j - \boldsymbol{x}_t\|) \to 0$, and $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) \to 1/2$, (P4).

Models Visualization

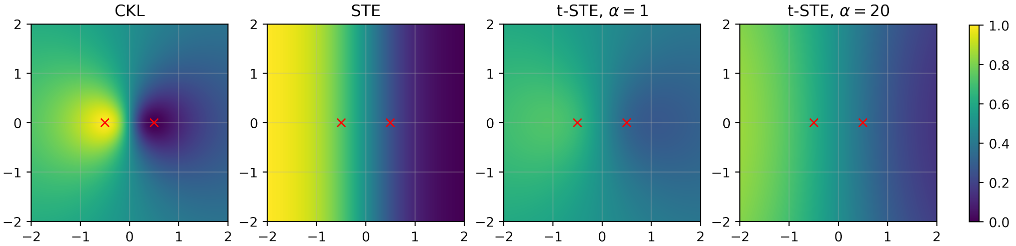


Figure 2.1: Likelihood $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ heatmaps in $\mathbb{R}^2$ for CKL and t-TSE. The red crosses represent the query points $\boldsymbol{x}_i = (-0.5, 0)$, $\boldsymbol{x}_j = (0.5, 0)$.

We examine the behavior of the likelihood functions of the CKL, STE and t-STE models in Figure 2.1. We fix a query $Q = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, where $\boldsymbol{x}_1 = (0, -0.5)$ and $\boldsymbol{x}_2 = (0, 0.5)$ and plot the likelihood $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ as a function of $\boldsymbol{x}_t \in \mathbb{R}^2$. The surface of the CKL likelihood has exactly one "peak" and one "valley", at $\boldsymbol{x}_t = \boldsymbol{x}_1$ and $\boldsymbol{x}_t = \boldsymbol{x}_2$, respectively. As we move away from these two spots (in any direction), the probability $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ is approaching $\frac{1}{2}$, which is desired: If the target point $\boldsymbol{x}_t$ is far away from both query points, we would expect to observe each outcome with approximately the same probability. The STE likelihood function has a very pronounced decision hyperplane between the two query points; moving away from the hyperplane results into either $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = 1$ or $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = 0$ in the limit. The t-STE has a fatter-tailed distribution with a smoother transition to the limit cases. Although, it might appear from the plot that the STE probability depends only on the hyperplane, this is not true; in general, for two points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ in $\mathbb{R}^d$, which are equidistant from the hyperplane and lie on the same side of it, the corresponding probabilities are not the same, $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{z}_1) \neq p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{z}_2)$. In the next subsection, we will introduce a *Probit* triplet model, for which the outcome probabilities depend exclusively on the bisecting hyperplane between the two query points.

2.2 New Models

2.2.1 Probit Model

As previously discussed, users can be inconsistent in their judgments on the triplet similarities between two objects, with respect to the third one. Under the *Probit model*, we postulate that this occurs when both query objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are roughly equidistant from the target object $\boldsymbol{x}_t$, and less likely when the distances are quite different. In other words, answers are most noisy when the target is close to the decision boundary, i.e., the bisecting normal hyperplane to the segment between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and the answers are less noisy when the target is far away from this hyperplane.

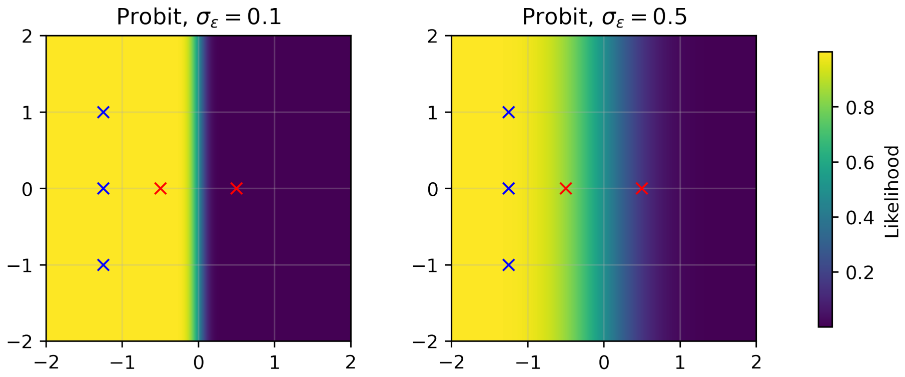


Figure 2.2: Probit model's likelihood heatmaps in $\mathbb{R}^2$ with different values of σ_ε. The red crosses represent the query points $\boldsymbol{x}_i = (-0.5, 0)$, $\boldsymbol{x}_j = (0.5, 0)$. The outcome probabilities are defined by the decision hyperplane between the query points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The three blue crosses with coordinates $\boldsymbol{z}_1 = (-1.25, -1)$, $\boldsymbol{z}_2 = (-1.25, 0)$, and $\boldsymbol{z}_3 = (-1.25, 1)$, are equidistant from the hyperplane and lie on the same side of it. The probabilities of the outcomes $(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{z}_k)$ are the same for $k = 1, 2, 3$.

We consider the following probabilistic model, called the *Probit model*; it captures the uncertainty in users' answers:

$$\begin{aligned} p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) &= p(\boldsymbol{x}_t^\top \boldsymbol{w}_{ij} + b_{ij} + \varepsilon > 0) \\ &= \Phi\left(\frac{\boldsymbol{x}_t^\top \boldsymbol{w}_{ij} + b_{ij}}{\sigma_\varepsilon}\right), \end{aligned} \tag{2.11}$$

where $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_t \in \mathbb{R}^d$ are the feature vectors of the query objects and the target, respectively,

$$h_{ij} = (\boldsymbol{w}_{ij}, b_{ij}) = \left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}, \frac{\|\boldsymbol{x}_j\|^2 - \|\boldsymbol{x}_i\|^2}{2\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}\right)$$

is the bisecting normal hyperplane to the segment between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is additive Gaussian noise, and Φ is the Standard Normal CDF. Indeed, if $\boldsymbol{x}_t$ is on the hyperplane h_{ij}, the answers to queries are pure coin flips, as the target point is equally far from both query points. Everywhere else, answers are biased toward the correct answer, and the probability of the correct answer depends only on the distance of $\boldsymbol{x}_t$ to the hyperplane h_{ij}. When $\boldsymbol{x}_t$ is close to - but not necessarily on - the hyperplane, the effect of random noise is still strong. As $\boldsymbol{x}_t$ becomes increasingly distant from the decision bound, the probability of the correct outcome increases, see Fig. 2.2. The model does not carry the identical-query-point property, since $p(\boldsymbol{x}_t, \boldsymbol{x}_j; \boldsymbol{x}_t) \neq 1$, and cannot be easily generalized to the k-triplet case. In Chapter 6, we will propose one possible workaround, which works well in practice, for the case when there are more than two alternatives in Q.

This model is reminiscent of pairwise comparison models such as those of [48] or [8], e.g., where $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \Phi[s(\boldsymbol{x}_i) - s(\boldsymbol{x}_j)]$ and $s(\boldsymbol{x}) = -\|\boldsymbol{x} - \boldsymbol{x}_t\|^2$, [1, 26]. These models have the undesirable property of favoring distant query points: given any $\boldsymbol{x}_i, \boldsymbol{x}_j \neq \mathbf{0}$, it is easy to verify that the pair $(2\boldsymbol{x}_i, 2\boldsymbol{x}_j)$ is strictly more discriminative for any target that does not lie on the bisecting hyperplane. The Probit model is different: in (2.11), the outcome probability depends on $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ *only* through their bisecting hyperplane.

In the previous Chapter, we have briefly mentioned that the comparison-based search algorithms typically take a Bayesian approach for modeling target position in the space of objects via discrete distribution over n points. One major advantage of Probit model is that its Gaussian noise term enables us to use d-dimensional Gaussian distribution as a prior and posterior distributions for the target location and effectively update them as the search progresses. This opens the door for more efficient search algorithms which computational complexity logarithmically depends on the size of the objects database n; which is a huge improvement over more classic Bayesian approaches to the comparison-based search problem, where the algorithms maintain full posterior over n points and have $O(n^3)$ computational complexity at each iteration of the search. We introduce these new search algorithms in Chapter 5 and compare them to the more computationally heavy state-of-the-art methods.

2.2.2 γ-CKL Model

Searching in databases with a large number n of objects could potentially cause another difficulty if the assumed comparison model does not possess the scale-invariance property. Once the belief distribution starts concentrating, as the search progresses, the information contained in queries might decrease. For example, suppose for exposition's sake that the algorithm has narrowed down the target to two candidates $\boldsymbol{x}_{t'}$ and $\boldsymbol{x}_{t''}$, and that the distance $\|\boldsymbol{x}_{t'} - \boldsymbol{x}_{t''}\|$ is small. Then any query pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ generates answers relative to $\boldsymbol{x}_{t'}$ and to $\boldsymbol{x}_{t''}$ that are nearly indistinguishable (i.e., they are Bernoulli random variables whose parameters are close). This means that the rate at which the belief distribution concentrates around $\boldsymbol{x}_t$ will slow down, leading to an unfavorable scaling of expected search cost when n grows large.

Therefore, it would be plausible to assume an oracle model that is scale-free, i.e., for which the probability of choosing $\boldsymbol{x}_i$ over $\boldsymbol{x}_j$ depends only on the ratio of $\|\boldsymbol{x}_i - \boldsymbol{x}_t\|$ to $\|\boldsymbol{x}_j - \boldsymbol{x}_t\|$. In other words, comparing two very dissimilar objects, with respect to the target that is very dissimilar from both, is not harder or easier to do than to compare two quite similar objects to a nearby target.

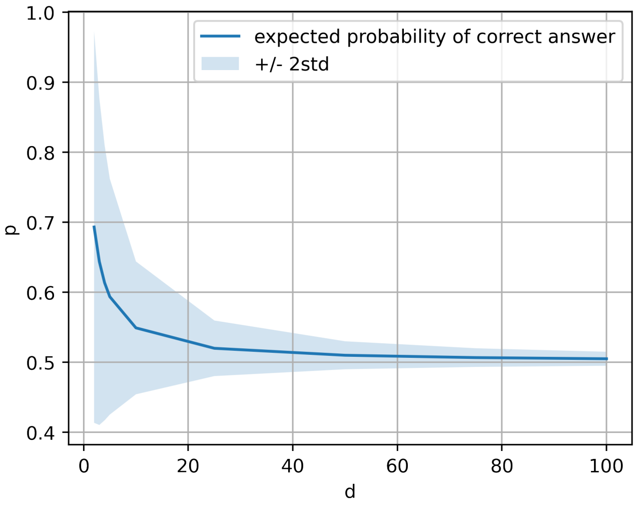


Figure 2.3: The effect of the "curse of dimensionality" with CKL model. As the dimensionality of the space d increases, the probability of choosing the closest point to the target, among two sampled points uniformly at random in unit a ball around the target, goes to $\frac{1}{2}$ under the original CKL model. The average probability of the correct answer and the 95% confidence intervals are reported over 10'000'000 random trials.

The CKL-model introduced in earlier in this Chapter is scale-free. However, one of its shortcomings is the high sensitivity to the “curse of dimensionality”: the probability of error grows quickly with d. Indeed, for a fixed target point and two query points sampled uniformly at random from a ball around the target, the predicted probability of the closest point to be chosen by the oracle (2.7) decays to 1/2 for $d \to \infty$, see Fig. 2.3. This scenario is relevant for a comparison-based search algorithms that, as discussed later in Chapter 5, will cover a potential target region with queries and rely on informative oracle replies.

To deal with this issue, we propose the following generalization of the original CKL model (2.7), by defining

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) = \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_t\|^\gamma}{\|\boldsymbol{x}_i - \boldsymbol{x}_t\|^\gamma + \|\boldsymbol{x}_j - \boldsymbol{x}_t\|^\gamma}, \tag{2.12}$$

where $\gamma \geq 1$ is an additional parameter.

	CKL/γ-CKL	STE/t-STE	Probit
k-triplet generalization	Yes	Yes	No
Identical query point property	Yes	No	No
Distant target behavior	$\frac{1}{2}$	1	$(0,1)$
Narrow query behavior	$\frac{1}{2}$	$\frac{1}{2}$	*const*
Scale-free	Yes	No	No

Table 2.1: Triplet comparison model properties.

We compare the new Probit model and the γ-CKL model to CKL and t-STE on the list of previously defined properties in Table 2.1.

Note that when γ is fixed and $d \to \infty$, the probability (2.12) for a uniformly selected pair of points $\boldsymbol{x}_i, \boldsymbol{x}_j$ goes to 1/2. On the other hand, when $\gamma \to \infty$ and d is fixed, the probability becomes an indicator function $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) \to \mathcal{I}\left\{\|\boldsymbol{x}_i - \boldsymbol{x}_t\| < \|\boldsymbol{x}_j - \boldsymbol{x}_t\|\right\}$.

This suggests that as the dimension d of the space grows (e.g. when working with high dimensional feature spaces), the new γ-CKL model should enable us to control the average outcome bias via scaling the parameter γ accordingly. In the following theorem, we show that the linear relationship between γ and d achieves this:

Theorem 1. *Consider a d-dimensional ball $\mathcal{B} \subset \mathbb{R}^d$ of radius 1. Let the target point $\boldsymbol{x}_t$ be the center of $\mathcal{B}$. For two points $\boldsymbol{x}_a, \boldsymbol{x}_b$ sampled uniformly from $\mathcal{B}$ let p_Q be the probability of the correct answer, i.e. choosing the closest point to the target, for a query $Q = \{\boldsymbol{x}_a, \boldsymbol{x}_b\}$ given the* *target* $\boldsymbol{x}_t$ *under the* γ*-CKL model with* $\gamma \geq 2$*,*

$$\begin{aligned} p_Q &= p(\boldsymbol{x}_a, \boldsymbol{x}_b; \boldsymbol{x}_t)\mathbb{1}\{\|\boldsymbol{x}_a - \boldsymbol{x}_t\| < \|\boldsymbol{x}_b - \boldsymbol{x}_t\|\} \\ &\quad + p(\boldsymbol{x}_b, \boldsymbol{x}_a; \boldsymbol{x}_t)\mathbb{1}\{\|\boldsymbol{x}_a - \boldsymbol{x}_t\| > \|\boldsymbol{x}_b - \boldsymbol{x}_t\|\}. \end{aligned}$$

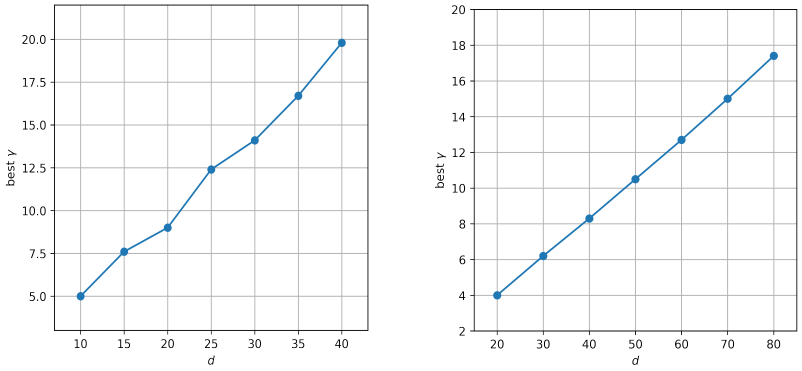
For any $c_2 \in [\frac{1}{2}, 1]$ *there is a constant* $c_1 > 0$*, s.t. if* γ *grows linearly with* d*,* $\gamma = c_1 d + o(d)$*, then* $p_Q = c_2 + o(1)$*.*

Proof sketch. When d grows large and γ as in the assumption, most of the points have a norm close to 1. This allows us to use Taylor approximation for (2.12). Then we show show that the error term becomes negligible as d approaches ∞. $\square$

In this way, scaling γ linearly with d helps to prevent the probabilities of random queries from becoming perfect coin flips, i.e. when both query items become equiprobable, and forces them to remain almost unchanged as d grows, up to an additional factor of $o(1)$. This insight is useful for practical applications of γ-CKL model. Usually, the dimensionality of the hidden features space is unknown and has to be estimated from the triplet outcomes data (see discussion in Chapter 4). Thus d and γ serve as model hyperparameters and are cross validated. Then the linear relationship between γ and d helps to shrink the search space of the potential values.

This linear relationship is also confirmed even for small values of γ and d through the following experiment.

Experiment on the relationship between γ and d. Let us first fix the true reference values $d^\star$ and $\gamma^\star$ for which compute the target average probability of the correct answer $p_Q(\gamma^\star, d^\star)$. Now, as d begins to grow, we would like to scale γ accordingly, in order to maintain average $p_Q(\gamma, d)$ to be as close as possible to the original $p_Q(\gamma^\star, d^\star)$ To simulate that, we iterate over the values of $d > d^\star$ and for each we find the corresponding γ that minimizes $|p_Q(\gamma^\star, d^\star) - p_Q(\gamma, d)|$ via a gridsearch. The best values of γ are reported in Fig. 2.4. In all trials we kept $N = 1000$ and $|\mathcal{T}| = 10'000$. We observe a linear relationship between γ and d even for finite values of d, which matches the limit result of Theorem 1.



(a) $d^\star = 10, \gamma^\star = 5$

(b) $d^\star = 20, \gamma^\star = 4$

Figure 2.4: Linear relationship between γ and d for finite values of d.

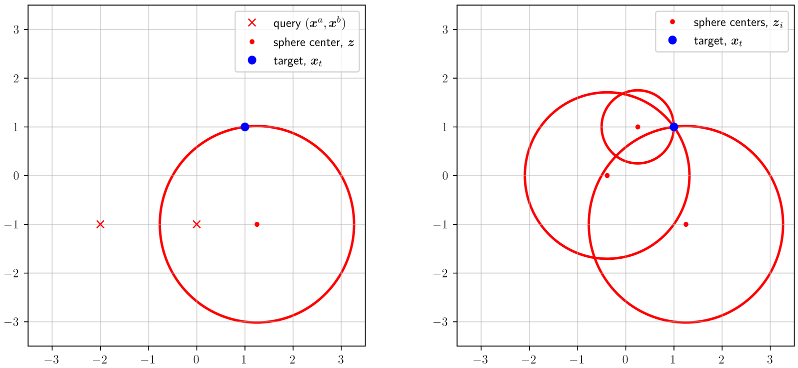
We can also provide some intuition on the geometrical structure of a set of queries that are rich enough to identify the target $\boldsymbol{x}_t \in \mathbb{R}^d$ under the γ-CKL model. In particular, we describe the properties of a finite set of queries $\hat{\mathcal{Q}} = \{\hat{Q}_1, \hat{Q}_2, \dots, \hat{Q}_L\}$, for which, by repeating each query $Q_i \in \hat{\mathcal{Q}}$ infinitely many times, we would be able to provably identify the target $\boldsymbol{x}_t$.

Proposition 1. *Assume oracle answers queries according to* γ*-CKL model* $p(\boldsymbol{x}_a, \boldsymbol{x}_b; \boldsymbol{x}_t)$. *Assume that* $\Omega \subset \mathbb{R}^d$ *is* d*-dimensional compact set and that the target* $\boldsymbol{x}_t$ *is sampled uniformly at random from* Ω. *Consider an infinite sequence of queries* $\mathcal{Q} = \{Q_0, Q_1, \dots\}$ *that is asked to the oracle, where each* $Q_i \in \hat{\mathcal{Q}} = \{\hat{Q}_1, \hat{Q}_2, \dots, \hat{Q}_L\}$ *and each* $\hat{Q}_i$, $i = 1, 2, \dots, L$, *is repeated infinitely many times. Also for each* $\hat{Q}_i = (\hat{\boldsymbol{x}}_i^a, \hat{\boldsymbol{x}}_i^b)$ *let* $c_i = \|\hat{\boldsymbol{x}}_i^a - \boldsymbol{x}_t\| / \|\hat{\boldsymbol{x}}_i^b - \boldsymbol{x}_t\|$ *and* $\hat{\boldsymbol{z}}_i = (c_i \hat{\boldsymbol{x}}_i^b - \hat{\boldsymbol{x}}_i^a)/(1 - c_i)$. *If* $\hat{\mathcal{Q}}$ *satisfies* $\operatorname{rank}(\boldsymbol{Z}) = d$, *where* $\boldsymbol{Z}$ *is the* $d \times (L-1)$ *matrix of vectors* $\{(\hat{\boldsymbol{z}}_i - \hat{\boldsymbol{z}}_L) : \hat{Q}_i \in \hat{\mathcal{Q}}, i = 1, \dots, L-1\}$, *then*

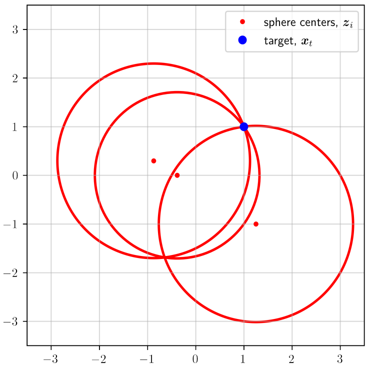
$$\operatorname*{arg\,max}_{\boldsymbol{x} \in \Omega} \mathbb{E}[p(\boldsymbol{x} \mid Y_{1:m})] \to \boldsymbol{x}_t$$

as $m \to \infty$, *where the expectation is taken over the oracle answers* $Y_{1:m} = Y_1, Y_2, \dots, Y_m$.

Proof sketch. First we show that, after repeating one query $Q_1 = (\boldsymbol{x}_{1_1}, \boldsymbol{x}_{1_2})$ infinitely many times, the subset of points of Ω that has the highest likelihood geometrically forms a sphere $\mathcal{S}_1 \subset \mathbb{R}^d$ with a center at $\boldsymbol{z}_1$, and of course $\boldsymbol{x}_t \in \mathcal{S}_1$. If we then ask a new query $Q_2 = (\boldsymbol{x}_{2_1}, \boldsymbol{x}_{2_2})$, different from Q_1, also by repeating it infinitely many times, the new set of points with highest likelihood for Q_1 and Q_2 will lie in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., at least in a $(d-1)$-dimensional sphere $\mathcal{S}_1 \cap \mathcal{S}_2$. We show that by sequentially constructing a set of queries, where for each additional query Q_{i+1} its induced sphere's center $\boldsymbol{z}_{i+1}$ does not lie on the hyperplane spanned by $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_i\}$, we decrease the dimensionality of $\mathcal{S}_1 \cap \dots \cap \mathcal{S}_i$ at least by 1. In total, $d+1$ queries constructed in this way will ensure that the corresponding $d+1$ spheres intersect at exactly one point, $\boldsymbol{x}_t$. $\square$



(a) One query $\{\boldsymbol{x}^a, \boldsymbol{x}^b\}$ (b) rank($\boldsymbol{Z}$) = 2



(c) rank($\boldsymbol{Z}$) = 1

Figure 2.5: Illustration of Proposition 1 for $d = 2$. We assume that each query is repeated infinitely many times.

Figure 2.5 illustrates the proposition in the case $d = 2$. For each query the subset of points in Ω that maximizes the likelihood of observed query outcomes geometrically is a sphere containing $\boldsymbol{x}_t$ with center $\boldsymbol{z}$, see Figure 2.5(a). Hence, if we repeat this query infinitely many times, the set of points that maximize the posterior probability will also have a form of that sphere in the limit. When the set of sphere centers $\{\boldsymbol{z}_i\}$ corresponding to the queries $\mathcal{Q}_\ell$ do not lay on the hyperplane in $\mathbb{R}^d$, these spheres intersect at exactly one point, $\boldsymbol{x}_t$, which is the only point that the joint likelihood, see Figure 2.5(b). Otherwise, there could be multiple points of common intersection of spheres (including $\boldsymbol{x}_t$), like in Figure 2.5(c), which will result into multimodal posterior in the limit.

Based on this result, we could construct a heuristic search algorithm that would consequently choose a set of $d+1$ queries such that their middle points $\|\boldsymbol{x}_i^a + \boldsymbol{x}_i^b\|/2$, which lie on the same line as $\boldsymbol{x}_i^a$, $\boldsymbol{x}_i^b$, and $\boldsymbol{z}_i$, span a volume in $\mathbb{R}^d$, which would help to remove the ambiguity in identifying the target position $\boldsymbol{x}_t$ after repeating each query infinitely many times. However in order to build a set of $d+1$ queries that satisfy the assumptions of the proposition, it is required to know (or at least have an estimate of) the likelihoods of the answers of each query, forcing to repeat queries to the user, which will not be suitable for practical applications. In Chapter 5, we introduce more rigorous and efficient search schemes that can efficiently locate the target. But first, we will demonstrate how the new proposed models compare to the previously existing ones for modeling human choices on real-world datasets.

2.3 Proofs

In this section we present proofs of Theorem 1 and Proposition 1.

2.3.1 Theorem 1

Proof of Theorem 1. Consider two points $\boldsymbol{x}_a, \boldsymbol{x}_b \in \mathbb{R}^d$ sampled uniformly from a unit ball $\mathcal{B}$ that form a query to the oracle $Q = (\boldsymbol{x}_a, \boldsymbol{x}_b)$. After asking Q we observe the answer $Y \in \{\boldsymbol{x}_a, \boldsymbol{x}_b\}$ under the γ-CKL model for some fixed $\gamma \geq 2$. Then the probability that the answer Y is correct, p_Q, is

$$\begin{aligned}
p_Q &= \int_{r_1=0}^{1}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} S_d(r_1) S_d(r_2) \frac{1}{V_d}\frac{1}{V_d} dr_1 dr_2 \\
&+ \int_{r_1=0}^{1}\int_{r_2=0}^{r_1} \frac{r_1^\gamma}{r_1^\gamma + r_2^\gamma} S_d(r_1) S_d(r_2) \frac{1}{V_d}\frac{1}{V_d} dr_1 dr_2 \\
&= \int_{r_1=0}^{1}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 \qquad (2.13)\\
&+ \int_{r_1=0}^{1}\int_{r_2=0}^{r_1} \frac{r_1^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2, \qquad (2.14)
\end{aligned}$$

where

$$S_d(r) = \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})} r^{d-1},\ V_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}$$

are the respective surface and volume of the unit ball $\mathcal{B}$.

Consider (2.13),

$$\int_{r_1=0}^{1}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2.$$

If we increase d, the distance from the center of the ball to a random inside point will be close to 1. We use Taylor approximation of the probability model at $(1,1) \in \mathbb{R}^2$:

$$\begin{aligned}\frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} &= \frac{1}{2} - (r_1 - 1)\frac{\gamma}{4} + (r_2 - 1)\frac{\gamma}{4} + R(r_1, r_2)\\ &= P(r_1, r_2) + R(r_1, r_2).\end{aligned}$$

Let's fix some $0 < \varepsilon < 1$. Then

$$\begin{aligned}(2.13) &= \int_{r_1=0}^{1}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2\\ &= \int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 + \int_{r_1=0}^{\varepsilon}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2\\ &= \int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} P(r_1, r_2) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 + \int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} R(r_1, r_2) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2\\ &\quad + \int_{r_1=0}^{\varepsilon}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2.\end{aligned}$$

First note that the last summand is $o(1)$ when $d \to \infty$:

$$\begin{aligned}\int_{r_1=0}^{\varepsilon}\int_{r_2=r1}^{1} \frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 &\le \int_{r_1=0}^{\varepsilon}\int_{r_2=r1}^{1} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2\\ &\le \frac{1}{2}\varepsilon^d (2 - \varepsilon^d) = o(1).\end{aligned}$$

Now the integral with the $P(r_1, r_2)$ term can be computed as follows:

$$
\begin{aligned}
&\int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} P(r_1,r_2) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 = \\
=&\int_{r_1=0}^{1}\int_{r_2=r1}^{1} \frac{1}{2} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 + \frac{1}{4}\varepsilon^d - \frac{1}{2}\varepsilon^{2d} \\
&+\int_{r_1=0}^{1}\int_{r_2=r1}^{1} (r_1-1)\frac{\gamma}{4} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 + \frac{\gamma d\varepsilon^d}{4}\left(\frac{\varepsilon^2-2}{2d} + \varepsilon\left(\frac{1}{d+1} - \frac{\varepsilon^d}{2d+1}\right)\right) \\
&+\int_{r_1=0}^{1}\int_{r_2=r1}^{1} (r_2-1)\frac{\gamma}{4} r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2 \\
&+\frac{\gamma\varepsilon^d}{8(d+1)(2d+1)}\Big((d(2d(\varepsilon-1)-3)-1)\varepsilon^d + 2(2d+1)\Big) \\
=&\frac{1}{4} + \frac{\gamma}{4}\frac{d^2(3d+1)}{2d^2(d+1)(2d+1)} - \frac{\gamma}{4}\frac{d^2}{4d^3+2d^2} + o(1) \\
=&\frac{1}{4} + \frac{\gamma}{4}\frac{d}{(d+1)(2d+1)} + o(1).
\end{aligned}
$$

Finally, consider the remaining integral,

$$
\int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} R(r_1,r_2) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2.
$$

Using Taylor's theorem for multivariate functions, we can get an upper bound for its absolute value:

$$
\begin{aligned}
&\left|\int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} R(r_1,r_2) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2\right| \\
\le& \frac{M(\gamma)}{2}\int_{\mathcal{X}} \left((r_1-1)^2 + 2(r_1-1)(r_2-1) + (r2-1)^2\right) r_1^{d-1} r_2^{d-1} d^2 dr_1 dr_2
\end{aligned}
$$

where

$$
\begin{aligned}
M(\gamma) &= \max_{\alpha=|2|,(r_1,r_2)\in\mathcal{X}} \left| D^\alpha\left[\frac{r_2^\gamma}{r_1^\gamma + r_2^\gamma}\right]\right|, \\
\mathcal{X} &= \{(r_1,r_2) \mid r_1 \in [\varepsilon,1],\ r_2 \in [r_1,1]\},
\end{aligned}
$$

and

$$\left|D^{(2,0)}\left[\frac{r_2^\gamma}{r_1^\gamma+r_2^\gamma}\right]\right| = \left|\frac{\gamma r_2^{\gamma-2} r_1^\gamma((\gamma-1)r_1^\gamma-(\gamma+1)r_2^\gamma)}{(r_1^\gamma+r_2^\gamma)^3}\right|,$$

$$\left|D^{(1,1)}\left[\frac{r_2^\gamma}{r_1^\gamma+r_2^\gamma}\right]\right| = \frac{\gamma^2 r_2^{\gamma-1} r_1^{\gamma-1}(r_2^\gamma-r_1^\gamma)}{(r_1^\gamma+r_2^\gamma)^3},$$

$$\left|D^{(0,2)}\left[\frac{r_2^\gamma}{r_1^\gamma+r_2^\gamma}\right]\right| = \left|\frac{\gamma r_2^{\gamma} r_1^{\gamma-2}((\gamma+1)r_1^\gamma)-(\gamma-1)r_2^\gamma}{(r_1^\gamma+r_2^\gamma)^3}\right|.$$

For a big enough d, if γ grows with d, the maximum of $M(\gamma)$ is achieved when $r_1 = r_2$ with $M(\gamma) \le \frac{\gamma}{4}\varepsilon^{-2}$. We will show this for $\left|D^{(2,0)}\left[\frac{r_2^\gamma}{r_1^\gamma+r_2^\gamma}\right]\right|$, the other two cases can be proved similarly. Indeed,

$$\begin{aligned}\left|D^{(2,0)}\left[\frac{r_2^\gamma}{r_1^\gamma+r_2^\gamma}\right]\right| &= \left|\frac{\gamma r_2^{\gamma-2} r_1^\gamma((\gamma-1)r_1^\gamma-(\gamma+1)r_2^\gamma)}{(r_1^\gamma+r_2^\gamma)^3}\right| \\ &= \gamma\frac{r_2^{\gamma-2} r_1^\gamma((\gamma+1)r_2^\gamma-(\gamma-1)r_1^\gamma)}{(r_1^\gamma+r_2^\gamma)^3} \\ &= \gamma\frac{\left(\frac{r_2}{r_1}\right)^\gamma((\gamma+1)\left(\frac{r_2}{r_1}\right)^\gamma-(\gamma-1))}{r_2^2(1+\left(\frac{r_2}{r_1}\right)^\gamma)^3},\end{aligned}$$

which is equal to $\frac{\gamma}{4}\varepsilon^{-2}$ when $r_1 = r_2$ and goes to 0 with $d \to \infty$ when $r_1 < r_2$.

Finally

$$\begin{aligned}&\int_{\mathcal{X}}\left((r_1-1)^2+2(r_1-1)(r_2-1)+(r2-1)^2\right)r_1^{d-1}r_2^{d-1}d^2dr_1dr_2 \\ &= \varepsilon^{2d}P_1+\varepsilon^d P_2+\frac{3d+4}{(d+1)^2(d+2)},\end{aligned}$$

where

$$P_1 = -\frac{d\left(d^2+(d+1)^2\varepsilon^2-2(d+2)^2\varepsilon+6d+13\right)+8}{(d+1)^2(d+2)}$$

and

$$P_2 = \frac{(2(d+2)d+1)d\varepsilon^2-4d(d+2)(d+1)\varepsilon+2(d+2)(d+1)^2}{(d+1)^2(d+2)}$$

are two polynomial fractions.

Putting everything together we can upper bound the remainder by

$$\left|\int_{r_1=\varepsilon}^{1}\int_{r_2=r1}^{1} R(r_1,r_2)r_1^{d-1}r_2^{d-1}d^2dr_1dr_2\right| \le \frac{\gamma\varepsilon^{-2}}{8}\left(\varepsilon^{2d}P_1+\varepsilon^{d}P_2+\frac{3d+4}{(d+1)^2(d+2)}\right).$$

Also, due to symmetry, (2.13) = (2.14), and thus

$$p_Q = \frac{1}{2}+\frac{\gamma}{2}\frac{d}{(d+1)(2d+1)}+\hat{R}+o(1)$$

where

$$|\hat{R}| \le \frac{\gamma\varepsilon^{-2}}{4}\left(\varepsilon^{2d}P_1+\varepsilon^{d}P_2+\frac{3d+4}{(d+1)^2(d+2)}\right).$$

We see that if

$$\frac{\gamma}{d} = c_1 + o(1)$$

and $d \to \infty$, then

$$p_Q = c_2 + o(1),$$

where $c_1 > 0$, $c_2 > 0$ are constants.

□

2.3.2 Proposition 1

Proof of Proposition 1. First we will show that for a query $Q_i = (\boldsymbol{x}_i^a, \boldsymbol{x}_i^b)$ the set of points $\mathcal{S}_i \subset \Omega$ that have the same probabilities of the observing an Y forms a d-dimensional sphere. For ease of reading, we drop the index i and simply write $Q = (\boldsymbol{x}_a, \boldsymbol{x}_b)$.

The oracle answers $Y = \boldsymbol{x}_a$ with probability $p(\boldsymbol{x}_a,\boldsymbol{x}_b;\boldsymbol{x}_t) = \frac{\|\boldsymbol{x}_b-\boldsymbol{x}_t\|^\gamma}{\|\boldsymbol{x}_a-\boldsymbol{x}_t\|^\gamma+\|\boldsymbol{x}_b-\boldsymbol{x}_t\|^\gamma}$. Let us consider all points $\boldsymbol{x} \in \Omega$ for which $p_{\boldsymbol{x}_a,\boldsymbol{x}_b,\boldsymbol{x}} = p_{\boldsymbol{x}_a,\boldsymbol{x}_b,\boldsymbol{x}_t}$. Now denoting

$$c := \frac{\|\boldsymbol{x}_a-\boldsymbol{x}_t\|^2}{\|\boldsymbol{x}_b-\boldsymbol{x}_t\|^2} = \left(\frac{1}{p}-1\right)^{\frac{2}{\gamma}},$$

and observing

$$p = \frac{\|\boldsymbol{x}_b-\boldsymbol{x}_t\|^\gamma}{\|\boldsymbol{x}_a-\boldsymbol{x}_t\|^\gamma+\|\boldsymbol{x}_b-vx_t\|^\gamma} = \frac{1}{\frac{\|\boldsymbol{x}_a-\boldsymbol{x}_t\|^\gamma}{\|\boldsymbol{x}_b-\boldsymbol{x}_t\|^\gamma}+1},$$

we can define the set $\mathcal{S}_Q$ by

$$\sum_{j=1}^{d} (\boldsymbol{x}_j - (\boldsymbol{x}_a)_j)^2 - c \sum_{i=j}^{D} (\boldsymbol{x}_j - (\boldsymbol{x}_b)_j)^2 = 0,$$

which can be rewritten as

$$\sum_{j=1}^{d} (\boldsymbol{x}_j - \boldsymbol{z}_j)^2 = r.$$

for

$$\begin{aligned} z_j &= \frac{c(\boldsymbol{x}_b)_j - (\boldsymbol{x}_a)_j}{1-c}, \\ r &= \sum_{j=1}^{d} \frac{(c(\boldsymbol{x}_b)_j - (\boldsymbol{x}_a)_j)^2}{(1-c)^2} - \frac{(\boldsymbol{x}_a)_j^2 - c(\boldsymbol{x}_b)_j^2}{(1-c)}. \end{aligned}$$

Hence, for a fixed query, the points that have the same outcome probabilities as $\boldsymbol{x}_t$ form a sphere in $\mathbb{R}^d$. For two distinct query points Q_1 and Q_2, the set of points that have the same outcome probabilities of answer for both Q_1 and Q_2 will lie in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., at least in a $(d-1)$-dimensional sphere $\mathcal{S}_1 \cap \mathcal{S}_2$. Consider the third query point Q_3. The intersection $\mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3$ is at least a $(d-2)$-dimensional sphere if $\boldsymbol{z}_3$ does not lie on the line intersecting $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, otherwise the points in $\mathcal{S}_1 \cap \mathcal{S}_2$ are equidistant from $\boldsymbol{z}_3$, and since $\boldsymbol{x}_t \in \mathcal{S}_1 \cap \mathcal{S}_2$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3$, and no additional dimensionality reduction of the spheres intersection is achieved (see Fig 2.5 for illustration). Similarly, for $d+1$ queries $Q_1, Q_2, \ldots, Q_{d+1}$, the sufficient condition for

$$\mathcal{S}_1 \cap \mathcal{S}_2 \cap \cdots \cap \mathcal{S}_{d+1} = \boldsymbol{x}_t$$

is

$$\operatorname{rank}(\tilde{\boldsymbol{z}}_1 - \tilde{\boldsymbol{z}}_{d+1}, \tilde{\boldsymbol{z}}_2 - \tilde{\boldsymbol{z}}_{d+1}, \ldots, \tilde{\boldsymbol{z}}_d - \tilde{\boldsymbol{z}}_{d+1}) = d. \tag{2.15}$$

The intersection of the $d+1$ corresponding spheres will result in exactly one point, $\boldsymbol{x}_t$. Thus the likelihood after $d+1$ such queries will be maximized only at $\boldsymbol{x}_t$. Now by chosing a uniform prior over Ω, in expectation over the outcomes of any set of queries $\tilde{\mathcal{Q}}$ that satisfies (2.15) the posterior will be maximized only at $\boldsymbol{x}_t$ and then the claim follows immediately. $\square$

3 Technical Background for Bayesian Inference

In this Chapter we describe some important techniques from Bayesian Inference that will be used in the later Chapters of this thesis.

In the problem of Bayesian inference we are trying to estimate a latent random variable $\boldsymbol{x}$ via observing data D about $\boldsymbol{x}$. We assume a prior distribution $p(\boldsymbol{x})$ for $\boldsymbol{x}$ before any data is observed. We also assume a probabilistic model $p(D|\boldsymbol{x})$ that gives us the likelihood of observing data D when $\boldsymbol{x}$ value is fixed. Once $p(\boldsymbol{x})$ and $p(D|\boldsymbol{x})$ are defined, and data D is observed, we would like to update our belief about $\boldsymbol{x}$ based on the information D, i.e. compute the posterior distribution of $\boldsymbol{x}$:

$$p(\boldsymbol{x}|D) = \frac{p(D|\boldsymbol{x})p(\boldsymbol{x})}{p(D)},$$

where $p(D)$ is the evidence. It is sometimes also written as

$$p(\boldsymbol{x}|D) \propto p(D|\boldsymbol{x})p(\boldsymbol{x}),$$

since $p(D)$ can be seen as a constant.

When $\boldsymbol{x}$ is a discrete random variable with a finite support $\mathcal{X}$, this updates is trivial and can be computed in $O(n)$, where n is the cardinality of the support. Indeed, computing $p(\boldsymbol{x}|D)$ involves for each $\boldsymbol{x}_i \in \mathcal{X}$ computing

$$p(\boldsymbol{x}_i|D) = \frac{p(D|\boldsymbol{x}_i)p(\boldsymbol{x}_i)}{\sum_{i=1}^{n} p(D|\boldsymbol{x}_i)p(\boldsymbol{x}_i)}.$$

In this work we refer to this scenario as a Bayesian update with full posterior on n points.

In the case when $\boldsymbol{x}$ is continuous, depending on the choice of the prior $p(\boldsymbol{x})$ and the likelihood function $p(D|\boldsymbol{x})$, (a) the posterior distribution might not lie in the same distribution family as

$p(\boldsymbol{x})$ and (b) there could be no closed form expression for $p(\boldsymbol{x}|D)$. The term $p(D)$ becomes an integral

$$p(D) = \int p(D|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x},$$

and is usually intractable to compute.

Below we describe two well-known methods to approximately compute $p(\boldsymbol{x}|D)$ in the continuous case. They are both based on the idea of finding an approximation of $p(\boldsymbol{x}|D)$ by another distribution $q(\boldsymbol{x})$ from some family Q (e.g. Gaussian) via minimizing the Kullback–Leibler divergence between the two.

The first method is called Expectation-Propagation [36] and it is aimed at minimizing

$$\min_{\tilde{q}\in Q} \mathrm{KL}(p(\boldsymbol{x}|D) \mid\mid \tilde{q}(\boldsymbol{x})).$$

We focus on a special case of this method when Q is a family of Gaussian distributions. In that case the idea of the EP method is to find a distribution $\tilde{q}(\boldsymbol{x})$ that has the same first two moments as $p(\boldsymbol{x}|D)$. We will derive a posterior update following this moment-matching idea and assuming Probit likelihood and a Gaussian prior.

The second moment is called Variational Inference and aims at minimizing the reverse

$$\min_{\tilde{q}\in Q} \mathrm{KL}(\tilde{q}(\boldsymbol{x}) \mid\mid p(\boldsymbol{x}|D)).$$

In this method the KL divergnce can be re-written as a sum of two terms to minimize which can be done via a gradient method with sampling. We will describe a general formulation of this method below.

3.1 Moment-Matching for Probit Model

Assume we have a Gaussian prior $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, and observe an outcome according to the Probit model with respect to some hyperplane $(\boldsymbol{w}, b)$. Then the posterior distribution will be $q(\boldsymbol{x}) \propto t(\boldsymbol{x}^T \boldsymbol{w} + b)p(\boldsymbol{x})$, where

$$t(x) = \Phi\left(\frac{x}{\sigma_\varepsilon}\right),$$

is the likelihood function based on the Gaussian CDF.

Using the idea of Expectation Propagation [36], we would like to find a Gaussian approximation

$\tilde{q}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$ of the true posterior q s.t. it minimizes the KL divergence

$$\mathrm{KL}(q(\boldsymbol{x}) \mid\mid \tilde{q}(\boldsymbol{x})).$$

Natural parameters. Since $p(x)$ is assumed to be Gaussian, as a part of the Exponential family of distributions, it has natural parameters. Let $(\boldsymbol{\nu}, \boldsymbol{P}) = (\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1})$ be the natural parameters of $p(\boldsymbol{x})$ and $(\nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) = ((\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w})^{-1}(\boldsymbol{\mu}^T\boldsymbol{w} + b), (\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w})^{-1})$ be the natural parameters of projected $p_{\text{proj}}(u) := p(\boldsymbol{x}^T\boldsymbol{w} + b) = \mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \boldsymbol{\mu}^T\boldsymbol{w} + b, \boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w})$. Then

$$\begin{aligned} q(\boldsymbol{x}) \propto\ & t(\boldsymbol{x}^T\boldsymbol{w} + b)p(\boldsymbol{x}) = t(\boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\nu}, \boldsymbol{P}) \\ &= t(\boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}). \end{aligned}$$

Now note, that finding a Gaussian approximation $\tilde{q}(\boldsymbol{x})$ that minimizes KL divergence is equivalent to finding a Gaussian likelihood that approximates $t(\boldsymbol{x}^T\boldsymbol{w} + b)$ that minimizes KL divergence, i.e. finding scalars $\tilde{\nu}$ and $\tilde{\tau}$ s.t.

$$\begin{aligned} t(\boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) &\approx \mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \tilde{\nu}, \tilde{\tau})\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) && (3.1) \\ &= \mathcal{N}(\boldsymbol{x} \mid (\tilde{\nu} - b\tilde{\tau})^T\boldsymbol{w}, \boldsymbol{w}^T\tilde{\tau}\boldsymbol{w})\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\nu}, \boldsymbol{P}) && (3.2) \\ &= \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\nu} + (\tilde{\nu} - b\tilde{\tau})^T\boldsymbol{w}, \boldsymbol{P} + \boldsymbol{w}^T\tilde{\tau}\boldsymbol{w}) && (3.3) \\ &\propto\ \tilde{q}(\boldsymbol{x}). && (3.4) \end{aligned}$$

Moment matching Now we will find $\tilde{\nu}$ and $\tilde{\tau}$ so that the first two moments of

$$\begin{aligned} \tilde{q}(\boldsymbol{x}) &\propto \mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \tilde{\nu}, \tilde{\tau})\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) \\ &= \mathcal{N}(u \mid \tilde{\nu}, \tilde{\tau})\mathcal{N}(u \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) \\ &=: \tilde{q}_{\text{proj}}(u). \end{aligned}$$

match the first two moments of the true posterior

$$\begin{aligned} q(\boldsymbol{x}) &\propto\ t(\boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \boldsymbol{\mu}^T\boldsymbol{w} + b, \boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}) \\ &= t(\boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x}^T\boldsymbol{w} + b \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) \\ &= t(u)\mathcal{N}(u \mid \nu_{p_{\text{proj}}}, \tau_{p_{\text{proj}}}) \\ &=: q_{\text{proj}}(u). \end{aligned}$$

First we will find the first two moments of the true posterior $q_{\text{proj}}(u)$. Let

$$Z(m, s^2) = \int t(u)\mathcal{N}(u \mid m, s^2)du$$

be the partition function with

$$\begin{aligned} m &= \boldsymbol{\mu}^T \boldsymbol{w} + b, \\ s^2 &= \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w}. \end{aligned}$$

We can find the moments of $q_{\text{proj}}(u)$ by computing the first and second derivatives of the log-partition function with respect to m.

$$\begin{aligned} \alpha &:= \frac{\partial}{\partial m} \log Z(m, \boldsymbol{\Sigma}) = \frac{1}{Z} \int t(u) \frac{\partial}{\partial m} p_{\text{proj}}(u) du = \frac{1}{Z} \int t(u) \frac{u - m}{s^2} p_{\text{proj}}(u) du \\ &= s^{-2} \left(\int u \frac{t(u) p_{\text{proj}}(u)}{Z} du - m \frac{1}{Z} \int t(u) p_{\text{proj}}(u) du \right) \\ &= s^{-2} (\mathbb{E}_{q_{\text{proj}}}[u] - m), \end{aligned}$$

Similarly,

$$\begin{aligned} \beta &:= \frac{\partial^2}{\partial m^2} \log Z(m, \boldsymbol{\Sigma}) = \frac{1}{Z} \int t(u) \frac{\partial^2}{\partial m^2} p_{\text{proj}}(u) du - \alpha \\ &= s^{-2} + s^{-4} (\mathbb{E}_{q_{\text{proj}}}[u^2] - \mathbb{E}_{q_{\text{proj}}}[u]^2). \end{aligned}$$

Thus

$$\begin{aligned} \mathbb{E}_{q_{\text{proj}}}[u] &= m + s^2 \alpha \\ \mathbb{E}_{q_{\text{proj}}}[u^2] - \mathbb{E}_{q_{\text{proj}}}[u]^2 &= s^2 + s^4 \beta. \end{aligned}$$

Now we match the first two moments of approximation $\tilde{q}_{\text{proj}}(u)$ to the first two moments of the true $q_{\text{proj}}(u)$, and thus the natural parameters of $\tilde{q}_{\text{proj}}(u)$ will be

$$\begin{aligned} \nu_{\tilde{q}_{\text{proj}}(u)} &= \frac{m + s^2 \alpha}{s^2 + s^4 \beta}, \\ \tau_{\tilde{q}_{\text{proj}}(u)} &= (s^2 + s^4 \beta)^{-1}. \end{aligned}$$

Now we can write the full expressions for $\tilde{\nu}$ and $\tilde{\tau}$:

$$\tilde{\nu} = \nu_{\tilde{q}_{\text{proj}}(u)} - \nu_{p_{\text{proj}}} = \frac{m + s^2\alpha}{s^2 + s^4\beta} - \frac{m}{s^2} = \frac{\alpha - m\beta}{1 + s^2\beta}, \tag{3.5}$$

$$\tilde{\tau} = \tau_{\tilde{q}_{\text{proj}}(u)} - \tau_{p_{\text{proj}}} = (s^2 + s^4\beta)^{-1} - s^{-2} = -\frac{\beta}{1 + s^2\beta}. \tag{3.6}$$

Proposition 2. *Assume a Gaussian prior $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a Probit likelihood function (2.11). Let $Z(\mu, \boldsymbol{\Sigma})$ be defined as above. Then*

$$\begin{aligned} Z(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \int \Phi\left(\frac{\boldsymbol{x}^T\boldsymbol{w} + b}{\sigma_\varepsilon}\right)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})\,d\boldsymbol{x} \\ &= \Phi\left(\frac{m}{\sqrt{s^2 + \sigma_\varepsilon^2}}\right), \end{aligned}$$

where

$$\begin{aligned} m &= \boldsymbol{\mu}^T\boldsymbol{w} + b \\ s^2 &= \boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}. \end{aligned}$$

Proof. Let us denote $X = \boldsymbol{x}^T\boldsymbol{w} + b \sim \mathcal{N}(x \mid \underbrace{\boldsymbol{\mu}^T\boldsymbol{w} + b}_{m}, \underbrace{\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}}_{s})$. Then,

$$P(X + \varepsilon \leq 0) = P(\boldsymbol{x}^T\boldsymbol{w} + b + \varepsilon \leq 0) = \Phi\left(\frac{m}{\sqrt{s + \sigma_\varepsilon^2}}\right).$$

On the other hand, using law of total probability,

$$\begin{aligned} P(X + \varepsilon \leq 0) &= \int P(X + \varepsilon \leq 0 \mid X = \boldsymbol{x}^T\boldsymbol{w} + b)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})\,d\boldsymbol{x} \\ &= \int P(\boldsymbol{x}^T\boldsymbol{w} + b + \varepsilon \leq 0)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})\,d\boldsymbol{x} \\ &= \int P(\varepsilon \leq -(\boldsymbol{x}^T\boldsymbol{w} + b))\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})\,d\boldsymbol{x} \\ &= \int \Phi\left(\frac{\boldsymbol{x}^T\boldsymbol{w} + b}{\sigma_\varepsilon}\right)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})\,d\boldsymbol{x} \\ &= Z(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \end{aligned}$$

□

Then in our one dimensional case $Z(m, s^2)$ we simply have

$$Z(m, s^2) = \Phi\left(\frac{m}{\sqrt{s^2 + \sigma_\varepsilon^2}}\right),$$

and the first two derivatives will be

$$\alpha := \frac{\partial}{\partial m} \log Z(m, s^2) = \frac{\mathcal{N}(z \mid 0, 1)}{\Phi(z)\sqrt{s^2 + \sigma_\varepsilon^2}}, \tag{3.7}$$

$$\beta := \frac{\partial^2}{\partial m^2} \log Z(m, s^2) = \frac{-z\mathcal{N}(z \mid 0, 1)\Phi(z) - \mathcal{N}(z \mid 0, 1)^2}{\Phi(z)^2(s^2 + \sigma_\varepsilon^2)} \tag{3.8}$$

$$= -\frac{\mathcal{N}(z \mid 0, 1)}{(s^2 + \sigma_\varepsilon^2)\Phi(z)}\left(z + \frac{\mathcal{N}(z \mid 0, 1)}{\Phi(z)}\right) \tag{3.9}$$

with

$$z = \frac{m}{\sqrt{s^2 + \sigma_\varepsilon^2}}.$$

3.2 Variational Inference

In the Variational Inference method, the idea is to find a distribution $q(\boldsymbol{x})$ that minimizes

$$\min_{q \in Q} \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x}|D)).$$

In this case by using the definition of the KL divergence we have

$$\begin{aligned} \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x}|D)) &= \mathbb{E}_q[\log q(\boldsymbol{x}) - \log p(\boldsymbol{x}|D)] \\ &= \mathbb{E}_q[\log q(\boldsymbol{x}) - \log p(\boldsymbol{x}) - \log p(D|\boldsymbol{x}) + \log p(D)] \\ &= \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})) - \mathbb{E}_q[\log p(D|\boldsymbol{x})] + \log p(D). \end{aligned}$$

Hence the optimization problem becomes

$$\min_{q \in Q} \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})) - \mathbb{E}_q[\log p(D|\boldsymbol{x})],$$

or, equivalently,

$$\max_{q \in Q} \mathbb{E}_q[\log p(D|\boldsymbol{x})] - \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})),$$

which is cold the Evidence Lower Bound (ELBO), since

$$\log p(D) \geq \mathbb{E}_q[\log p(D|\boldsymbol{x})] - \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})),$$

since KL is non-negative. Under the assumption that the data D is independent, the likelihood term $\mathbb{E}_q[\log p(D|\boldsymbol{x})]$ can be rewritten as an expectation of a sum of data points. Optimization of the ELBO,

$$\min_{v \in \Omega} \mathcal{L}(v) = \min_{v \in \Omega} \mathbb{E}_q[\log p(D|\boldsymbol{x})] - \mathrm{KL}(q(\boldsymbol{x}) \mid\mid p(\boldsymbol{x})),$$

can be done using gradient descent in the parameters space Ω of a distributional family Q, and using Monte-Carlo methods to sample from q_v to approximate the expectations.

4 Learning Object Embeddings from Triplet Comparisons

In this Chapter, we describe a general method of learning object embeddings via finding the maximum likelihood estimator (MLE) for a given set of triplet comparisons and a given probabilistic model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$. The learned embedding serves two purposes. First, it can be used to argue how well a given probabilistic triplet model is reflecting real world triplet comparison data through measuring the "goodness of fit" of that embedding. Second, a learned embedding is an essential part of our LEARN2SEARCH algorithm that is our proposed solution in the blind setting.

We will justify our two new models, Probit and γ-CKL, via comparing the qualities of the embeddings learned for Probit and γ-CKL versus the qualities of the embeddings learned for the state-of-the-art models t-STE and CKL on real-world datasets. Then we will introduce a distributional embedding method that captures the uncertainties in the exact positions of the objects in $\mathbb{R}^d$ given the comparison data, which can further improve the quality of the searches in the LEARN2SEARCH framework in the next Chapters.

4.1 Learning Embedding via Maximum Likelihood Estimator

Suppose we are given a set of objects $[n] = \{1, 2, \dots, n\}$, known to have representations $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ in a hidden feature space. Although we do not have access to $\boldsymbol{X}$, we observe a set of noisy triplet-based relative similarities of these objects:

$$\mathcal{T} = \{(i, j; t) \mid \text{object } i \text{ is closer to } t \text{ than } j \text{ is}\},$$

obtained with respect to an unknown probabilistic model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$, $i, j, t \in [n]$. Our goal is to find an embedding $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times \hat{d}}$ that maximizes the probabilities of the observed triplet comparisons $\mathcal{T}$ assuming probabilistic model $\hat{p}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$. This can be done by maximizing the negative log-likelihood over $\mathcal{T}$:

$$\hat{\boldsymbol{X}} := \operatorname*{arg\,max}_{\boldsymbol{X}\in\mathbb{R}^{n\times\hat{d}}} \sum_{(i,j;t)\in\mathcal{T}} -\log \hat{p}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t). \tag{4.1}$$

This optimization problem is usually solved using (stochastic) gradient descent. In order to prevent overfitting, an additional regularization term (e.g. an L_2 norm) for $\hat{\boldsymbol{X}}$ is added:

$$\hat{\boldsymbol{X}} := \operatorname*{arg\,max}_{\boldsymbol{X}\in\mathbb{R}^{n\times\hat{d}}} \sum_{(i,j;t)\in\mathcal{T}} -\log \hat{p}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t) + \lambda\|\boldsymbol{X}\|_L. \tag{4.2}$$

Thus, in this embedding method, we aim to find objects' representations $\hat{\boldsymbol{X}}$ in $\mathbb{R}^{n\times\hat{d}}$, as the original dimensionality d is also unobservable, such that the set of triplet observations $\mathcal{T}$ is explained as much as possible under assumed probabilistic model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$. Then different choices of $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$ can be compared on the basis of goodness of fit using various metrics, e.g. accuracy:

$$\operatorname{acc}(\hat{\boldsymbol{X}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(i,j;t)\in\mathcal{T}} \mathbb{1}\{||\hat{\boldsymbol{x}}_i - \boldsymbol{x}_t|| < ||\hat{\boldsymbol{x}}_j - \hat{\boldsymbol{x}}_t||\}$$

or simply average negative log-likelihood

$$\operatorname{NLL}(\hat{\boldsymbol{X}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(i,j;t)\in\mathcal{T}} -\log \hat{p}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j; \hat{\boldsymbol{x}}_t).$$

4.2 Models Performance

In order to argue about how well the proposed Probit and γ-CKL models reflect the actual human choices, we compare the performance of the embeddings, learned via solving (4.1)-(4.2), for Probit, γ-CKL, t-STE, and CKL, in terms of the accuracy and the log-likelihood metrics, on three real-world datasets.

Food. Wilber et al. [55] collected $|\mathcal{T}| = 190\text{'}376$ triplet comparisons of food dishes for $n = 100$ unique pictures of food dishes. In this dataset, each triplet $(i, j; t)$ corresponds to the respondent's answer "The dish i tastes more similar to the dish t than to the dish j".

Musical Artists. Ellis et al. [18] conducted a web-based survey in which the human participants were asked to chose the most similar musical artist among several presented options to a given target artist. The number of possible answer options varied in each experiment. Following [54], we convert the comparisons of the form $(i, Q; t)$ to triplet comparisons $\{(i, j; t)\}_{j\in Q, j\neq i}$. In total 1’032 users participated in that survey and 22’310 triplets were collected. After cleaning the data, similarly to [54], we have left with $n = 400$ artists and $|\mathcal{T}| = 9’107$ triplets.

Movie Actors. In [14] Chumbalov et al. built a web-based application for searching for famous movie actors via comparing their face pictures. They collected a dataset of a total of $|\mathcal{T}| = 50’026$ triplets for $n = 552$ unique movie actors. The queries during the searches were posed in the form of “Which actors among the given set of actors Q is the most similar to your target” with $|Q| = 4$, and the target actors were either chosen for the users prior the searches or were later revealed by the casual users of their web service after the search was completed and the target was found. We decompose their comparison outcomes $\{(i, Q; t)\}$ into triplet comparisons similarly as for the Musical Artists dataset.

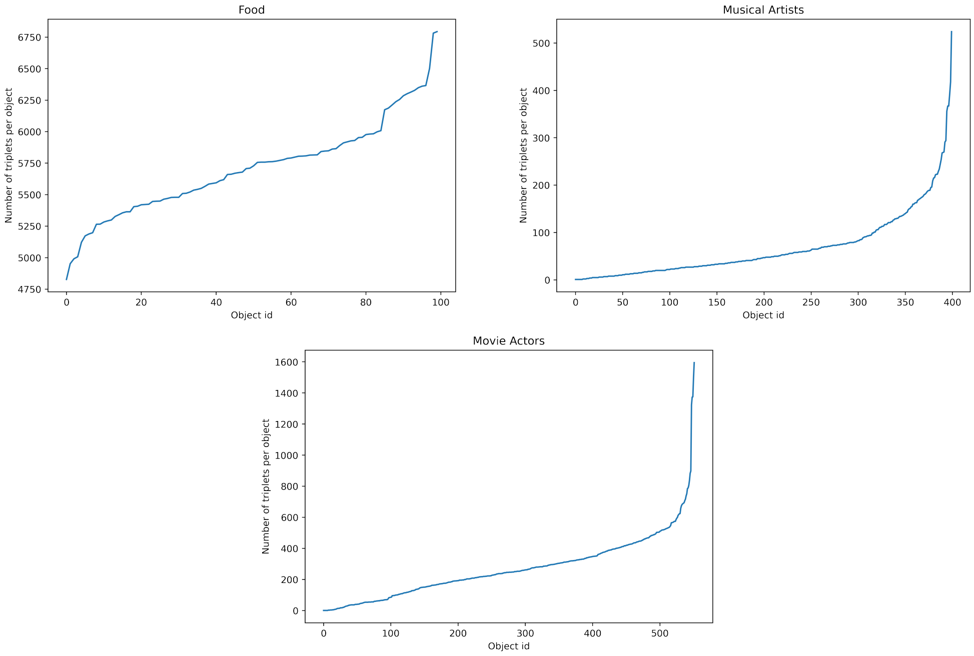


Figure 4.1: Distribution of the number of triplets per object, $m(\text{object})$ for Food, Musical Artists, and Movie Actors datasets. The gap in the triplet participation is especially pronounced for the latter two datasets.

To get a better sense of the datasets, let $m(i)$ be the number of triplets in $\mathcal{T}$ the object i appears in (either as a target or as an alternative). We plot the distribution of $m(\cdot)$ for each of the dataset in Fig. 4.1. For Musical Artists and Movie Actors the distribution is fat-tailed and this is due to the way the data was collected. Indeed, in [18] the number of alternatives in the queries to the user varied, hence the amount of triplets collected from a single user answer could be different for different artists. In [14], the triplet comparisons were collected from the searches done on a web-site designed by the authors. The searches were performed by both participants of the project as well as random users from the web. There was no full control over the targets users were searching for and some of the actors were being added and removed over time. Hence the final triplet participation distribution across the actors was skewed.

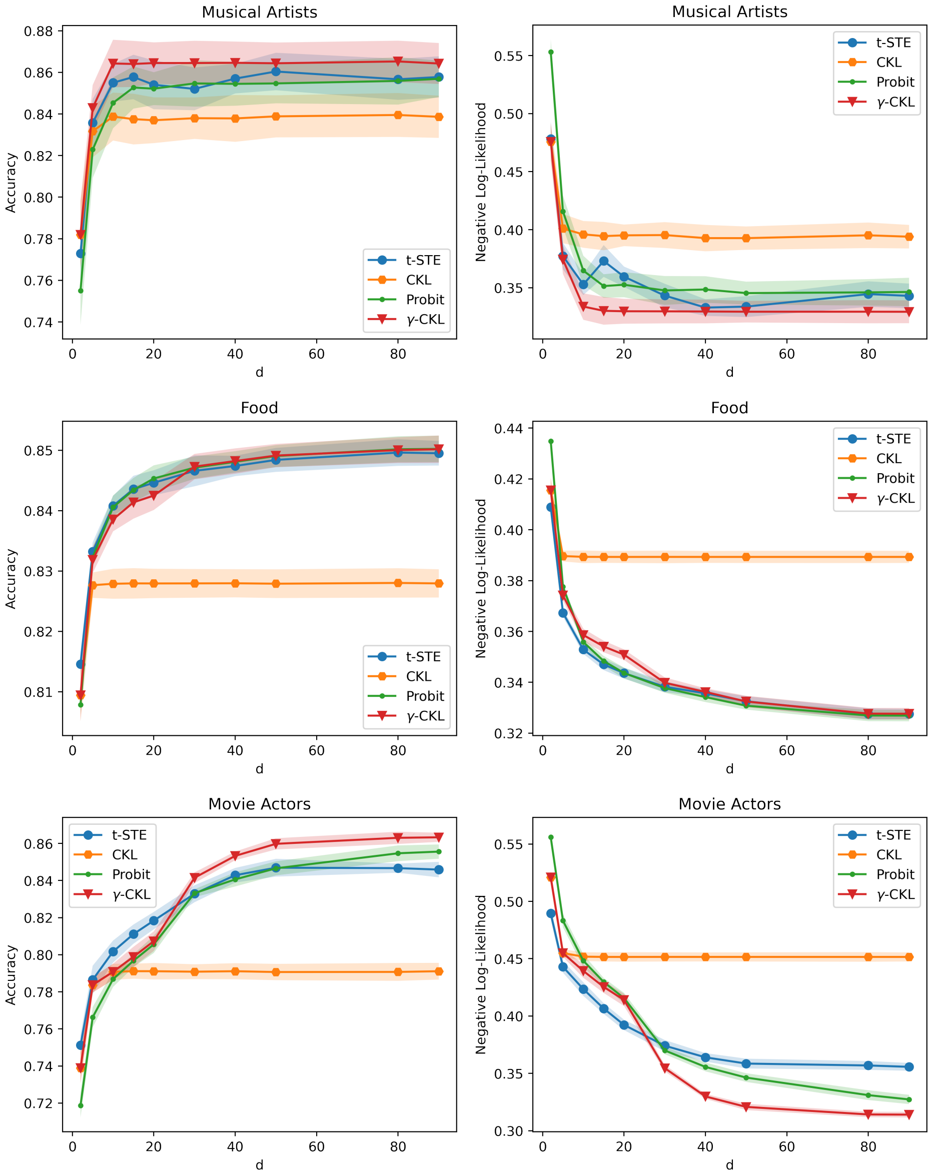


Figure 4.2: Comparison of the quality of the embedding produced using different probabilistic models. The results are reported on a 10-fold holdout triplet set for accuracy and negative log-likelihood, including mean estimates with 3 standard error bars. The proposed Probit and γ-CKL models either beat or are on par with the state-of-the-art oracle models.

Following [47], for each dataset we learned an embedding for every model using different number of dimensions $d \in [2,5,10,15,20,30,40,50,80,90]$, since the true dimensions of the features spaces in the datasets are unobserved. Similarly to [47], we performed a 10-fold cross validation to compute the average accuracy and negative log-likelihood on holdout sets. In particular, we randomly divided each original dataset into 10 equal sized subsets. Then, for each subset i, $i = 1,2,\dots,10$, we used it as a hold-out test set and the rest 9 were used to train the models. When reporting the accuracy and negative log-likelihood results in Fig. 4.2, we averaged them over 10 fold runs to get the mean estimates and standard errors.

The hyperparameters for each model were optimized and the best performing configurations for each d are reported. We used the following grid of hyperparameters: $lr \in [1e-2, 1e-3, 1e-4, 1e-5]$, $batchsize \in [128, 256, 512, |\mathcal{T}|]$, L_2 regularizer $\lambda \in [0, 0.4, 1]$, $\sigma_\varepsilon \in [0.4, 0.3, 0.2, 0.1, 0.01]$, $\gamma \in [2,3,5,10,15,20,25]$. No regularization was used for CKL and γ-CKL because these models are scale-free.

The best hyperparemeter configuration for each dataset are given below:

- **Musical Artists**
  - t-STE (accuracy 86%, nll 0.333): $D = 50$, $lr = 1e-4$, $\lambda = 0$, $batchsize = |\mathcal{T}|$
  - CKL (accuracy 83.9%, nll 0.395): $D = 80$, $lr = 1e-5$, $batchsize = 256$
  - Probit (accuracy 85.6%, nll 0.352): $D = 90$, $lr = 1e{-}2$, $\lambda = 0.4$, $\sigma_\varepsilon = 0.1$, $batchsize = 512$
  - γ-CKL (accuracy 86.5%, nll 0.329): $D = 80$, $lr = 1e-3$, $\gamma = 5$, $batchsize = |\mathcal{T}|$
- **Food**
  - t-STE (accuracy 84.9%, nll 0.327): $D = 80$, $lr = 1e-2$, $\lambda = 0$, $batchsize = |\mathcal{T}|$
  - CKL (accuracy 82.8%, nll 0.389): $D = 80$, $lr = 1e-3$, $batchsize = |\mathcal{T}|$
  - Probit (accuracy 85%, nll 0.327): $D = 90$, $lr = 1e{-}2$, $\lambda = 1.0$, $\sigma_\varepsilon = 0.01$, $batchsize = |\mathcal{T}|$
  - γ-CKL (accuracy 85.%, nll 0.327): $D = 90$, $lr = 1e-4$, $\gamma = 25$, $batchsize = |\mathcal{T}|$
- **Movie Actors**
  - t-STE (accuracy 84.6%, nll 0.4): $D = 50$, $lr = 1e-2$, $\lambda = 0$, $batchsize = 512$
  - CKL (accuracy 79.1%, nll 0.452): $D = 15$, $lr = 1e-2$, $batchsize = 512$
  - Probit (accuracy 85%, nll 0.327): $D = 90$, $lr = 1e-4$, $\lambda = 0,$, $\sigma_\varepsilon = 0.2$, $batchsize = |\mathcal{T}|$
  - γ-CKL (accuracy 86.3%, nll 0.314): $D = 90$, $lr = 1e-3$, $\gamma = 20$, $batchsize = |\mathcal{T}|$

Overall, across the three datasets, Probit and γ-CKL correctly model between 84% and 86% of triplets. On Musical Artists and Food, Probit and γ-CKL are on par with t-STE and significantly outperform CKL. On Movie Actors dataset, Probit and γ-CKL outperform their competitors. We can see that the γ-CKL model immediately benefits from having a general γ parameter already in small dimensions compared to the original CKL.

We also found out that increasing d benefits the quality of the learned embedding for Probit and γ-CKL, but not necessarily for t-STE and CKL; for γ-CKL as d increases, the best performing values of γ tend to also increase, which is aligned with the findings of Theorem 1, see Fig. 4.3. For each dataset and each value of d we report the running average of the mean of the top 10 best performing values of γ for that d. We see that the running average value of γ is uniformly lower for for the Musical Artists dataset than for the other two datasets. We suspect this is because the dataset itself contains relatively small average number of triplets per object. That is why the γ-CKL embedding does not profit from increasing the values of γ, which could lead the model to be more confident when predicting outcome probabilities.

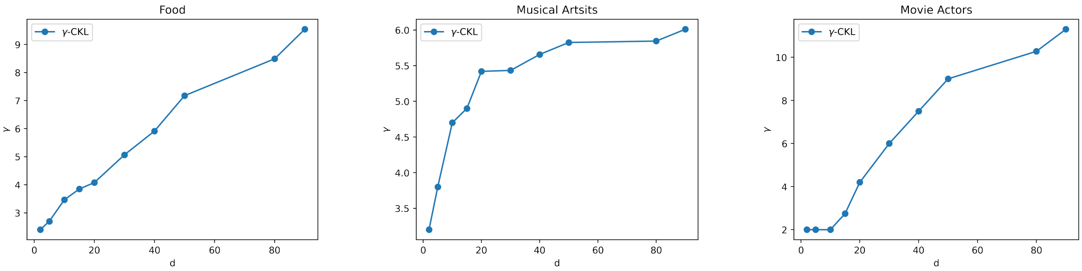


Figure 4.3: Running average of the best performing values of γ in γ-CKL embedding as we increase the embedding dimensionality d.

In our experiments we did not notice any strong sensitivity to any of the models to the hyperparameters for the same dataset, apart from the dimensionality of the embedding D. The only exception is γ-CKL for which the best configurations of hyperparameters usually tend to have all the same values of γ, which usually increase with D, as discussed above.

Overall, we can conclude that the new proposed oracle models, Probit and γ-CKL, reflect the real user behaviour on the comparison-like tasks very well.

4.3 Estimating d via SVD

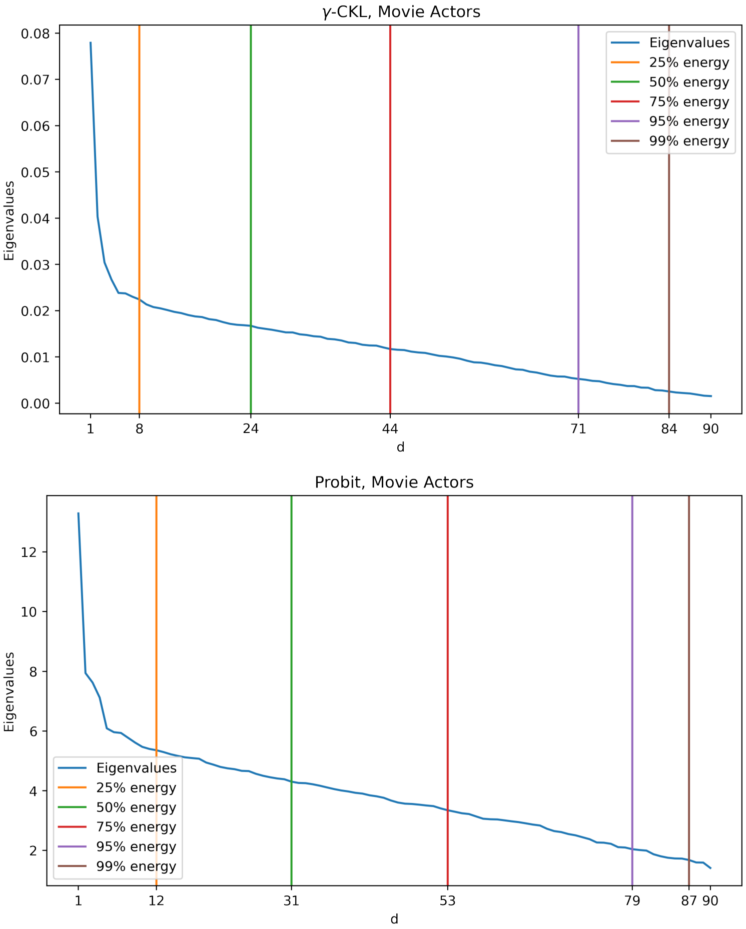


Figure 4.4: The % of the energy explained by the dimensionality values. The dimensionality of the initial embedding was chosen as $\hat{d} = 90$. For the γ-CKL model, $\hat{d} = 84$ has 99% of the energy, for the Probit model, $\hat{d} = 87$ has 99% of the energy.

One way to choose the dimensionality of the embedding is to perform an exhaustive cross-validation over different values of $\hat{d}$, which could be computationally-heavy in some cases. An alternative and more practical approach is to first learn an embedding for some big value of d, and then examine the spectrum of the learned embedding matrix $\hat{\boldsymbol{X}}$, see Fig. 4.4.

Given an initial embedding matrix $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times \hat{d}}$, the dimensionality that contains $\%k$ of energy is defined as the smallest $d \leq \hat{d}$ s.t.

$$\frac{\sum_i^d \sigma_i}{\sum_i^{\hat{d}} \sigma_i} \leq \frac{k}{100},$$

where $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_{\hat{d}}$ are the singular values of $\hat{\boldsymbol{X}}$.

Fig. 4.4 suggests that most of the dimensions in the embedding are useful and higher values of $\hat{d}$ are required to carry 99% of the energy, given by the singular values of the embedding matrix. This is aligned with the findings in Fig. 4.2, where the most accurate embeddings tend to be high-dimensional.

4.4 Embedding Visualization

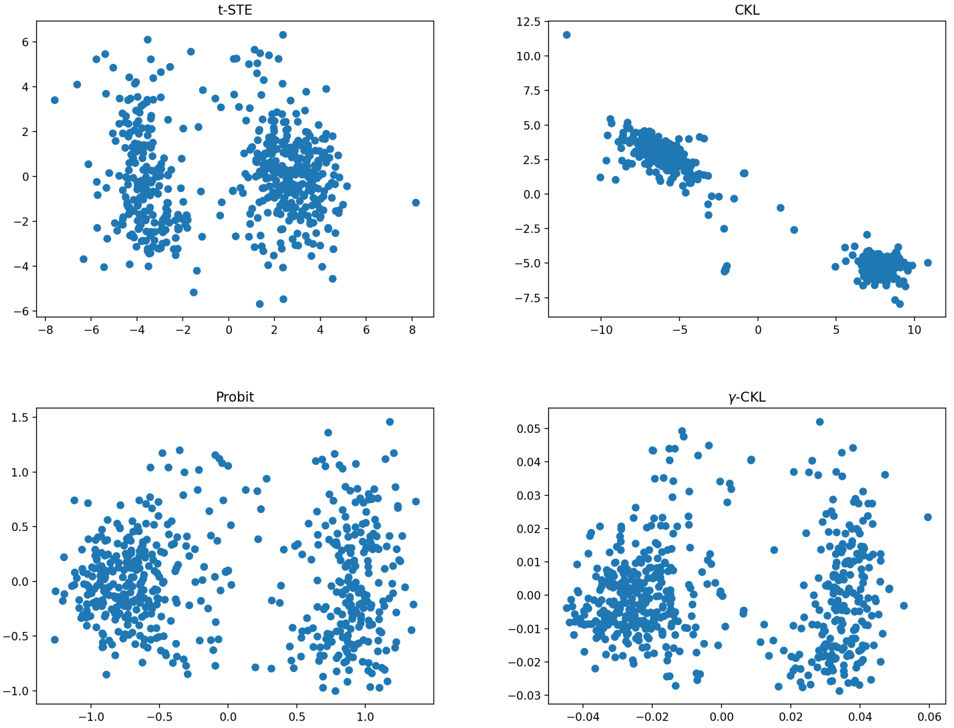


Figure 4.5: 2D PCA of the embeddings of the Movie Actors dataset using different probabilistic oracle models.

In Fig. 4.5. we visualize in 2D the embeddings produced by different oracle models for the Movie Actors dataset (in total, slightly more than 50’000 triplets), using PCA. All models managed to separate actors based on gender into two clusters. In Fig. 4.6 (Probit) and Fig. 4.7 (γ-CKL), we can also examine what the 2 main components represent: the x-axis clearly separates male and female actors; male actors that are closer to the female cluster are younger, female actors that are closer to the male cluster have short hair and appear more masculine.

The y-axis separates skin color: we see actors with darker skin on one side of the spectrum, and actors with lighter skin on the other. Actors that have similar facial features also appear to have similar representations. The embedding methods did not have access to the actual pictures of the actors faces, and the produced embeddings are based exclusively on the triplet comparisons collected via crowdsourcing. This suggests, that when humans compare faces, the first two most important features they look at is gender and skin color.

Overall, both our models did a very good job in embedding the actors, resulting in high quality representations; this is coherent with the quantitative results presented in Fig. 4.2.

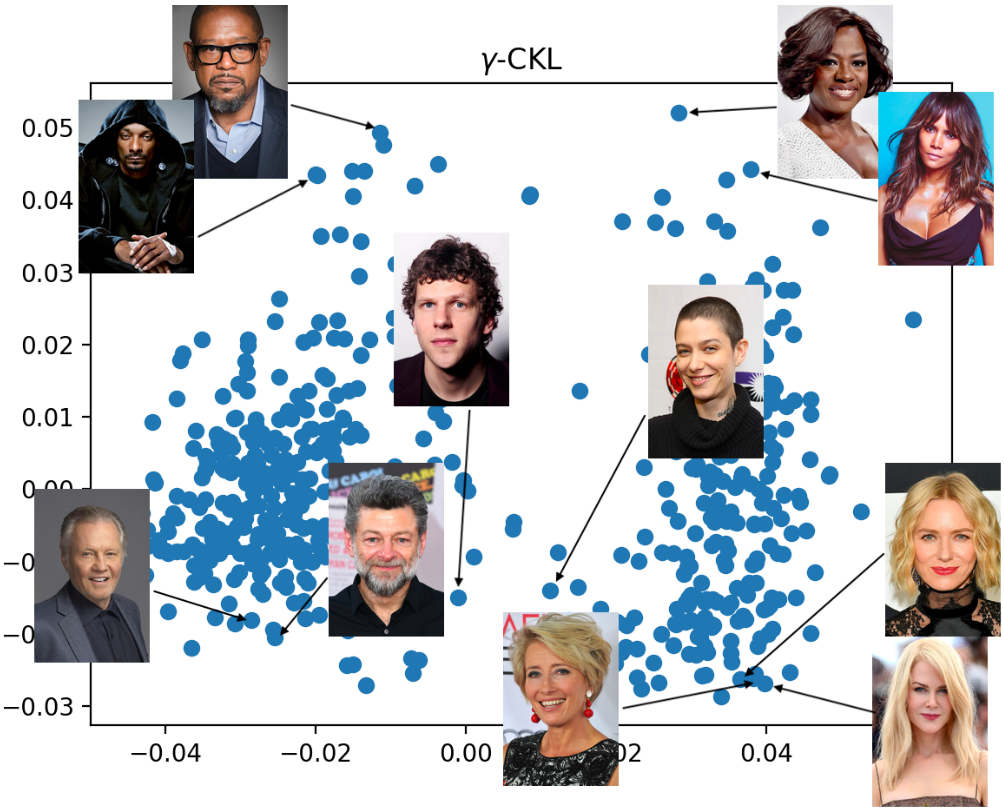


Figure 4.6: 2-PCA on learned embedding with the γ-CKL model for the Movie Actors dataset.

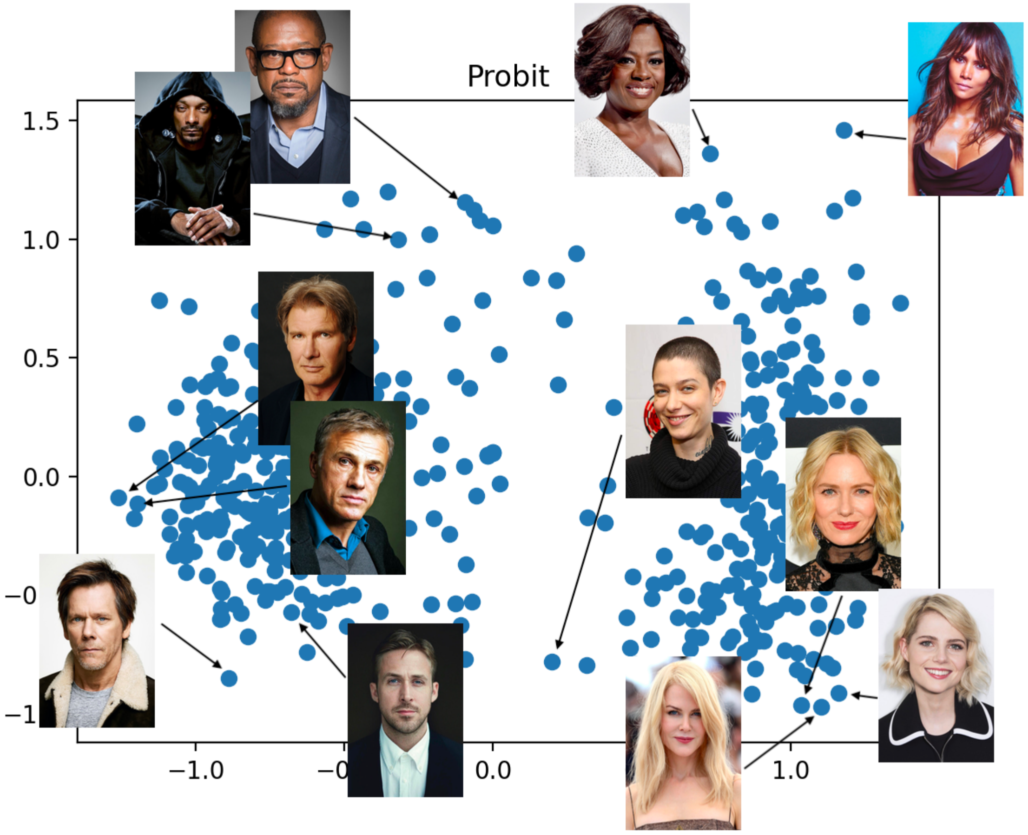


Figure 4.7: 2-PCA on learned embedding with the Probit model for the Movie Actors dataset.

4.5 Variational Embedding

In this last section we introduce a new general embedding method that produces distributional embedding, which, as we will see later, will be very useful in symbiosis with search methods in the LEARN2SEARCH framework.

In a given dataset of triplet comparison outcomes, one object might be included in few triplets while another might be included in many (e.g., due to the latter being a popular search target), thus we could have more information about one object and less information about another. Indeed, the distributions of the number the triplets per target in the real world datasets could be far from uniform as we have seen in Fig. 4.1 in Section 4.2.

This suggests that when approximating $\boldsymbol{X}$, an *uncertainty* about the object's position should be taken into account. The latest should also affect the operation of the search algorithm: After observing the outcome of oracle's comparison between two objects i and j, whose positions $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ in our estimated embedding space are noisy, the target posterior distribution should have higher entropy than if $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ were exact (as we had assumed so far).

We incorporate this dependence by generating a distributional embedding, where each estimate $\hat{\boldsymbol{x}}_i$ is endowed with a distribution, thus capturing the uncertainty about its true position $\boldsymbol{x}_i$. We take a Bayesian viewpoint and aim to find an approximate posterior distribution over the objects' embedding, given the triplets $\mathcal{T}$. For computational tractability, we restrict ourselves to an mean-field approximate posterior, where $p(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. The goal is to learn a d-dimensional Gaussian distribution embedding

$$q(\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \dots, \hat{\boldsymbol{x}}_n) := \prod_{i=1}^{n} \mathcal{N}(\hat{\boldsymbol{x}}_i; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^d$, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d\times d}$ and diagonal, such that objects that are similar with respect to their "true" representations $\{\boldsymbol{x}_i\}_{i=1}^{n}$ (reflected by $\mathcal{T}$) are also similar in the learned embedding. Here for each object, $\boldsymbol{\mu}_i$ represents the mean guess on the location of object i in $\mathbb{R}^d$, and $\boldsymbol{\Sigma}_i$ represents the uncertainty about its position.

We learn this Gaussian embedding via maximizing the ELBO (introduced in the Technical Background chapter):

$$\mathcal{L}(\{(\mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^{n}) = \mathbb{E}_q \left[\log p(\mathcal{T}, \hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n) - \log q(\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n)\right], \tag{4.3}$$

where the joint distribution $p(\mathcal{T}, \hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n)$ is the product of the prior

$$p(\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n) = \prod_{i=1}^{n} \mathcal{N}(\hat{\boldsymbol{x}}_i; \boldsymbol{0}, \boldsymbol{I})$$

and the likelihood $p(\mathcal{T} \mid \hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n)$ that is defined via probabilistic model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$:

$$p(\mathcal{T} \mid \hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_n) = \prod_{(i,j;t)\in|\mathcal{T}|} p(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j; \hat{\boldsymbol{x}}_t).$$

We optimize the ELBO by using stochastic backpropagation with the reparameterization trick [30, 40]. A detailed algorithm is outlined in Algorithm 1. The total complexity of executing one epoch e is $O(|\mathcal{T}|d)$.

Using the resulting variational embedding in the LEARN2SEARCH framework brings a natural tradeoff between exploration and exploitation: At first, the system tends to ask queries about objects with high variance $\boldsymbol{\Sigma}_i$ more frequently, and the large effective noise reflected by this variance restricts the speed at which the search narrows down the space of the target candidates. After multiple searches, as the system gets more confident about the embedding, it begins exploiting the structure of the space more assertively. In Section 5.1.5 we will empirically demonstrate that using this variational embedding in the latent setting greatly profits the

Algorithm 1 Learning Variational Embedding

```
Input: Triplet model $p(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{x}_t)$, objects $\{1, 2, \ldots, n\}$, triplets $\mathcal{T}$, $n_s$, $E$, $\alpha$
Output: posterior distribution for each point $\{(\mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n$
Initialize $\boldsymbol{\mu}_i \leftarrow \boldsymbol{0}$, $\boldsymbol{\Sigma}_i \leftarrow \boldsymbol{I}$, $i = 1, 2, \ldots, n$
$p(\hat{\boldsymbol{x}}_i) \leftarrow \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $i = 1, 2, \ldots, n$
$q(\hat{\boldsymbol{x}}_i) \leftarrow p(\hat{\boldsymbol{x}}_i)$, $i = 1, 2, \ldots, n$
for $e = 1, 2, \ldots, E$ do
    Compute the Kullback–Leibler divergence term $D := \sum_{i=1}^n D_{\mathrm{KL}}\left(p(\hat{\boldsymbol{x}}_i) || q(\hat{\boldsymbol{x}}_i)\right)$
    For each $\boldsymbol{x}_i$ sample $\{\boldsymbol{s}_i^1, \boldsymbol{s}_i^2, \ldots, \boldsymbol{s}_i^{n_s}\}$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $i = 1, 2, \ldots, n_s$
    $\hat{\boldsymbol{s}}_i^k = \mu_i + \boldsymbol{\Sigma}_i \boldsymbol{s}_i^k$, $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, n_s$
    Compute the log-likelihood term $L := \frac{1}{n_s} \sum_{k=1}^{n_s} \sum_{(i,j;t) \in \mathcal{T}} \log p(\hat{\boldsymbol{s}}_i^k, \hat{\boldsymbol{s}}_j^k; \hat{\boldsymbol{s}}_t^k)$
    Compute the gradient $\nabla \mathcal{L} = \nabla (L - K)$ with respect to $\{(\mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n$
    $\{(\mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n \leftarrow \{(\mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n + \alpha \nabla \mathcal{L}$
    $q(\hat{\boldsymbol{x}}_i) \leftarrow \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $i = 1, 2, \ldots, n$
end for
```

search procedure in comparisons to using point-estimate embeddings based on other model, however in the scope of modeling human choices, we have already established the usefulness of the Probit model in Section 4.2, and do not expect Algorithm 1 to outperform its point-estimate Probit version in that task, since the underlying probabilistic model remains the same.

Note, that the procedure of learning a variational embedding can be also used as an alternative to the conventional way of learning through MLE with a regularization (4.2). The KL divergence term can be seen as a regularizer itself, because it forces the posterior $q(\hat{\boldsymbol{x}}_i)$ to not differ much from the prior distribution $p(\hat{\boldsymbol{x}}_i)$. In principle, the covariance matrices can be later ignored and only the means $\boldsymbol{\mu}_i$ can be used as the point estimates of $\boldsymbol{x}_i$.

5 Comparison-based Search

In this Chapter, we develop efficient comparison-based search algorithms for the Probit and the γ-CKL models. We begin by introducing an efficient Probit-based search method called GAUSSSEARCH, which (a) generates query pairs and (b) navigates towards the target. It is provably guaranteed to converge to the target, due to the specific form of the Probit model and key properties of the search algorithm. In contrast with prior adaptive search methods, where generating a query pair is typically expensive, our algorithm is also computationally efficient. We then study the performance of GAUSSSEARCH in a series of synthetic experiments in two regimes: when the objects features $\boldsymbol{X}$ are available to us during the searches and when they are hidden. In the latter we combine GAUSSSEARCH with the variational embedding method GAUSSEMBED introduced in the previous Chapter under a general search framework LEARN2SEARCH.

Next, we develop algorithms that are based on the γ-CKL model. Informally, in this model, the work required to halve the volume of the belief region does not depend on the scale of the current belief region. This suggests that there is hope that this volume can shrink exponentially with the number of queries. We propose an algorithm that provably achieves exponential convergence to the target when $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d$. The algorithm relies on the backtracking mechanism that asks non-local queries and "regrows" the belief region when the belief region looses the target in order to recapture the target. The result is non-trivial because there is always the possibility of errors, such that the current belief moves too far away from the target. Then we describe a practical implementation of this scheme and show exponential convergence over a series of experiments.

Finally, we derive a heuristic algorithm for the γ-CKL that works well for relatively small n. For this case, it is computationally feasible to maintain full belief distributions over all objects, and to choose queries close to optimally by approximately maximizing expected information gain.

5.1 Probit Model: GAUSSSEARCH

In this subsection we present a search algorithm GAUSSSEARCH assuming the Probit model (2.11). Generally, we are interested in methods that are

1. *Efficient* in the average number of queries per search, and
2. *Scalable* in the number of objects n.

Scalability requires *low computational complexity* of the procedure for choosing the next query. As the users answers are stochastic and their choices can vary from one user to another, we also require our algorithms to be *robust* to the probabilistic nature of the human feedback.

Gaussian Model. Due to the sequential and probabilistic nature of the problem, we take a Bayesian approach in order to model the uncertainty about the location of the target. In particular, we maintain a d-dimensional Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{x}}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ that reflects our current belief about the position of the user's target point $\boldsymbol{x}_t$ in $\mathbb{R}^d$, $t \in [n]$. We model user answers with the Probit likelihood (2.11), and we apply approximate inference techniques for updating $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ every time we observe a query outcome. The space requirement of the model is $O(d^2)$.

The motivation behind such a choice of parametrization is that

1. The size of the model does not depend on n, guaranteeing scalability,
2. One can characterize a general pair of points in $\mathbb{R}^d$ that maximizes the expected information gain, and
3. The sampling scheme that chooses the next pair of query points informed by 2 is simple and works extremely well in practice.

5.1.1 Choosing the Next Query

To generate the next query, we follow a classic approach from information-theoretic active learning [35]: find the query that minimizes the expected posterior uncertainty of our estimate $\hat{\boldsymbol{x}}$, as given by its entropy. More specifically, we find a pair of points $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ corresponding to the objects $i, j \in [n]$ that maximizes the expected *information gain* for our current estimation of $\boldsymbol{x}_t$ at step m:

$$I[\hat{\boldsymbol{x}}; Y \mid \{\boldsymbol{x}_i, \boldsymbol{x}_j\}] = H(\hat{\boldsymbol{x}}) - \mathbb{E}_{Y \mid \boldsymbol{x}_i, \boldsymbol{x}_j, \hat{\boldsymbol{x}}}[H(\hat{\boldsymbol{x}} \mid Y)],$$

where the random variable $\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the current belief about the target position $\boldsymbol{x}_t$ in the space $\mathbb{R}^d$, $H(\boldsymbol{x})$ is Shannon's entropy, and $Y \sim p(Y \mid \boldsymbol{x}_i, \boldsymbol{x}_j, \hat{\boldsymbol{x}})$ is the current belief over the answer to the comparison between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Exhaustively evaluating the expected information gain over all $O(n^2)$ pairs for each query (e.g., as done in [13]) is prohibitively expensive. Instead, we propose a more efficient approach, that runs in time $O(\log n)$ with $O(n \log n)$ preprocessing using a kd-tree, that is done only once for an embedding. Recall from equation (2.11) that the comparison outcome probability $p(Y \mid \hat{\boldsymbol{x}}, \boldsymbol{x}_i, \boldsymbol{x}_j)$ depends on $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ only through the corresponding bisecting normal hyperplane h_{ij}. Therefore, instead of looking for the optimal pair of points from $\mathcal{X}$, we first find the hyperplane h that maximizes the expected information gain. Following [25], it can be rewritten as:

$$\begin{aligned} I[\hat{\boldsymbol{x}}; Y \mid h] &= H(\hat{\boldsymbol{x}}) - \mathbb{E}_{Y \mid \hat{\boldsymbol{x}}, h}[H(\hat{\boldsymbol{x}} \mid Y)] \\ &= H(Y \mid \hat{\boldsymbol{x}}, h) - \mathbb{E}_{\hat{\boldsymbol{x}}}[H(Y \mid \hat{\boldsymbol{x}}, h)]. \end{aligned} \tag{5.1}$$

The following theorem gives us the key insight about the general form of a hyperplane $h = (\boldsymbol{w}, b)$ that optimizes this utility function by splitting the Gaussian distribution into two equal "halves":

Theorem 2. *Let $\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and let $\mathcal{H} = \{(\boldsymbol{w}, b) \mid \|\boldsymbol{w}\| = 1, \boldsymbol{w}^\top \boldsymbol{\mu} + b = 0\}$ be the set of all hyperplanes passing through $\boldsymbol{\mu}$. Then,*

$$\operatorname*{arg\,max}_{h \in \mathcal{H}} I[\hat{\boldsymbol{x}}; Y \mid h] = \operatorname*{arg\,max}_{(\boldsymbol{w}, b) \in \mathcal{H}} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}.$$

Proof sketch. Let $h^\star = (\boldsymbol{w}^\star, b^\star)$ be an optimal hyperplane. Since $\boldsymbol{w}^\top \boldsymbol{\mu} + b = 0$, we have $p(Y \mid \hat{\boldsymbol{x}}, h) \equiv 1/2$ and $H(Y \mid \hat{\boldsymbol{x}}, h) \equiv 1$. Thus, $h^\star$ is such that $\mathbb{E}_{\hat{\boldsymbol{x}}}[H(Y \mid \hat{\boldsymbol{x}}, h)]$ is minimal. We show that

$$\mathbb{E}_{\hat{\boldsymbol{x}}}[H(Y \mid \hat{\boldsymbol{x}}, h)] = \mathbb{E}_z[H(\Phi(z))],$$

where $z \sim \mathcal{N}(0, \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w})$. As the binary entropy function has a single maximum in 1/2, the expectation on the r.h.s. is minimized when the variance of z is maximized. □

In other words, Theorem 2 states that any optimal hyperplane is orthogonal to a direction that maximizes the variance of $\hat{\boldsymbol{x}}$, i.e. its first eigenvector.

Consequently, we find a hyperplane h that maximizes the expected information gain via an eigenvalue decomposition of $\boldsymbol{\Sigma}$. However, we still need to find a pair of objects $\{i, j\}$ from $[n]$ to form the next query. We propose a sampling strategy based on Theorem 2, which we describe in Algorithm 2. We maintain a set of points $\mathcal{U} \subseteq [n]$ that were used in any of the queries during the search and do not repeat queries containing objects the user have seen so far, since they were not indicated as targets by the user. After computing the optimal hyperplane $h^\star$, we sample a point $\boldsymbol{z}_1$ from the current Gaussian belief $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ over the location of the target, and reflect it across $h^\star$ to give the second point $\boldsymbol{z}_2$. These two points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ span $h^\star$, but are not guaranteed to be in $\mathcal{X}$. Then we construct our query by finding the closest pair of (distinct and yet unseen) points $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ to $\{\boldsymbol{z}_1, \boldsymbol{z}_2\}$ from $\mathcal{X}$. The hyperplane spanned by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ now approximates the optimal hyperplane $h^\star$ that maximizes the expected information gain.

Assuming that the feature vectors $\mathcal{X}$ are organized into a k-d tree [5] at a one-time computational cost of $O(n \log n)$ for a given $\mathcal{X}$, Algorithm 2 runs in time $O(\log n + d^3)$. This includes $O(d^3)$ for the eigenvalue decomposition, $O(d^2)$ for sampling from a d-dimensional Gaussian distribution, $O(d)$ for mirroring the sample, and $O(\log n)$ on average for finding the closest pair of points from the dataset using the k-d tree.

If needed, computing the eigenvector that corresponds to the largest eigenvalue can be well approximated via the power method with $O(d^2)$ complexity (assuming a constant number of iterations). This will bring down the total complexity of Algorithm 2 to $O(\log n + d^2)$.

Algorithm 2 SAMPLEMIRROR

```
Input: current belief $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\mathcal{X}, \mathcal{U}$
Output: query pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$
1: Compute optimal hyperplane $h^\star$ for $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
2: Sample a point $\boldsymbol{z}_1$ from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
3: Obtain $\boldsymbol{z}_2$, as the reflection of $\boldsymbol{z}_1$ across $h^\star$.
4: Find objects $i$ and $j$, s.t. $i \neq j$ and $i, j \notin \mathcal{U}$, with the closest representations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ to $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively.
```

Note that the actual hyperplane defined by the obtained pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ in Algorithm 2 does not, in general, coincide with the optimal one. Nevertheless, the hyperplane h_{ij} approximates $h^\star$ increasingly better as n grows. For large enough n we could make a reasonable assumption of high density of points $\{\boldsymbol{x}_i\}_{i=1}^n$ in $\mathbb{R}^d$, and hence expect $\boldsymbol{h}_{i,j}$ to be a good enough approximation for $\hat{\boldsymbol{h}}$.

5.1.2 Updating the Model

Finally, after querying the pair $\{i, j\}$ and observing outcome $\bar{y}$, we update our belief on the location of target $\hat{\boldsymbol{x}}$. Suppose that we are at the m-th step of the search, and denote the belief before observing $\bar{y}$ by $p_{m-1}(\hat{\boldsymbol{x}}) = \mathcal{N}(\hat{\boldsymbol{x}}; \boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1})$. Ideally, we would like to update it using

Bayesian inference:

$$p^\star_m(\hat{\boldsymbol{x}}) \propto p(Y = \bar{y} \mid Q = \{\boldsymbol{x}_i, \boldsymbol{x}_j\}, \hat{\boldsymbol{x}}) \cdot p_{m-1}(\hat{\boldsymbol{x}}).$$

However, this distribution is no longer Gaussian. Therefore, we approximate it by the "closest" Gaussian distribution $p_m(\hat{\boldsymbol{x}}) := \mathcal{N}(\hat{\boldsymbol{x}}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. We use a method known as *Expectation-Propagation* [36], which solves the following program:

$$\min_{\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m} D_{\mathrm{KL}}\left[p^\star_m(\hat{\boldsymbol{x}}) \parallel p_m(\hat{\boldsymbol{x}})\right],$$

where $D_{\mathrm{KL}}[p \| q]$ is the Kullback-Leibler divergence from p to q. Because of the form of the Probit model, this can be done in closed form by computing the first two moments of the distribution $p^\star_m$, with running time $O(d^2)$, see full derivations in the Technical Background chapter. Formally, at each step of the search, we perform computations outlined in Algorithm 3.

Algorithm 3 Update

```
Input: $\boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1}, (\boldsymbol{w}, b)$
Output: $\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m$
  $\mu_{\mathrm{proj}} \leftarrow \boldsymbol{\mu}_{m-1}^\top \boldsymbol{w} + b$
  $\alpha \leftarrow \frac{\partial}{\partial \mu_{\mathrm{proj}}} \log \Phi(\mu_{\mathrm{proj}}, \boldsymbol{w} \boldsymbol{\Sigma}_m \boldsymbol{w}^\top + \sigma_\varepsilon)$
  $\beta \leftarrow \frac{\partial^2}{\partial \mu_{\mathrm{proj}}^2} \log \Phi(\mu_{\mathrm{proj}}, \boldsymbol{w} \boldsymbol{\Sigma}_m \boldsymbol{w}^\top + \sigma_\varepsilon)$
  $\nu \leftarrow \frac{\alpha - \beta \mu_{\mathrm{proj}}}{1 + \beta \boldsymbol{w}^\top \boldsymbol{\Sigma}_{m-1} \boldsymbol{w}}$
  $\tau \leftarrow \frac{-\beta}{1 + \beta \boldsymbol{w}^\top \boldsymbol{\Sigma}_{m-1} \boldsymbol{w}}$
  $\boldsymbol{\Sigma}_m \leftarrow \left(\boldsymbol{\Sigma}_{m-1}^{-1} + \tau \boldsymbol{w} \boldsymbol{w}^\top\right)^{-1}$
  $\boldsymbol{\mu}_m \leftarrow \boldsymbol{\Sigma}_m [\boldsymbol{\Sigma}_{m-1}^{-1} \boldsymbol{\mu}_{m-1} + (\nu - b\tau) \boldsymbol{w}]$
```

We summarize the full active search procedure in Algorithm 4.

Algorithm 4 GAUSSSEARCH

Input: Objects $[n]$, feature vectors $\mathcal{X}$.

Output: target object t

```
Initialize $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$
$\mathcal{U} \leftarrow \emptyset$
$m \leftarrow 1$
repeat
    $\boldsymbol{x}_i, \boldsymbol{x}_j \leftarrow$ SAMPLEMIRROR($\boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1}, \mathcal{X}, \mathcal{U}$)
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{i, j\}$
    $\bar{y} \leftarrow$ noisy comparison outcome as per (2.11)
    $\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m \leftarrow$ UPDATE($\boldsymbol{x}_i, \boldsymbol{x}_j, \bar{y}, \boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1}$)
    $m \leftarrow m + 1$
until $t \in \{i, j\}$
```

5.1.3 Convergence of GAUSSSEARCH

For finite n, GAUSSSEARCH is guaranteed to terminate because objects are used in queries without repetition. However, in a scenario where n is effectively infinite because the feature space is dense, i.e., $\mathcal{X} = \mathbb{R}^d$, we are able to show a much stronger result. The following theorem asserts that, for any initial Gaussian prior distribution over the target with full-rank covariance matrix, the GAUSSSEARCH posterior asymptotically concentrates at the target.

Theorem 3. *Assuming Probit oracle model (2.11) and letting $\mathcal{X} = \mathbb{R}^d$, for any initial $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0 \succ 0$, Algorithm 4 satisfies*

1. $\mathrm{Tr}(\boldsymbol{\Sigma}_m) \to 0$,
2. $\boldsymbol{\mu}_m \to \boldsymbol{x}_t$

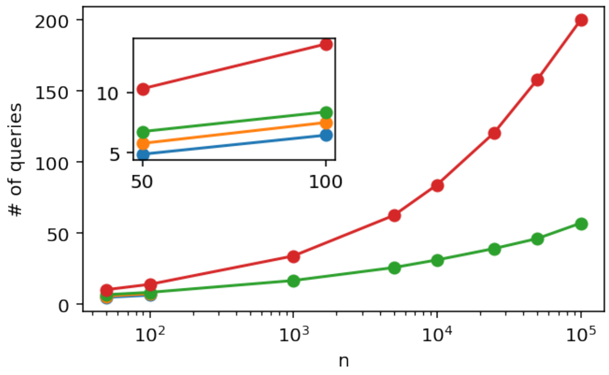
as $m \to \infty$ almost surely.

Proof sketch. We obtain crucial insights on the asymptotic behavior by studying the case $d = 1$, when $\boldsymbol{x}_t$, $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ become scalars x_t, μ_m and σ_m^2, respectively. We first show that $\sigma_m^2 = \Theta(1/m)$. Next, we are able to recast $\{\mu_m\}_{m=0}^{\infty}$ as a random walk biased towards x_t, with step size σ_m^2. By drawing on results from stochastic approximation theory [41, 6], we can show that $\mu_m \to x_t$ almost surely. The extension to $d > 1$ follows by noticing that, in the dense case $\mathcal{X} = \mathbb{R}^d$, we can assume (without loss of generality) that the search sequentially iterates over each dimension in a round-robin fashion. □

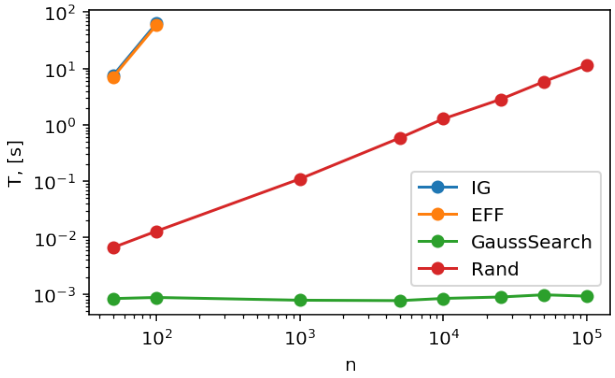
This result is not a priori obvious, especially in light of the Probit model: answers become coin flips as the sample points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ get closer. Nevertheless, Theorem 3 guarantees that the algorithm continues making progress towards the target.

5.1.4 Experimental Results: Non-latent Setting

We conducted an experiment in order to compare the query complexity and computational performance of GAUSSSEARCH and the state-of-the-art comparison-based search algorithms under the assumption that the feature vectors $\mathcal{X}$ are accessible.



(a) Query complexity



(b) Computational complexity

Figure 5.1: Average (a) query complexity and (b) computational complexity of four search algorithms in the non-latent setting for $d = 5$ and increasing n. Search success is declared when $\arg\max_{i\in[n]} p_i = t$.

The main difference between GAUSSSEARCH and previous methods is the way the uncertainty about the target object is modeled. All previous approaches consider a discrete posterior distribution $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ over all objects from $[n]$, which is updated after each step using

Bayes' rule. We compared GAUSSSEARCH to the baselines that operate on the full discrete distribution and that use different rules to choose the next query: (1) IG: the next query (i, j) is chosen to maximize the expected information gain, (2) EFF: a fast approximation of the EC^2 active learning algorithm [22] and (3) RAND: the pair of query points is chosen uniformly at random from $[n]$.

In order to assess how well GAUSSSEARCH scales in terms of both query and computational complexities in comparison to the other baselines, we run simulations on synthetically generated data of n points uniformly sampled from a $d = 5$ dimensional hypercube. We vary the number of points n from 50 to 100000. For each value of n, we run 1000 searches, each with a new target sampled independently and uniformly at random among all objects. During the search, comparison outcomes are sampled from the Probit model (2.11), using a value of σ_ε^2 chosen such that approximately 10% of the queries' outcomes are flipped. In order not to give an undue advantage to our algorithm (which never uses an object in a comparison pair more than once), we change the stopping criterion, and declare that a search is successful as soon as the target t becomes the point with the highest probability mass under a given method's target model P (ties are broken at random). For the GAUSSSEARCH procedure, we take p_i to be proportional to the density of the Gaussian posterior at $\boldsymbol{x}_i$. We use two performance metrics.

Query complexity The average number of queries until $\arg\max_{i \in [n]} p_i = t$, i.e., the true target point has the highest posterior probability.

Computational complexity The (relative) total time needed for an algorithm to decide which query to make next and to update the posterior upon receiving a comparison outcome.

Fig. 5.1a and Fig. 5.1b show the results averaged over 1000 experiments. Both IG and EFF require $O(n^3)$ operations to choose the next query, as they perform a greedy search over all possible combinations of query pair and target, and for each pair need to update the full posterior on n points. For these two methods, we report results only for $n = 50$ and $n = 100$, as the time required for finding the optimal query and updating the posterior for $n > 100$ takes over a minute per one step. For $n \in \{50, 100\}$, IG performs best in terms of query complexity, with 4.9 and 6.47 queries per search on average for $n = 50$ and $n = 100$, respectively. However, the running time is prohibitively long already for $n = 100$. In contrast, GAUSSSEARCH has a favorable tradeoff between query complexity and computational complexity: in comparison to IG, it requires only 1.8 (for $n = 50$) or 2 (for $n = 100$) additional queries on average, while the running time is improved by several orders of magnitude. We observe that, even though in theory the running time of GAUSSSEARCH is not independent of n, in practice it is almost constant throughout the range of values of n that we investigate. Finally, we note that RAND performs worst in terms of average number of queries, despite having a higher running time per search step (due to maintaining a discrete posterior). We illustrate the progress of GAUSSSEARCH in Fig. 5.2-5.4.

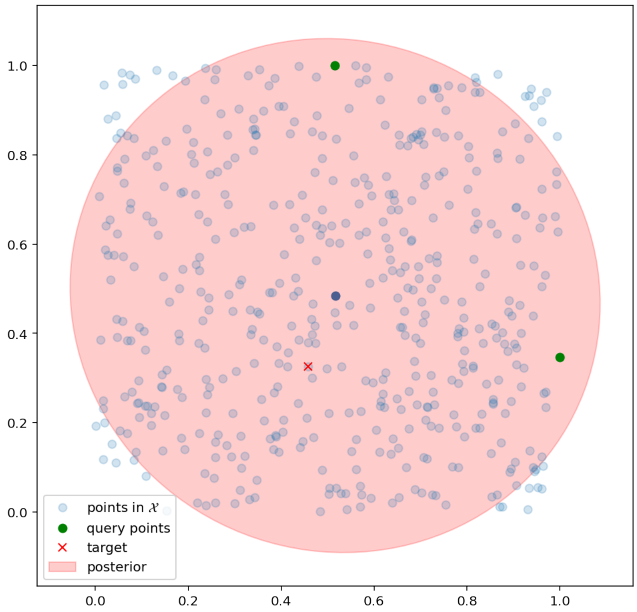


Figure 5.2: Vizualization of GAUSSSEARCH progress. Beginning stage. The Gaussian posterior covers most of the points reflecting the uncertainty in the early stages.

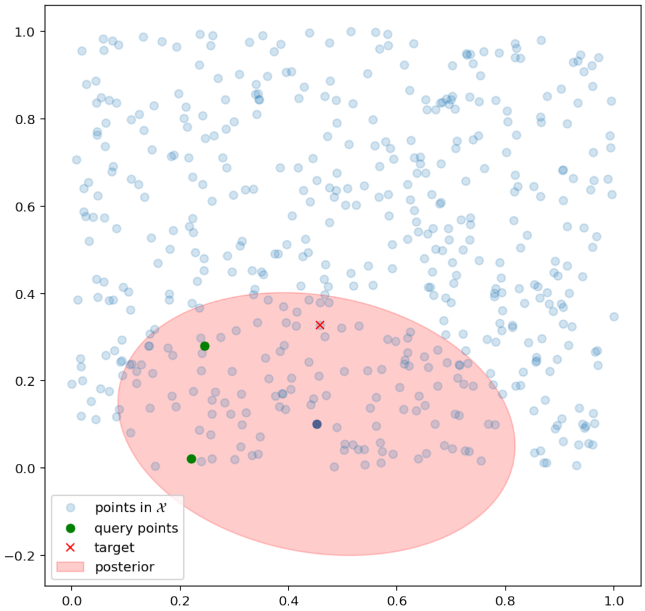


Figure 5.3: Vizualization of GAUSSSEARCH progress. Middle of the search. The Gaussian posterior shrinks and begins to concentrate around the target.

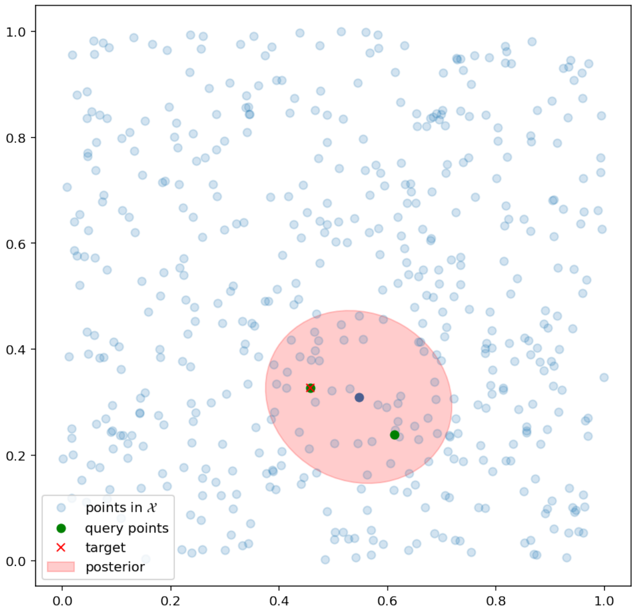


Figure 5.4: Vizualization of GAUSSSEARCH progress. Final stage. The Gaussian posterior concentrates around the target even more. The target is found.

5.1.5 Experimental Results: Search in Latent Setting

We now turn our attention to the latent setting, briefly introduced in Chapter 1. In this setting we assume that the ground-truth objects features $\boldsymbol{X}$ are not observable to us, and only the oracle has access to them when making a comparison according to a probabilistic choice model. We give a formal general description below.

This framework consists of two main ingredients: a search algorithm and an embedding algorithm.

Search. At each step m of a search episode the search scheme $\mathcal{S}$ proposes the user a set of candidate objects Q_m, $|Q_m| > 1$, i.e. a query. In our experiments we use $|Q_m| = 2$. The user is asked to choose one object $y_m \in Q_m$ that he considers the most similar to his target t. The choice of Q_m depends on the scheme's internal objects representations $\hat{\boldsymbol{X}}$ and on all previous queries and user choices:

$$Q_m = Q_m(\hat{\boldsymbol{X}}, (y_1, Q_1), (y_2, Q_2) \ldots, (y_{m-1}, Q_{m-1})).$$

The user is assumed to answer the scheme's query according to a comparison model $p(i, Q; t)$ with the true $\boldsymbol{X}$ hidden from the system. This process repeats until the target t appears as one of the objects in Q_m. The user then confirms his target and the search is complete. The search algorithm is designed for a specific choice of $p(i,Q;t)$.

Embedding. After a few completed searches, the system collects choice data of the form (i,Q_j), which is used to update the objects embedding $\hat{\boldsymbol{X}}$ assuming comparison model $p(i,Q;t)$. The new searches are run on the updated $\hat{\boldsymbol{X}}$.

The general scheme of LEARN2SEARCH is given in Algorithm 5.

Algorithm 5 Learn2Search

```
1: Initialize $\hat{\boldsymbol{X}}$
2: Initialize $\mathcal{T} \leftarrow \emptyset$
3: while true do
4:     Run $k$ interactive searches using embedding $\hat{\boldsymbol{X}}$
5:     Collect comparisons from these searches $\mathcal{T}_{\hat{\boldsymbol{X}}} = \{(y,Q;t)\}$
6:     Aggregate $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_{\hat{\boldsymbol{X}}}$
7:     Update $\hat{\boldsymbol{X}}$ on data $\mathcal{T}$
8: end while
```

We study the performance of the framework LEARN2SEARCH assuming Probit oracle model on two real world datasets: the *red wine* dataset [15] and the *music* dataset [56]. These datasets were studied in the context of comparison-based search in [29]. We assume that the true feature vectors $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ are latent: in our experiments we use them only to generate comparison outcomes, but $\mathcal{X}$ is not available to Algorithm 5. Instead, we use an embedding $\hat{\mathcal{X}}$ learned from the triplets from the previous searches as the objects representation in $\mathbb{R}^d$ for the search algorithm in Algorithm 5 in line 4. We also periodically update $\hat{\mathcal{X}}$ between searches as new triplet comparisons become available. See Fig. 1.2 for illustration.

We compare using different approaches to learning an estimate of the feature vectors from comparison outcomes in Algorithm 5 in line 7. In addition to our distributional embedding method GAUSSEMBED (listed in Algorithm 1), we consider two state-of-the-art embedding algorithms: t-STE [54] and CKL [47] (both of which produce point estimates of the feature vectors), as well as the ground-truth vectors $\mathcal{X}$. The search part of Algorithm 5 is handled by GAUSSSEARCH using $\hat{\mathcal{X}}$.

Usage of the distributional embedding GAUSSEMBED requires the following modifications in GAUSSSEARCH.

1. in SAMPLEMIRROR, we use the Mahalanobis distance $(\boldsymbol{\mu} - \boldsymbol{z})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{z})$ when finding the closest objects i, j to the sampled points $\boldsymbol{z}_1, \boldsymbol{z}_2$.

2. in the UPDATE step, we combine the variance of the noisy outcome σ_ε^2 with the uncertainty over the location of $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ along the direction perpendicular to the hyperplane, resulting in an effective noise variance $\sigma_{\text{eff}}^2 = \sigma_\varepsilon^2 + \boldsymbol{w}^\top \boldsymbol{\Sigma}_i \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{\Sigma}_j \boldsymbol{w}$.

Both modifications results in a natural tradeoff between exploration and exploitation within LEARN2SEARCH: In the early stages of the system, when only small set of triplet comparisons is available to us, we tend to sample points that have higher uncertainty in GAUSSEMBED more frequently, and the large effective noise variance restricts the speed at which the covariance of the belief decreases. After multiple searches, as we get more confident about the embedding, we start exploiting the structure of the space more assertively.

We run 9000 searches in total, starting from a randomly initialized $\hat{\mathcal{X}}$ and updating the embedding on the k-th search for $k \in [2^0, 2^1, \ldots, 2^{13}]$ For each search, the target object is chosen uniformly at random among all objects. The outcomes of the comparisons queried by GAUSSSEARCH are sampled from the Probit model (2.11) based on the ground-truth feature vectors $\mathcal{X}$. A search episode ends when the target appears as one of the two objects in the query. As before, the noise level σ_ε^2 is set to corrupt approximately 10% of the answers on average. As we jointly perform searches and learn the embedding, we measure the number of queries needed for GAUSSSEARCH to find the target using the current version of $\hat{\mathcal{X}}$ for each search, and then we average these numbers over a moving window of size 1000. Thus the resulting first value is the average number of queries in the first 1000 searches, the second value is the average number of queries in the searches $2, \ldots, 1001$ and so on.

We present the results, averaged over 100 experiments, in Fig. 5.5. The combination of our methods, GAUSSSEARCH and GAUSSEMBED, manages to learn object representations that give rise to search episodes that are as query-efficient as they would have been using the ground-truth vectors $\mathcal{X}$, and significantly outperforms variants using t-STE and CKL for generating embeddings. It appears that taking into account the uncertainty over the points' locations, as is done by GAUSSEMBED, helps to make fewer mistakes in early searches and thus leads to a lower query complexity in comparison to using point-estimates of vector embeddings. As the number of searches grows, GAUSSEMBED learns better and better representations $\hat{\mathcal{X}}$ and enables GAUSSSEARCH to ask fewer and fewer queries in order to find the target. In the final stages, our latent search method is as efficient as if it had access to the true embedding on both *wine* and *music* datasets.

One of the challenges that arises when learning an embedding in the latent setting is the choice of d. We denote the estimated dimensionality by D. In the case when the true d is not known, we estimated it by first conservatively setting $D = 100$, and then, after collecting around 10000 triplets, picking the smallest value of D with 98% of the energy in the eigenvalue decomposition of the covariance matrix of the mean vectors $\hat{\boldsymbol{X}}_\mu$. This number for both datasets was 20, and $D = 20$ was used as the approximation of the actual d. The results of running our LEARN2SEARCH with the estimated d is shown in Fig. 5.5 as the dashed lines. For the wine dataset, we achieved almost the same query complexity as for true d. For the music dataset, after 4000 searches our scheme with estimated d actually outperformed the true features $\mathcal{X}$: on average, when the search is run on $\hat{\mathcal{X}}$, GAUSSSEARCH needs fewer queries than when it is performed on $\mathcal{X}$. This suggests that LEARN2SEARCH is not only robust to the choice of D, but is also capable of learning useful object representations for the comparison-based search independent of the true features of the objects.

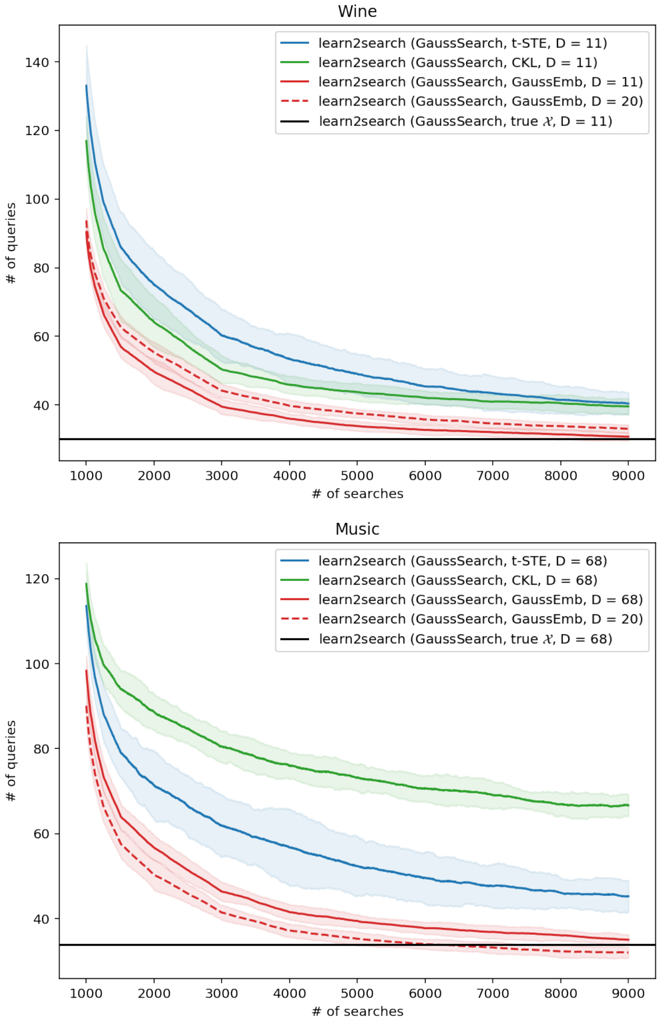


Figure 5.5: Combined search and embedding framework LEARN2SEARCH on two datasets with different embedding techniques and choices of D. Results are reported over a sliding window of size 1000.

5.2 γ-CKL model: Exponential Convergence and γ-CKLSearch

In this section we assume the γ-CKL oracle model (2.12). Due to its scale-free property, we are interested in search algorithms with an *exponential* convergence rate. We begin with proposing a scheme with a backtracking mechanism that provably navigates towards the target exponentially fast, assuming the dense space $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d$.

5.2.1 Exponential Convergence: a Scheme with Backtracking

In this subsection, we prove a search algorithm with backtracking mechanism that has an exponential convergence rate. Then we propose a heuristic implementation of that algorithm and demonstrate its exponential convergence rate in a number of experiments.

As before, assume a continuous target $\boldsymbol{x}_t \in \Omega$, where $\Omega = [0,1]^d \subset \mathbb{R}^d$. In this continuous setting, a search algorithm should be able to “zoom in” indefinitely, thus finding ever smaller regions containing the target.

Due to the scale-free nature of γ-CKL model, at each zoom level we expect the same maximum constant level of noise, as long as the target lays in the considered region. This would enable us to make progress towards the target (via shrinking current considered belief region) by asking a constant number of queries at each zoom level, hence suggesting an exponential rate of convergence, assuming we do not lose the target during this process.

We consider the following algorithm. At each stage $s \geq 1$, the algorithm operates in its current belief region $X_s \subset \Omega$ and asks queries to the oracle, until it decides to either

(P) Proceed (zoom in) to a child region of X_s of a smaller size or

(B) Backtrack (zoom out) to the parent region in Ω containing X_s.

When the algorithm arrives to a new region, a new stage begins and the algorithm discards all information about previous region, previous queries and their outcomes. Thus the decisions taken by the algorithm are local and do not depend on the past, given a target location. This is an important property and it will be very useful further in our analysis.

We frame our search process as a random walk on a graph, where each node corresponds to a region $X \in \Omega$. We will show that, under certain assumptions on transition probabilities between regions, an erroneous decision, i.e., zooming into a region that does not contain the target, must eventually be undone with probability 1. A high-level overview of this idea is given in Algorithm 6. The algorithm begins in $X_0 = \Omega$ and is given a budge of queries to be asked to the oracle M. At each stage s, it keeps track of the queries asked to the oracle during that stage and the received answers via maintaining a history $\mathcal{D}$. This history is getting emptied at the beginning of each stage. The algorithm begins asking queries to the oracle chosen via the subroutine NEXTQUERY($X_s, \mathcal{D}$) and receiving the comparison outcomes $\hat{y}$ according to the γ-CKL probabilistic model, until the decision (action) A_s to either proceed (P) or backtrack(B) is taken via the soubroutine DECIDE($X_s, \mathcal{D}$). If the decision A_s is to backtrack, the algorithm begins a new phase in a larger region $X_{s+1} = u(X_s)$, otherwise the algorithm zooms in into a smaller region $X_{s+1} = d(X_s)$, provided by DECIDE($X_s, \mathcal{D}$). The algorithm stops once the total number of queries made to the oracle exceeds the budge M.

Algorithm 6 General exponential search

```
Input: query budget $M$
$s \leftarrow 0$                                                  {Stage}
$X_0 \leftarrow \Omega$                                           {Current region}
$m \leftarrow 0$                                                  {Number of queries asked}
repeat
    $\mathcal{D} = \{\}$                                          {Erase history}
    repeat
        $(\hat{x}_i, \hat{x}_j) \leftarrow$ NEXTQUERY($X_s, \mathcal{D}$)
        $\hat{y} \sim$ Bernoulli($p(\hat{x}_i, \hat{x}_j; x_t)$)  {Obtain oracle's answer}
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\hat{x}_i, \hat{x}_j, \hat{y})\}$
        $m \leftarrow m + 1$
    until $A_s$ = DECIDE($X_s, \mathcal{D}$)
    if $A_s$ = (B) then
        $X_{s+1} \leftarrow u(X_s)$                               {Backtrack}
    end if
    if $A_s$ = (P) then
        $X_{s+1} \leftarrow d(X_s)$                               {Proceed}
    end if
until $m > M$
```

We assume that the regions have forms of hypercubes and $X_0 = \Omega$ (nevertheless, the idea of the algorithm applies to arbitrary regions). For convenience, we also assume that after arriving to the region X, it is centered at the origin. Let us now define child and parent regions.

Let $X \subset \Omega$ be the current region of belief. Without loss of generality, due to the scale-free nature of the γ-CKL model, we assume X has an edge length 1 and is centered at the origin. Let S be a hypercube centered at the origin with edge length $\frac{3}{2}$, S contains X. Let $\mathcal{T}(S, \frac{1}{4})$ be a set of hypercubes with edge length $\frac{1}{4}$ that tile S. Then

- A set of hypercubes $D(X)$ within S with edge length $\frac{1}{2}$ that can be constructed by joining tiles in $\mathcal{T}(S, \frac{1}{4})$ are called the **children** regions of $\mathcal{R}$, in total $|D(X)| = 5^d$. A children region from $D(X)$ is denoted by $d(X)$.
- The hypercube of edge length 4, centered at the origin, is called the **parent** region of X, because it contains all regions X' from which we could have proceeded to X, i.e. $X \in D(X')$, hence a union of all direct ancestors of X. We denote the parent region by $u(X)$.

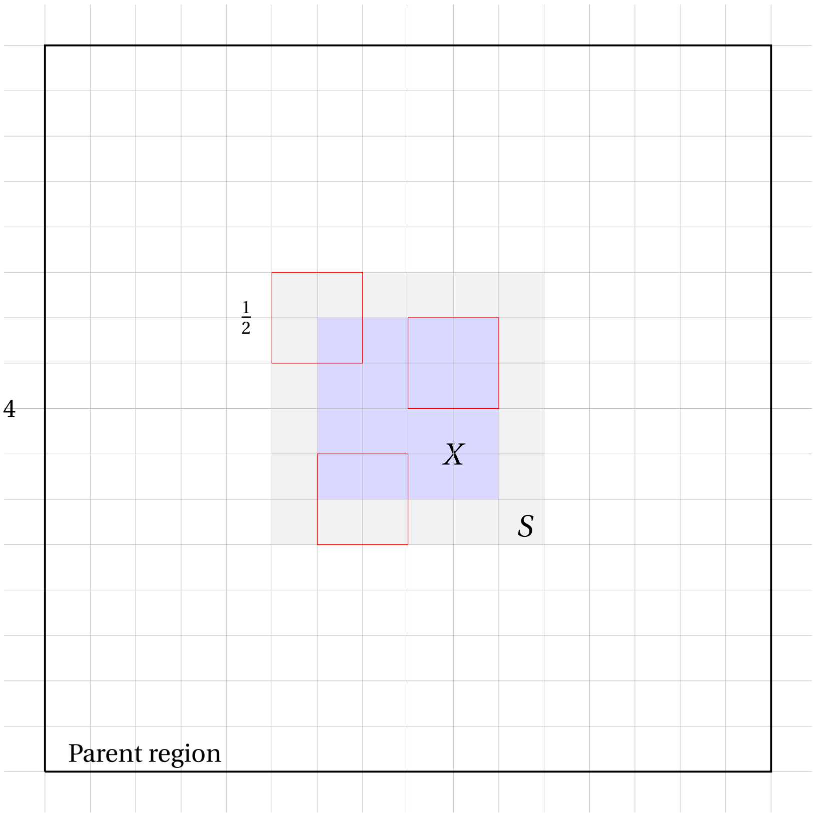


Figure 5.6: Parents and Children. The blue square represents the current region X. The red squares are the children of X, contained in $D(X)$. The union of grey and blue areas is the region S. The big square around S is the parent region $u(X)$.

For illustration, see Fig. 5.6. If we go to a child region, we decrease the edge length of the current region by a factor 1/2, thus allowing exponentially fast convergence. If we backtrack to a parent region, we increase the edge length by a factor of 4. The backtracking mechanism allows to undo potential mistakes and return to a bigger region in order to ask more global queries, instead of further zooming into a region that does not contain the target.

For our analysis we also need define the color of a region. If a region X contains $\boldsymbol{x}_t$, we call it green (correct), otherwise we call it red (incorrect). Note, that since the parent region $u(X)$ contains X, backtracking from a green region X will always result to backtracking into also a green region. And if X is green, it must have at least on green child in $D(X)$.

At each stage, our algorithm collects query replies from the oracle, until it decides on an action A_s, either (P) or (B), via routine $\textsc{Decide}(X_s, \mathcal{D})$ that has the following transition probabilities:

- X is green:
  - Proceed to one of its *green children* with $p_d(X, \boldsymbol{x}_t)$.
  - Stray, i.e. proceed to one of its *red children*, with $q_s(X, \boldsymbol{x}_t)$.
  - Backtrack to its (green) *parent* region with at most $q_u(X, \boldsymbol{x}_t)$.
- X is red:
  - Proceed to one of its *red children* with $q_d(X, \boldsymbol{x}_t)$.
  - Recover, i.e. proceed to one of its *green children*, with $p_r(X, \boldsymbol{x}_t)$.
  - Backtrack to its *parent* region with at least $p_u(X, \boldsymbol{x}_t)$.

Naturally, we use p when denoting the probability of a transition, if the action is *correct*, and q when the action is *incorrect*. Indeed, proceeding to a green node is a correct decision, whereas backtracking from a green node is incorrect. Proceeding to a red node is incorrect, whereas backtracking from a red node is correct. For a green region, there are two incorrect decisions: backtracking and proceeding to a red child. For a red region, there are two correct decisions: backtracking and recovering by proceeding to a green child. We summarize the probabilities of correct and incorrect actions based on the color of a region in Table 5.1.

	X is green	X is red
Correct transition:	$p_d(X, \boldsymbol{x}_t)$	$p_u(X, \boldsymbol{x}_t) + p_r(X, \boldsymbol{x}_t)$
Incorrect transition:	$q_u(X, \boldsymbol{x}_t) + q_s(X, \boldsymbol{x}_t)$	$q_d(X, \boldsymbol{x}_t)$

Table 5.1: Transition probabilities based on the color of the region.

We illustrate an example of a transition graph in Fig. 5.7.

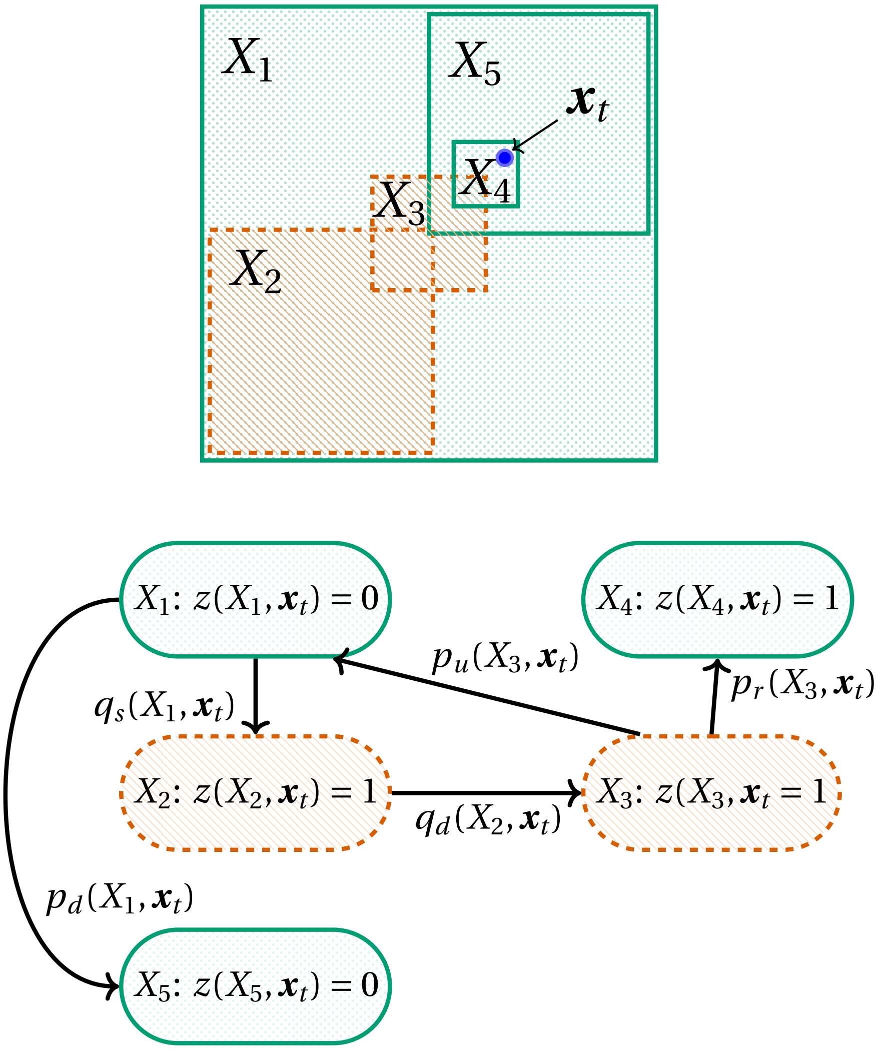

X_1
X_5
$\boldsymbol{x}_t$
X_4
X_3
X_2
X_1: $z(X_1, \boldsymbol{x}_t) = 0$
X_4: $z(X_4, \boldsymbol{x}_t) = 1$
$p_u(X_3, \boldsymbol{x}_t)$
$p_r(X_3, \boldsymbol{x}_t)$
$q_s(X_1, \boldsymbol{x}_t)$
X_2: $z(X_2, \boldsymbol{x}_t) = 1$
$q_d(X_2, \boldsymbol{x}_t)$
X_3: $z(X_3, \boldsymbol{x}_t = 1$
$p_d(X_1, \boldsymbol{x}_t)$
X_5: $z(X_5, \boldsymbol{x}_t) = 0$

Figure 5.7: (*credits to L. Klein*) Regions and target (blue dot) at the top, selected transitions at the bottom.

Lemma 1. *The sequence of regions X_s visited in each stage s of the search process forms a random walk.*

Proof. Let D be the decision made by the algorithm. This is a random variable which can take values in $(B, P_1, P_2, ..., P_{5^d})$, for backtracking or proceeding to one of the children of X. When arriving at a region X, the algorithm discards information about previous query outcomes. Then it asks a series of queries, prescribed by our Inner Loop algorithm. Once the query outcomes have been observed, the decision is deterministic. Therefore, to describe the distribution of D it is sufficient to describe the distribution of queries and query outcomes. The queries that we ask depend only on the current region. The outcomes are conditionally independent, given a target location. Therefore, the distribution of D only depends on the current region and the latent $\boldsymbol{x}_t$. This means that the decision to proceed or backtrack is Markovian. The sequence of regions $\{X_s\}_{s=0}^{\infty}$ is then a random walk. □

Intuitively the necessary conditions for a scheme to be able to successfully navigate to towards the target, would be to perform correct transitions with higher probability than incorrect.

We denote the total correct decision probability by

$$p(X,\boldsymbol{x}_t) = p_d(X,\boldsymbol{x}_t)\mathbb{1}\{X \text{ is green}\} + (p_u(X,\boldsymbol{x}_t) + p_r(X,\boldsymbol{x}_t))\mathbb{1}\{X \text{ is red}\}$$

and the total incorrect decision probability by

$$q(X,\boldsymbol{x}_t) = (q_u(X,\boldsymbol{x}_t) + q_s(X,\boldsymbol{x}_t))\mathbb{1}\{X \text{ is green}\} + q_d(X,\boldsymbol{x}_t)\mathbb{1}\{X \text{ is red}\}.$$

To quantify the progress of our search, we keep track of the *depth* $P(X)$ of a region, i.e. the (minimum) number of consecutive proceed actions needed to reach this region, starting from Ω. A hupercube region X at depth $P(X)$ has the edge length of $\frac{1}{2}^{P(X)}$. The k-th ancestor $u(X,k)$ is reached by backtracking k times from X. Assuming the subroutine DECIDE satisfies two mild assumptions, the following theorem shows the exponential rate of convergence of the Algorithm 6: (1) at any stage s of the algorithm, $u(X_s,k)$ contains the target with high probability with a probability that does not depend on s, and the depth of $u(X_s,k)$ increased linearly with s.

Theorem 4. *Assume for any $b \in (0,1]$ there exists an algorithm* DECIDE *such that for any $\boldsymbol{x}_t \in \Omega$ and for any $X \subset \Omega$ its transition probabilities satisfy*

(A1) $p(X,\boldsymbol{x}_t) - q(X,\boldsymbol{x}_t) > b,$

(A2) $\boldsymbol{x}_t \in X \implies p_d(X,\boldsymbol{x}_t) - 2q_u(X,\boldsymbol{x}_t) - q_s(X,\boldsymbol{x}_t)\frac{b+1}{2b} > 0.$

under γ-CKL oracle model. Then for any desired probability of error $\delta > 0$, there exist constants $k, C > 0$ such that

$$\mathbf{P}(\boldsymbol{x}_t \in u(X_s,k)) > 1-\delta$$
$$\mathbb{E}[P(u(X_s),k)] < Cs.$$

Proof sketch. We define a stopping time of arriving at a green region after leaving a green region by proceeding to a red region. Under assumption $(A1)$, we prove an upper bound for the expectation of this stopping time which does not depend on s. Then using assumption

$(A2)$, we show that the expected depth of each consecutive green region increases linearly. Finally, by using the random walk Z_s as a stochastic upper bound, we use its time-homogenous transition probabilities to prove the statement. □

We will prove the existence of a DECIDE algorithm that satisfies $(A1)$ and $(A2)$ later in the section, but first we describe the main pieces needed to prove Theorem 4. To that end, we need several auxiliary lemmas.

First, at each step let $z(\boldsymbol{x}_t, X_s)$ be the number of backtracking decisions needed to reach a green region from the current region X_s. In the beginning, $z(\boldsymbol{x}_t, X_0) = 0$. If X_s is green, $z(\boldsymbol{x}_t, X_0) = 0$. If the search proceeds to a red child, $z(\boldsymbol{x}_t, X_s)$ is either increased by 1 or stays unchanged (the latter is possible if no additional backtracking is required, i.e. $z(\boldsymbol{x}_t, X_s) = z(\boldsymbol{x}_t, X_{s-1})$. If X_s is red, but we recover, i.e. proceed to a green child, $z(\boldsymbol{x}_t, X_s)$ is set to 0.

From assumption $(A1)$ we get

- $\boldsymbol{x}_t \in X_s \Rightarrow q_u(X, \boldsymbol{x}_t) + q_s(X, \boldsymbol{x}_t) < \frac{1-b}{2}$,
- $\boldsymbol{x}_t \not\in X_s \Rightarrow q_d(X, \boldsymbol{x}_t) < \frac{1-b}{2}$.

We now construct a time-homogenous random walk Z_s that will serve as a stochastic upper bound for $z(\boldsymbol{x}_t, X_s)$. Let Z_s be a random walk on natural numbers, starting at $Z_0 = z(\boldsymbol{x}_t, X_0) = 0$. At each step, Z_s is incremented with probability $\frac{1-b}{2}$ and decremented with probability $\frac{1+b}{2}$. Once $Z_s = 0$, $Z_{s+1} = 0$ with probability $\frac{1+b}{2}$ or $Z_{s+1} = 1$ with probability $\frac{1-b}{2}$. Then

Lemma 2. *Given a soubroutine* DECIDE *that satisfies Assumption* $(A1)$, Z_s *is a stochastic upper bound for* $z(\boldsymbol{x}_t, X_s)$, *denoted by* $z(\boldsymbol{x}_t, X_s) \preceq_{st.} Z_s$.

Proof sketch. We construct a coupling between the random walk $\tilde{X}_s$ and a random variable $\tilde{Z}$. We then use induction to show that with probability 1 it holds that $\tilde{Z} > z(\boldsymbol{x}_t, \tilde{X}_s)$. □

Now let $\tau_X = \inf\{s > 0 \mid \boldsymbol{x}_t \in X_s, X_0 = X\}$ be the stopping time of reaching a green region, starting from X. As long as we are on a green path, i.e. keep proceeding to green children, τ_{X_s} is 0. Once we make an incorrect decision by proceeding to a red child, the expected return time is bounded by a constant and does not depend on s:

Lemma 3. *Let* X_s *be red and* $u(X_s)$ *be green (this occurs after just having strayed from a green region). Given a soubroutine* DECIDE *that satisfies Assumption* $(A1)$,

$$\mathbb{E}[\tau_{X_s}] \leq \frac{1}{b}.$$

Proof sketch. Given the stochastic upper bound Z_s for $z(\boldsymbol{x}_t, X_s)$, we show $\mathbb{E}[\tau_{X_s}] \leq \mathbb{E}[\tau_{Z=1}] = \frac{1}{b}$, where $\tau_{Z=N} = \inf\{s > 0 \mid Z_s, Z_0 = N\}$. □

Existence of a subroutine decision criterion $\textbf{D}\textsc{ecide}(X_s, \mathcal{D})$. Finally, we show the existence of a decision scheme that asks at most a constant number of queries, until it arrives at a decision A_s, such that the transaction probabilities (based on the lines 12-17 of Algorithm 6) satisfy the assumptions $(A1) - (A2)$ from Theorem 4.

We present a subroutine that is based on a test for the hypothesis (H) "*$\boldsymbol{x}_t$ is in the region X*" for an arbitrary region $X \subset \Omega$ (not to be confused with the current belief region X_s). As $\boldsymbol{x}_t$ approaches the boundary of X, it becomes increasingly hard to distinguish whether the target point is inside or outside of X. We then define a region of uncertainty U around X in which our hypothesis test is not reliable.

Without loss of generality, let the region of interest X be a hypercube of edge length 2, centered at the origin and let U be a hypersphere with radius $r_u > 1$, also centered at the origin. Everything outside of U is called $F = \Omega \setminus U$.

With the following lemma we show that it is possible to construct a query $Q = (\boldsymbol{x}_i, \boldsymbol{x}_j)$ and chose a value of r_u such that by repeatedly querying Q to the oracle and observing the outcomes, it will enables us, with probability $> 1 - \delta$, to accept (H) if $\boldsymbol{x}_t \in X$, or to reject (H) if $\boldsymbol{x}_t \in F$.

Lemma 4. *Let the query* $Q = (\vec{\mathbf{0}}, (1+d)\mathbf{e})$, *where* $\mathbf{e} = (1, 0, 0, \dots)$ *is a unit vector along an arbitrarily chosen axis in* $\mathbb{R}^d$. *Let* $r_u = 1 + \frac{d + \sqrt{d^3 + d^2 - d}}{d - 1}$. *Let* X, U, F *be defined as above. Then for any* $\delta > 0$ *asking the query* Q *to the oracle a constant number of times is enough to apply a one-tailed binomial hypothesis test which with probability* $1 - \delta$ *will:*

- *accept (H), if* $\boldsymbol{x}_t \in X$,
- *reject (H), if* $\boldsymbol{x} \in F$.

The necessary number of observations does not depend on X *and* $\boldsymbol{x}_t$.

Proof sketch. The query compares a point inside of the region X with a point outside. We can lower bound the probability of the point inside being preferred, assuming that $\boldsymbol{x}_t \in X$. We then calculate a radius r_u such that for any $\boldsymbol{x}_t \in F$, the probability of the point inside being preferred is strictly smaller. From there we can apply a binomial hypothesis test. □

Now consider our current belief region X_s at stage s of the Algorithm 6. We cannot directly apply Lemma 4 to X_s because some of child regions $d(X_s)$ could be included inside the region of uncertainty U. Instead, we construct a finer discretization grid of S, which will enable us to apply lemma for the cells of that grid.

As we mention before, in Lemma 4 we assume a region of edge length 2. As our oracle model is scale-free, we can apply the hypothesis test to regions of arbitrary size. Let $\mathcal{T}(S, r_c)$ be a tiling of S with hypercubes of edge length r_c, we refer to the cells in this tiling by $c_k, k = 1, 2, \ldots, K$, the respective centers are $\boldsymbol{x}_{c_k}$. We can now apply Lemma 4 to c_k. Since the edge lengths of c_k are r_c instead of 2, the uncertainty region needs to be scaled as well. Particularly, for a region with edge length r_c, the radius of the uncertain region is scaled by factor $r_c/2$. If we pick $r_c < \frac{1}{8r_u}$, where r_u is as in the lemma, that will lead to an uncertain region with radius $r_u\frac{r_c}{2} < \frac{1}{16}$. If the edge length of S is not divisible by r_c, it is always possible to pick a smaller value for r_c.

After applying the hypothesis test from Lemma 4, i.e. (H) is "x_t is in the region c_k", each cell c_k in the tiling $\mathcal{T}(S, r_c)$ will belong to one of three classes:

- (A) $\boldsymbol{x}_t \in c_k$. When using the hypothesis test, with high probability, our test will not reject (H). We assume that cells include their border. If the target happens to lie exactly on the boundary between cells, then all of them belong to class (A).
- (B) $\boldsymbol{x}_t \notin c_k \wedge ||\boldsymbol{x}_t - \boldsymbol{x}_{c_k}|| < \frac{1}{16}$. When using the hypothesis test, the target lies in the uncertain region. We do not make any assumption about whether (H) is rejected or not.
- (C) $\boldsymbol{x}_t \notin c_k \wedge ||\boldsymbol{x}_t - \boldsymbol{x}_{c_k}|| \geq \frac{1}{16}$. When using the hypothesis test, with high probability, our test will reject hypothesis (H).

When using the hypothesis test, we know that, with high probability, all cells in class (C) are rejected. We can show that the remaining cells in classes (A) and (B) fit in a small bounding box:

Lemma 5. *Assume $\mathbf{x}_t \in S$. Then there is a hypercube $\mathcal{B} \subset S$ with an edge length of less than $\frac{1}{4}$, such that all cells in the classes (A) and (B) are fully contained in $\mathcal{B}$.*

Proof. Let $\mathcal{B}'$ be a hypercube, centered at $\boldsymbol{x}_t$ and with edge length $2\frac{1}{16} = \frac{1}{8}$. If a cell c_k is in class (A) or (B) then its center $\mathbf{x}_{c_k}$ must lie in the hypercube B. We now extend the edge length of this hypercube to fully contain any cell whose center lies in the hypercube. Let $\mathcal{B}$ be a hypercube, centered at $\boldsymbol{x}_t$ and with edge length $2\frac{1}{16} + r_c$. It follows immediately that $\bigcup_{c_k \text{ has class (B) or (A)}} c_k \subseteq \mathcal{B}$. We have chosen $r_c < \frac{1}{8r_u} < \frac{1}{8}$. Therefore it follows that the edge length of $\mathcal{B}$ is $2\frac{1}{16} + r_c < \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$. □

We illustrate this observation in Figure 5.8.

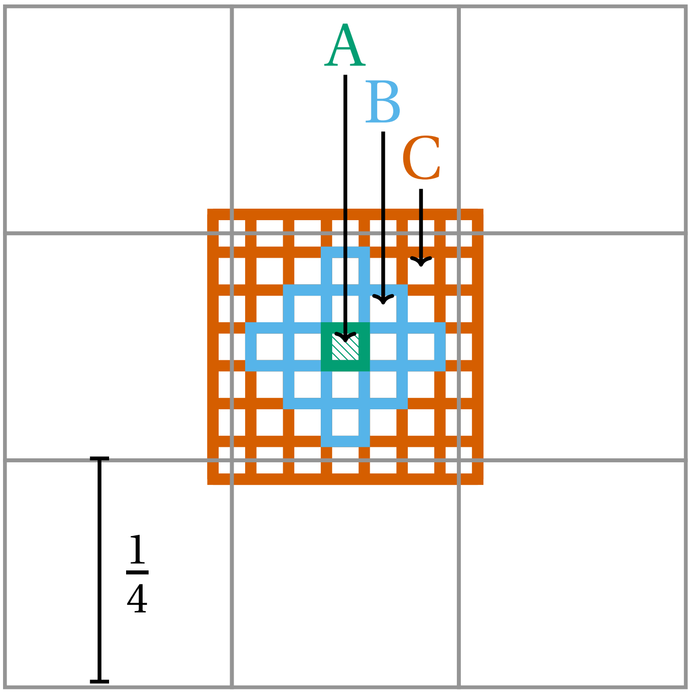


Figure 5.8: (*credits to L. Klein*) Layout of the nested grid with three classes of cells. Small squares (cells) with think edges are part of the tiling of S of size r_c. The grey square in middle is $\mathcal{B}$ containing all cells from classes (A) and (B). The big square of size $\frac{3}{4}$ is a part of S.

We now propose the following subroutines NEXTQUERY and DECIDE for Algorithm 6.

NEXTQUERY: given the current region of belief X_s, we ask queries as in Lemma 4 for each cell in $\mathcal{T}(S, r_c)$ and apply the hypothesis tests for a desired error probability δ.

DECIDE: let $\mathcal{B}$ be the bounding box containing all cells c_k for which (H) "x_t is in the region c_k" was not rejected. If there is a child region of X_s that fully contains $\mathcal{B}$, we proceed to it. Otherwise we backtrack.

This mechanism is formalized in Algorithm 7.

Algorithm 7 Hypothesis test criterion for NEXTQUERY and DECIDE

```
Input: Current belief region $X_s$, region $S$
Set up a discretization $\mathcal{T}(S, r_c)$
$H \leftarrow \emptyset$
for $c_k \in \mathcal{T}(S, r_c)$ do
    Perform the hypothesis test from Lemma 4 for $c_k$
    if (H) is not rejected then
        $H \leftarrow H \cup c_k$
    end if
    if $\exists d(X_s) \in D(X_s) : H \subseteq d(X_s)$ then
        $A_s$ = (P), proceed to $d(X_s)$
    else
        $A_s$ = (B), backtrack to parent $u(X_s)$
    end if
end for
```

The following Theorem 5 shows that this algorithm enables us to make decisions that satisfy assumptions $(A1)$ and $(A2)$ from Theorem 4.

Theorem 5. *For any desired $\hat{\delta} > 0$, Algorithm 7 needs only a finite number of queries in order to make a decision (P) or (B) which is incorrect with probability $\hat{\delta}$. In particular, choosing $\hat{\delta}$ small enough ensures that the Algorithm 7 is compatible with assumptions* $(A1)-(A2)$.

Proof sketch. We analyze the possible constellations, in which classes (A), (B), and (C) can overlap with the region X_s and its children $D(X_s)$. If there is no error in the hypothesis tests, the decision of the algorithm is correct, for any possible constellation. The accuracy of the hypothesis tests δ can be adjusted to ensure an overall probability of error (i.e. taking an incorrect decision based on the color of the current region X_s) is $\hat{\delta}$. $\square$

Combining Algorithm 6 and Algorithm 7 we conclude the description of a search scheme that has a provable exponential convergence rate.

5.2.2 Implementation of the Scheme with Backtracking

In practice, it is not efficient to conduct a series of independent hypothesis tests. Instead we propose a heuristic that relies on numerical integration. In our implementation we use hyperspheres to represent regions instead of hypercubes. Within each stage s, the algorithm operates in a region $X_s \subset \mathbb{R}^d$ with radius r_s, collects oracle replies and updates an approximation of the posterior distribution of the target location, until a decision can be made. We define two auxiliary regions, $E_s = \{\boldsymbol{e} \in \mathbb{R}^d \mid \exists \boldsymbol{x} \in X_s : \|\boldsymbol{x} - \boldsymbol{e}\| < w_e r_s\}$, which plays a role of a region of uncertainty, and $F_s = \{\boldsymbol{e} \in \mathbb{R}^d \mid \exists \boldsymbol{x} \in X_s : \|\boldsymbol{x} - \boldsymbol{e}\| < w_f r_s\} \setminus (X_s \cup E_s)$, which is used to make a backtracking decision.

Our decisions to either proceed or to backtrack is based on the posterior distribution in the region $E_s \cup F_s$ given query outcomes. We model this distribution using a grid of randomly sampled points $C = \{\boldsymbol{c}_i\}_{i=1}^K$ that discretizes $E_s \cup F_s$ and by keeping track of discretization points individual posterior mass $p(\boldsymbol{c}_k) = \sum_{Q_j, y_j} \log p(Y = y_j \mid Q_i, \boldsymbol{c}_k)$ with respect to the queries and outcomes $\{Q_j, y_j\}$ during the stage s. The queries are asked inside the region $X_s \cup E_s$.

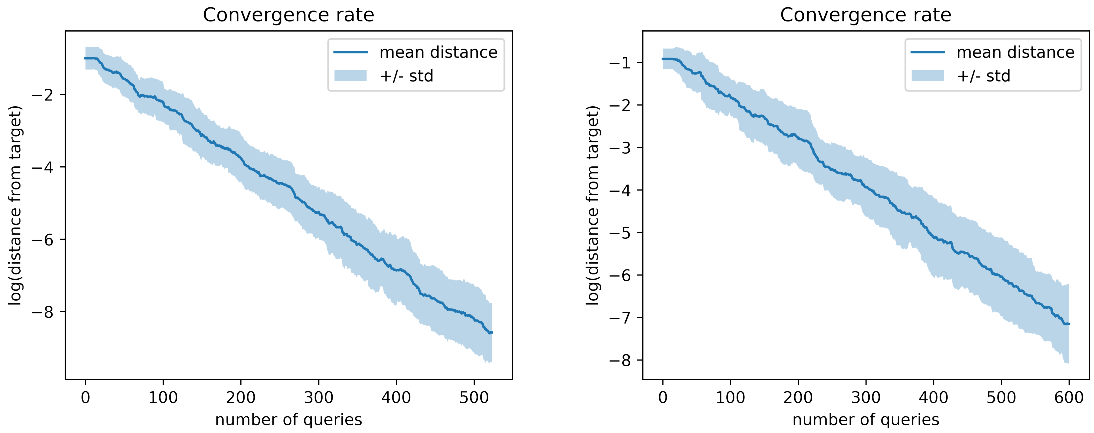
The algorithm proceeds if there is child subregion $d(X) \subset X_s$ with $p(d(X), C) = \frac{\sum_{\boldsymbol{c}_k \in d(X)} p(\boldsymbol{c}_k)}{\sum_{\boldsymbol{c}_k} p(\boldsymbol{c}_k)} > p_f$ for some threshold $p_f \in [0,1]$, where the set of child subregions is defined as a set of all possible hyperspheres of a smaller radius $r_s w_d$ inside X_s. If we observe $p(X, C) < 1 - b_f$, $b_f \in [0,1]$, the algorithm backtracks to the parent region $u(X_s)$ defined as a hypersphere of a bigger radius $r_c w_b$ centered at the center of X_s. If we still discard prior information of visited regions, history of queries and their outcomes at the beginning of each stage, this algorithm will exhibit an exponential rate of convergence.

We outline this heuristic implementation in Algorithm 8.

Algorithm 8 Exponential search: a heuristic

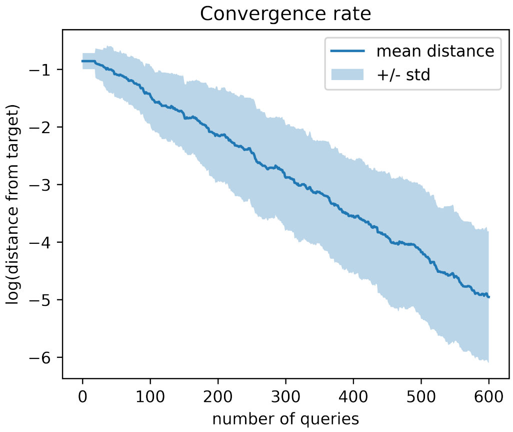
```
Input: $p_f, b_f, w_e, w_f, w_d, w_b, K$
$X_0 \leftarrow \Omega$
for $s = 0 \dots \infty$ do
    $E_s \leftarrow \{\boldsymbol{e} \mid \exists \boldsymbol{x} \in X_s : \|\boldsymbol{x} - \boldsymbol{e}\| < w_e r_s\}$
    $F_s \leftarrow \{\boldsymbol{e} \mid \exists \boldsymbol{x} \in X_s : \|\boldsymbol{x} - \boldsymbol{e}\| < w_f r_s\} \setminus (X_s \cup E_s)$, an envelope around $X_s \cup E_s$
    Sample i.i.d. points $C = \boldsymbol{c}_1, ..., \boldsymbol{c}_K \sim \mathcal{U}(E_s \cup F_s)$
    $L(\boldsymbol{c}_k) \leftarrow 0$, $k = 1, \dots, K$
    for $j = 0 \dots \infty$ do
        Sample $Q_j = \{\boldsymbol{a}_j, \boldsymbol{b}_j\} \sim \mathcal{U}(X_s \cup E_s)$
        Observe the outcome of $y_j \sim p(Y_j \mid Q_j = \{\boldsymbol{a}_j, \boldsymbol{b}_j\})$
        $p(\boldsymbol{c}_k) \leftarrow p(\boldsymbol{c}_k) \log p(Y = y_j \mid Q = \{\boldsymbol{a}_i, \boldsymbol{b}_j\}, \boldsymbol{c}_k)$, $k = 1, \dots, K$
        if $p(E_s \cup F_s \setminus X_s) > b_f$ then
            $X_{s+1} \leftarrow u(X_s)$
            break
        end if
        if $p(X_s) > p_f$ then
            $X_{s+1} \leftarrow d^\star(X_s)$, $d^\star(X_s)$ is the child region centered at the mean of $p(X_s)$
            break
        end if
    end for
end for
```

We have evaluated the performance of Algorithm 8 in a series of synthetic experiments by varying γ and d. The results are presented in Fig. 5.9. We plot the average distance to the target as well as the standard deviation as a function of number of queries. We observe the exponential convergence to the target, as the number of queries grow. Our implementation provides a variety of tunable hyperparameters. We find that the algorithm performs reliably for a wide range of parametrizations. Our synthetic search experiments presented are all based on the same parametrization and show an exponential rate of convergence towards the target. In particular, we used the following values for the input: $p_f = 0.85$, $b_f = 0.92$, $w_e = 1.25$, $w_f = 1.5$, $w_d = 0.7$, $w_b = 3.35$.



(a) $d = 3, \gamma = 5$

(b) $d = 4, \gamma = 7$



(c) $d = 5, \gamma = 9$

Figure 5.9: Synthetic experiment, average distance as a function of the number of queries for different values of d and γ. Y-axis is in natural log scale. The results are averaged over 50 runs.

5.2.3 γ-CKLSEARCH for Moderate n

In the case when there is only a finite number n of points, we can keep the full posterior distribution $P = [p_1, p_2, \ldots, p_n]$ over all n objects and propose a more efficient algorithm that the ones introduced in the previous subsection for continuous Ω. Since $\boldsymbol{x}_t$ is not known by the system during the search, we take a Bayesian approach to model the probability of the objects in $[n]$ to be the target, and at each step m of the search maintain a full belief $\mathcal{P}^m = [p_1^m, p_2^m, \ldots, p_n^m]$ over all n objects. We start with a uniform prior $\mathcal{P}_0 = [\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}]$.

Choosing the next query to ask the user. Similarly to GAUSSSEARCH, at each step we would like to ask a query $\{i, j\}$ that would maximize the *expected information gain* given the current posterior belief $\mathcal{P}_m$ at step m of the search:

$$\{i,j\} := \max_{i \neq j} \big(H(\mathcal{P}_m) - \mathbb{E}_{Y|\boldsymbol{x}_i,\boldsymbol{x}_j}[H(\mathcal{P}_m \mid Y)] \big), \tag{5.2}$$

where $Y \sim p(Y|\boldsymbol{x}_i,\boldsymbol{x}_j)$ is the marginalized belief over the answers to the query $\{i,j\}$, i.e.

$$p(Y = i \mid \boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{k=1}^{n} p(\boldsymbol{x}_i,\boldsymbol{x}_j;\boldsymbol{x}_k)\ p_k^m.$$

Performing an exhaustive search over all $O(n^2)$ possible pairs (i,j) in order to find the optimal query in terms of (5.2) would be prohibitively slow, so we propose an alternative heuristic that has good performance in practice.

We first detect the direction along which the variance of the belief is maximized, for that a sample mean and a covariance matrix $(\bar{\boldsymbol{\mu}}_m,\bar{\boldsymbol{\Sigma}}_m)$ are computed from the current belief $\mathcal{P}_m$. Next we build a proto-query as a pair of two points $(\tilde{\boldsymbol{z}}_1,\tilde{\boldsymbol{z}}_2)$ in $\mathbb{R}^d$ that lie in the direction of the maximum variance of $\bar{\boldsymbol{\Sigma}}_m$ on opposite sides of the sample mean $\bar{\boldsymbol{\mu}}_m$. In order to have a desired explore-exploit trade-off of a query, we control the distance from $\tilde{\boldsymbol{z}}_j$ to $\bar{\boldsymbol{\mu}}_m$ by a multiplication parameter $r \in \mathbb{R}$. Finally, we find two distinct objects (i,j) from $[n]$ which have the closest representations to $(\tilde{\boldsymbol{z}}_1,\tilde{\boldsymbol{z}}_2)$ in a $\mathcal{P}_m$-weighted Euclidean distance, which favors the near and more probable points. This pair (i,j) becomes the next query to the oracle.

Posterior Update. After we obtain the response from the user, $\bar{y} \in \{i,j\}$, the posterior probabilities are updated using Bayes rule,

$$p_k^{m+1} = p_k^m\ p(Y = \bar{y} \mid \boldsymbol{x}_i,\boldsymbol{x}_j)/C, \quad k = 1,2,\ldots,n \tag{5.3}$$

where $C = \sum_{k=1}^{n} p_k^m\ p(Y = \bar{y} \mid \boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)$ is the normalizing costant.

The search finishes when the user indicates one of the query objects as his target, otherwise both query objects are considered to be non-target and further do not appear in the search. We keep track of the objects that we have displayed to the user already using the set of seen objects $\mathcal{U}$. The overall search algorithm is outlined in Algorithm 9.

The complexity of each step of the Algorithm 9 is $O(nd + d^2)$, since computing the sample covariance is $O(nd)$ and finding the principle eigenvector can be approximated with the power method in $O(d^2)$. Since in practice the number of features d remains constant, the complexity is linear in the number of objects n.

Algorithm 9 γ-CKLSEARCH

```
$m \leftarrow 0$
$\mathcal{U} \leftarrow \emptyset$
Initialize the prior $\mathcal{P}_0$ with $p_k^0 \leftarrow \frac{1}{n}$, $\forall k = 1, 2, \ldots, n$
repeat
    Compute the sample mean $\bar{\boldsymbol{\mu}}_m$ and the sample covariance $\bar{\boldsymbol{\Sigma}}_m$ from the current belief $\mathcal{P}_m$
    Find the largest eigenvalue of $\bar{\boldsymbol{\Sigma}}_m$ and its eigenvector, $\lambda_{\max}$ and $\boldsymbol{v}_{\max}$ respectively
    $\tilde{\boldsymbol{z}}_1 \leftarrow \bar{\boldsymbol{\mu}}_m + r \cdot \sqrt{\lambda_{\max}} \boldsymbol{v}_{\max}$
    $\tilde{\boldsymbol{z}}_2 \leftarrow \bar{\boldsymbol{\mu}}_m - r \cdot \sqrt{\lambda_{\max}} \boldsymbol{v}_{\max}$
    Find two objects $i \neq j$, s.t.
        $i = \arg\min_{i \in [n], i \notin \mathcal{U}} p_i^m ||\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_1||_2,$
        $j = \arg\min_{j \in [n], j \notin \mathcal{U}} p_j^m ||\boldsymbol{x}_j - \tilde{\boldsymbol{z}}_2||_2$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{i, j\}$
    Obtain the response $\bar{y}$ from the user
    Update belief $\mathcal{P}_{m+1} \leftarrow \textsc{Update}(\mathcal{P}_m, \bar{y})$ using (5.3)
    $m \leftarrow m + 1$
until $t \in \{i, j\}$
```

5.3 Proofs

5.3.1 Theorem 2

Proof of Theorem 2. Without loss of generality, assume that $\sigma_\varepsilon = 1$. Let y be a binary random variable such that $p(y = 1 | \boldsymbol{w}, b, \boldsymbol{x}) = \Phi(\boldsymbol{x}^\top \boldsymbol{w} + b)$. Then,

$$
\begin{aligned}
&\arg\max_{(\boldsymbol{w},b)\in\mathcal{H}} I[\boldsymbol{x}; y \mid (\boldsymbol{w}, b)] \\
&\quad = \arg\max_{(\boldsymbol{w},b)\in\mathcal{H}} \left\{1 - \mathbb{E}_{\hat{\boldsymbol{x}}}[H(y \mid \boldsymbol{w}, b, \boldsymbol{x})]\right\} && (5.4) \\
&\quad = \arg\min_{(\boldsymbol{w},b)\in\mathcal{H}} \int_{\mathbb{R}^d} H\left[\Phi(\boldsymbol{x}^\top \boldsymbol{w} + b)\right] \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{x} \\
&\quad = \arg\min_{(\boldsymbol{w},b)\in\mathcal{H}} \int_{\mathbb{R}} H[\Phi(t)] \mathcal{N}(t; 0, \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}) dt && (5.5) \\
&\quad = \arg\max_{(\boldsymbol{w},b)\in\mathcal{H}} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}. && (5.6)
\end{aligned}
$$

In (5.4), we use (5.1) and the fact that, as the hyperplane is passing through $\boldsymbol{\mu}$,

$$H\left[\int_{\mathbb{R}^d} p(y=1|\boldsymbol{w},b,\boldsymbol{x})\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})d\boldsymbol{x}\right] = H(1/2) = 1.$$

In (5.5), we use the fact that $\boldsymbol{x}^\top\boldsymbol{w}+b \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})$, by properties of the Gaussian distribution. Finally, in (5.6), we start by noting that, for all c_1, c_2 such that $c_1/c_2 > 1$, $H[\Phi(c_1 t)] \le H[\Phi(c_2 t)]$ for all t with equality iff $t = 0$. Hence, if $\tilde{\sigma}^2 > \sigma^2$, then

$$\begin{aligned}
&\int_{\mathbb{R}} H[\Phi(t)]\mathcal{N}(t;0,\tilde{\sigma}^2)dt = \int_{\mathbb{R}} H[\Phi(\tilde{\sigma}t)]\mathcal{N}(t;0,1)dt \\
&< \int_{\mathbb{R}} H[\Phi(\sigma t)]\mathcal{N}(t;0,1)dt = \int_{\mathbb{R}} H[\Phi(t)]\mathcal{N}(t;0,\sigma^2)dt.
\end{aligned}$$

Therefore, maximizing $\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w}$ minimizes the expected entropy of y. □

5.3.2 Theorem 3

Proof of Theorem 3. To prove this theorem, first, let us assume that $d = 1$ for simplicity. We will explain how to generalize the result to any $d > 1$ later. Denote by x_t the location of the target object, and let $\mathcal{N}(\hat{x};\mu_m,\sigma_m)$ be the belief about the target's location after m observations. Without loss of generality, let $\sigma_\varepsilon^2 = 1$. In this case, the updates have the following form.

$$\begin{aligned}
\sigma_{m+1}^2 &= \sigma_m^2 + \beta_m\sigma_m^4, \\
\mu_{m+1} &= \mu_m + \alpha_m\sigma_m^2 \cdot z_m,
\end{aligned} \tag{5.7}$$

where $z_m \in \{\pm 1\}$ with $P(z_m = 1) = \Phi(x_t - \mu_m)$, where $\Phi(x)$ is the standard normal CDF, and

$$\begin{aligned}
\alpha_m &= c/\sqrt{\sigma_m^2 + 1}, \\
\beta_m &= -c^2/(\sigma_m^2 + 1), \\
c &= \sqrt{2/\pi}.
\end{aligned}$$

We start with a lemma that essentially states that σ_m^2 decreases as $1/m$.

Lemma 6. *For any initial $\sigma_0^2 > 0$ and for all $m \ge 0$, the posterior variance σ_m^2 can be bounded as*

$$\frac{\min\{0.1,\sigma_0^2\}}{m+1} \le \sigma_m^2 \le \frac{\max\{10,\sigma_0^2\}}{m+1}.$$

Proof. From (5.7), we know that

$$\sigma_{m+1}^2 = \left(1 - c^2 \frac{\sigma_m^2}{\sigma_m^2 + 1}\right)\sigma_m^2$$

First, we need to show that

$$f(x) = \left(1 - c^2 \frac{x}{x+1}\right)x$$

is increasing on $\mathbf{R}_{>0}$. This is easily verified by checking that

$$f'(x) = \frac{-(c^2-1)x^2 - 2(c^2-1)x + 1}{(x+1)^2} \geq 0,$$

for all $x \in \mathbf{R}_{>0}$ since $c^2 < 1$. Next, we consider the upper bound. Let $b = \max\{10, \sigma_0^2\}$. We will show that $\sigma_m^2 \leq b/(m+1)$ by induction. The basis step is immediate: by definition, $\sigma_0^2 \leq b$. The induction step is as follows, for $m \geq 1$. Because $f(x)$ is increasing on $\mathbf{R}_{>0}$,

$$\begin{aligned}
\sigma_m^2 &= \left(1 - c^2 \frac{\sigma_{m-1}^2}{\sigma_{m-1}^2 + 1}\right)\sigma_{m-1}^2 \\
&\leq \left(1 - c^2 \frac{b}{b+m}\right)\frac{b}{m} \leq \frac{b}{m+1} \\
\iff{}& 1 - c^2 \frac{b}{b+m} - \frac{m}{m+1} \leq 0 \\
\iff{}& b + m - c^2(bm + b) \leq 0 \\
\iff{}& m(1 - c^2 b) + b(1 - c^2) \leq 1 - b(\underbrace{2c^2 - 1}_{\approx 0.27}) \leq 0,
\end{aligned}$$

where the first inequality holds because $f(x)$ is increasing. The lower bound can be proved in a similar way. □

For completeness, we restate Theorem 3 for $d = 1$.

Theorem 3 (Case $d = 1$)**.** *Assuming Probit oracle model (equation 2.11) and updates (5.7), for any initial* μ_0 *and* $\sigma_0^2 > 0$ *and as* $m \to \infty$,

$$\sigma_m^2 \to 0,$$
$$\mu_m \to x_t$$

almost surely.

Proof. The first part of the theorem ($\sigma_m^2 \to 0$) is a trivial consequence of Lemma 6. The second part follows from the fact that our update procedure can be cast as the Robbins-Monro algorithm [41]. In the RM algorithm a function $M(\mu)$ is assumed to have a unique root $\mu^\star$. Then by observing the realizations of a random variable $N(\mu)$ that satisfies $\mathbb{E}[N(\mu)] = M(\mu)$, the iterative updates

$$\mu_{m+1} = \mu_m - a_m N(\mu_m)$$

assure convergence of μ_{m+1} to $\mu^\star$, provided that $M(\mu)$ is non-decreasing, $M'(\mu) > 0$, $N(\mu)$ is bounded, and

$$\sum_{m=1}^{\infty} a_m = \infty,$$
$$\sum_{m=1}^{\infty} a_m^2 < \infty.$$

We apply the RM result to function $M(\mu) := 2\Phi(x_t - \mu) - 1$, which has a unique root in $\mu = x_t$. We can see that $N(\mu_m) = z_m$ from (5.7) is then a stochastic estimate of M at μ_m, since $\mathbb{E}(z_m) = 2\Phi(\mu_m - x_t) - 1 = M(\mu_m)$. The remaining conditions to check are as follows.

- the learning rate $a_m = \alpha_m \sigma_m^2$ satisfies

$$\sum_{m=0}^{\infty} a_m \geq \frac{c \cdot \min\{\sigma_0^2, 0.1\}}{\sqrt{\sigma_0^2 + 1}} \sum_{m}^{\infty} \frac{1}{m} = \infty,$$
$$\sum_{m=0}^{\infty} a_m^2 \leq (c \cdot \max\{\sigma_0^2, 10.0\})^2 \sum_{m}^{\infty} \frac{1}{m^2} < \infty$$

- $|z_m| \leq 1$ for all m

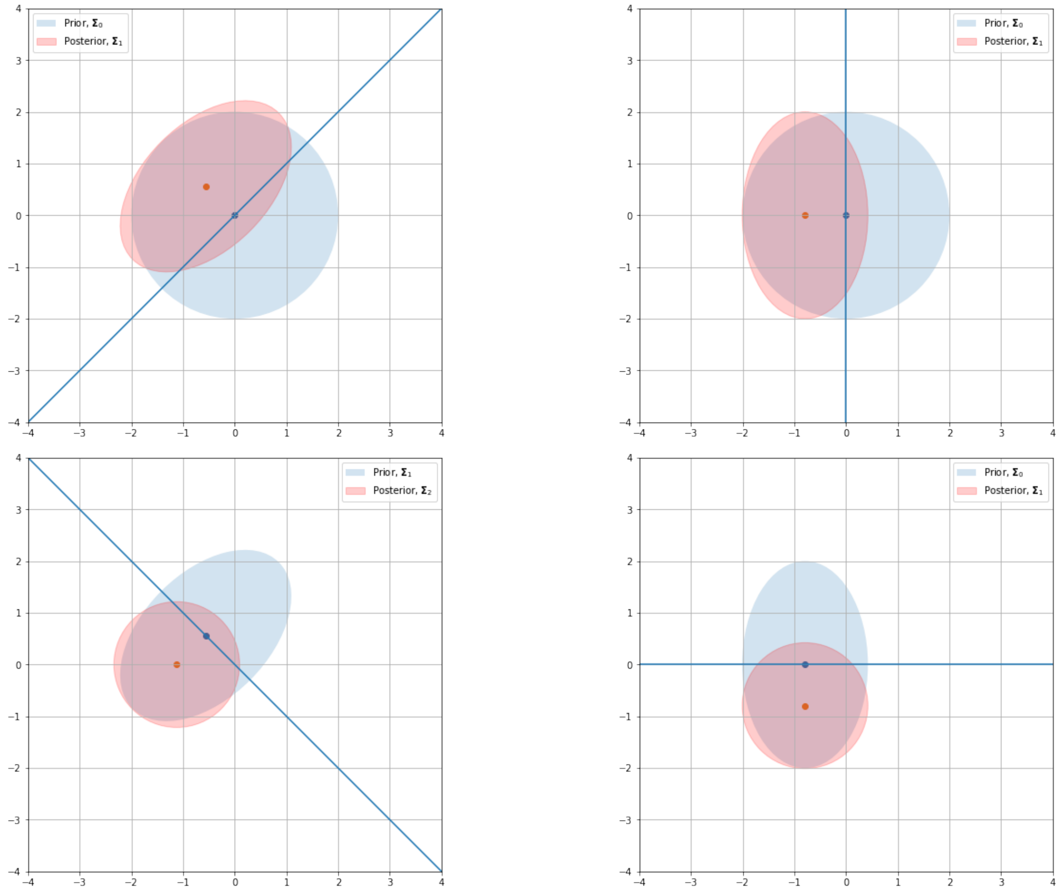
Almost sure convergence then follows directly from the results derived in [41] and [6]. □

Case $d > 1$. Assume we start with $\boldsymbol{\Sigma}_0 = \sigma_0^2 \boldsymbol{I}$. Consider the first d steps of GAUSSSEARCH and the corresponding hyperplanes $h_1, h_2, \ldots, h_d$. Every query Q_m at iteration m gives us information on the position of the target *only* along the line spanned by the largest eigenvector $\boldsymbol{v}_m$ of $\boldsymbol{\Sigma}_m$, which is orthogonal to the bisecting optimal hyperplane h_m. This can be seen, e.g., from the update rule of the covariance matrix

$$\boldsymbol{\Sigma}_{m+1} = \left(\boldsymbol{\Sigma}_m^{-1} + \tau \boldsymbol{w} \otimes \boldsymbol{w}^\top\right)^{-1},$$

which reveals that the precision matrix (i.e., the inverse of the covariance matrix) is affected only in the subspace spanned by $\boldsymbol{w}_{i_m, j_m}$. Since at each iteration $m \leq d$, the optimal hyperplane h_m is orthogonal to the first eigenvector of $\boldsymbol{\Sigma}_m$ and the variance of the initial $\boldsymbol{\Sigma}_0$ shrinks along the dimension defined by $\boldsymbol{v}_m$ only, leaving the other dimensions defined by $V \setminus \{\boldsymbol{v}_m\}$

untouched, these eigenvectors define a basis $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d\}$ in $\mathbb{R}^d$. Hence, after d iterations, the updated posterior covariance matrix is again diagonal $\boldsymbol{\Sigma}_d = \sigma_d^2 \boldsymbol{I}$. The same holds for the next d iterations and so on. For illustration, see Fig. 5.10.



(a) Hyperplanes are not aligned with the standard basis.

(b) Hyperplanes are aligned with the standard basis.

Figure 5.10: Illustration of the updates for $d = 2$. Originally, $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$. After one iteration (top), μ_0 is updated and $\boldsymbol{\Sigma}_0$ is shrunk only across the dimension spanned by the first eigenvector, $(-1, 1)^\top$ in (a), $(0, 1)^\top$ in (b), orthogonal to the optimal hyperplane h_1. After second iteration (bottom), μ_1 is updated and $\boldsymbol{\Sigma}_1$ is shrunk only across the second dimension spanned by the second eigenvector, $(-1, -1)^\top$ in (a), $(1, 0)^\top$ in (a), and orthogonal to the optimal hyperplane h_2. After $d = 2$ iterations, in both cases (a) and (b) the final covariance matrices $\boldsymbol{\Sigma}_2$ are both scalar and are identical, and the vectors $\boldsymbol{\mu}_2$ are equal up to a rotation given by the change of basis.

Therefore, without loss of generality, we can assume that the search procedure sequentially iterates over the dimensions in $\mathbb{R}^d$ in standard basis. After M iterations, let k be the current dimension of update and $(\mathbf{\Sigma}_M)_{(k,k)}$ be the k-th diagonal element of the covariance matrix $\mathbf{\Sigma}_M$ at step M. Then $(\mathbf{\Sigma}_{M+1})_{(k,k)} = (\mathbf{\Sigma}_M)_{(k,k)} + \beta_M(\mathbf{\Sigma}_M)^2_{(k,k)}$, where $\beta_M = -c^2/((\mathbf{\Sigma}_M)_{(k,k)}+1)$ as in (5.7), and the other entries of the matrix remained unchanged. Hence $(\mathbf{\Sigma}_M)_{(k,k)} \to 0$ and $\mathrm{Tr}(\mathbf{\Sigma}_m) \to 0$ as $m \to \infty$. Now since the update of $\boldsymbol{\mu}_M$ is also affecting only a single dimension k of $\boldsymbol{\mu}_M$, similarly as in the one dimensional case, we have $(\boldsymbol{\mu}_{M+1})_k = (\boldsymbol{\mu}_M)_k + \alpha_M(\mathbf{\Sigma}_{M+1})_{(k,k)} z_M$, where $\alpha_M = c/\sqrt{((\mathbf{\Sigma}_M)_{(k,k)}+1)}$ as in (5.7). Now since every d iterations we make an update in each dimension exactly one time, $\boldsymbol{\mu}_m \to \boldsymbol{x}_t$.

For the general choice of the initial $\mathbf{\Sigma}_0$, similar argument applies. We can assume that the matrix $\mathbf{\Sigma}_0$ is diagonal, otherwise we can just re-parameterize the space $\mathbb{R}^d$ by using a rotation matrix. After that, since at each iteration the updates of $\mathbf{\Sigma}_m$ and $\boldsymbol{\mu}_m$ modify only a single dimension (in standard/eigen basis), we can view updating each dimension of $(\boldsymbol{\mu}_m, \mathbf{\Sigma}_m)$ as a single step of a one-dimensional search procedures. The ambiguity in the choice of the optimal hyperplane is possible only if the entries of $\mathbf{\Sigma}_m$ are not unique, which is handled in the case $\mathbf{\Sigma}_0 = \sigma_0^2 \boldsymbol{I}$ above.

□

5.3.3 Lemma 2, Lemma 3, and Theorem 4

Proof of Lemma 2. We show an equivalent statement: There exists a coupling $\tilde{X}_s$ and $\tilde{Z}_s$, such that $\forall s \geq 0$ $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_s) \leq \tilde{Z}_s] = 1$ and the distributions are identical, $F_{X_s} = F_{\tilde{X}_s}, F_{Z_s} = F_{\tilde{Z}_s}$ This is done via induction.

Induction base:
For $s = 0$ we are looking at a constant, which is the same in both cases: $Z_0 = z(\boldsymbol{x}_t, X_0)$. Immediately $\mathbf{P}[z(\boldsymbol{x}_t, X_0) = \tilde{Z}_0] = 1$

Induction step:
We are given a random variable $\tilde{X}_s$ which has the same distribution as X_s and we know that $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_s) \leq \tilde{Z}_s] = 1$. We will now construct two random variables $\tilde{X}_{s+1}$ and $\tilde{Z}_{s+1}$ for which it holds that $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{Z}_{s+1}] = 1$.

Let $u \sim U[0,1]$ be a sample from the uniform distribution on $(0,1)$. We use this to couple the two random walks. Depending on u and the current state of the random walk is $\tilde{X}_s$, the following transition is taken:

- $\tilde{X}_s$ is green. This means $z(\boldsymbol{x}_t, \tilde{X}_s) = 0$
    - if $u \leq p_d(\tilde{X}_s, \boldsymbol{x}_t)$, then proceed to a green child. This means $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 0$
    - if $p_d(\tilde{X}_s, \boldsymbol{x}_t) < u \leq p_d(\tilde{X}_s, \boldsymbol{x}_t) + q_u(\tilde{X}_s, \boldsymbol{x}_t)$, then backtrack to the parent region. This

  means $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 0$

  - else $p_d(\tilde{X}_s, \boldsymbol{x}_t) + q_u(\tilde{X}_s, \boldsymbol{x}_t) < u \leq 1$, stray to a red child region. Now we have $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 1$

- $\tilde{X}_s$ is red. This means $z(\boldsymbol{x}_t, \tilde{X}_s) > 0$
  - if $u \leq p_u(\tilde{X}_s, \boldsymbol{x}_t)$, then backtrack to the parent region. This means $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) - z(\boldsymbol{x}_t, \tilde{X}_s) = -1$
  - if $p_u(\tilde{X}_s, \boldsymbol{x}_t) \leq u < p_r\tilde{X}_s, \boldsymbol{x}_t) + p_u(\tilde{X}_s, \boldsymbol{x}_t)$,then recover by proceeding to a green child region. This means that $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 0$. Recovering is only possible if one of the child regions contains the target. We know that the parent of a region is a superset of all the child regions $u(X) \supset \bigcup_{X_c \in D(X)} X_c$. Therefore, whenever a recovery transition is possible, backtracking must likewise lead to a green region. This shows that recovery is only possible when $z(\boldsymbol{x}_t, \tilde{X}_s) = 1$. Therefore we have shown that $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) - z(\boldsymbol{x}_t, \tilde{X}_s) = -1$
  - else $p_r\tilde{X}_s, \boldsymbol{x}_t) + p_u(\tilde{X}_s, \boldsymbol{x}_t) \leq u \leq 1$, then proceed to a red child. This means $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) - z(\boldsymbol{x}_t, \tilde{X}_s) \in \{0, 1\}$

We now construct a coupled variable $\tilde{D}$ such that $\tilde{Z}_{s+1} = \tilde{Z}_s + \tilde{D}$. Since $\tilde{Z}$ is a random walk on natural numbers, with a self-loop at 0, we need to distinguish between two scenarios:

- $\tilde{Z}_s > 0$
  - if $u \leq \frac{1+b}{2}$, then $\tilde{D} = -1$
  - else, $\tilde{D} = 1$
- $\tilde{Z}_s = 0$
  - if $u \leq \frac{1+b}{2}$, then $\tilde{D} = 0$
  - else, $\tilde{D} = 1$

We will now show that the construction of $\tilde{D}$ ensures that $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{D} + \tilde{Z}_s] = 1$

Case 1, $\tilde{Z}_s = 0$:

The induction assumptions imply $z(\boldsymbol{x}_t, \tilde{X}_s) = 0$, which in turn implies that $\tilde{\mathcal{X}}_s$ is a green region. In this case we know that $\tilde{D} \in \{0, 1\}$ and $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \in \{0, 1\}$.

It holds that $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 1$ iff $p_d(\tilde{X}_s, \boldsymbol{x}_t) + q_u(\tilde{X}_s, \boldsymbol{x}_t) = 1 - q_s(\tilde{\mathcal{X}}_s, \boldsymbol{x}_t) < u$. It holds that $\tilde{D} = 1$ iff $\frac{1+b}{2} < u$. From assumption (A1) we know that for all possible regions X and targets $\boldsymbol{x}_t$,

$1 - q_s(X, \boldsymbol{x}_t) > \frac{1+b}{2}$. Therefore $\tilde{D} = 0 \implies z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 0$. Therefore $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{D} + \tilde{Z}_s \mid \tilde{Z}_s = 0] = 1$

Case 2, $\tilde{Z}_s > 0, z(\boldsymbol{x}_t, \tilde{X}_s) = 0$:

Again, $\tilde{X}_s$ is a green region. So we know that $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \in \{0, 1\}$, and $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 1$ iff $p_d(\tilde{X}_s, \boldsymbol{x}_t) + q_u(\tilde{X}_s, \boldsymbol{x}_t) = 1 - q_s(\tilde{\mathcal{X}}_s, \boldsymbol{x}_t) < u$.

Since $\tilde{Z}_s > 0$ and $z(\boldsymbol{x}_t, \tilde{X}_s) = 0$ we know $\tilde{D} \geq 0 \implies z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{Z}_s + \tilde{D}$. We only need to analyse the case of $\tilde{D} = -1$. We know that $\tilde{D} = -1 \implies u < \frac{1+b}{2}$. Using assumption (A1), $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 1 \implies u > 1 - p_s(\tilde{X}_s, \boldsymbol{x}_t) > 1 - \frac{1-b}{2} = \frac{1+b}{2}$. This is a contradiction. We now know that $\tilde{D} = -1 \implies z(\boldsymbol{x}_t, \tilde{X}_{s+1}) = 0$. Therefore $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{D} + \tilde{Z}_s \mid \tilde{Z}_s > 0, z(\boldsymbol{x}_t, \tilde{X}_s) = 0] = 1$

Case 3, $\tilde{Z}_s > 0, z(\boldsymbol{x}_t, \tilde{X}_s) > 0$:

$\tilde{D} = -1$ implies $u < \frac{1+b}{2}$. From Assumption (A1) we know that $\forall X, \boldsymbol{x}_t : \frac{1+b}{2} \leq p_u(X_s, \boldsymbol{x}_t) + p_r X_s, \boldsymbol{x}_t)$. Therefore the event $\tilde{D} = -1$ implies $z(\boldsymbol{x}_t, \tilde{X}_{s+1}) - z(\boldsymbol{x}_t, \tilde{X}_s) = -1$. Therefore $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{D} + \tilde{Z}_s \mid \tilde{Z}_s > 0, z(\boldsymbol{x}_t, \tilde{X}_s) > 0] = 1$

Hence $\mathbf{P}[z(\boldsymbol{x}_t, \tilde{X}_{s+1}) \leq \tilde{D} + \tilde{Z}_s] = 1$. □

Proof of Lemma 3. Let $\tau_{Z=N} = \inf\{s > 0 \mid Z_s, Z_0 = N\}$ be the stopping time of Z_s reaching 0, starting from N. We have shown that the random walk Z can be used as a stochastic upper bound. Therefore we know $\mathbb{E}[\tau_X] \leq \mathbb{E}[\tau_{Z=1}]$.

We will now calculate the stopping time of Z. Indeed,

$$\mathbb{E}[\tau_{Z=1}] = \frac{1+b}{2} + \frac{1-b}{2}(2\mathbb{E}[\tau_{Z=1}] + 1),$$

because at $N = 1$ we either decrement with probability $\frac{1+b}{2}$ or increment with probability $\frac{1-b}{2}$ and then have to wait twice the time of $\mathbb{E}[\tau_{Z=1}]$ in order to return to $N = 0$, since Z_s is a random walk. Solving this equation we get $\mathbb{E}[\tau_{Z=1}] = \frac{1}{b}$, hence $\mathbb{E}[\tau_X] \leq \frac{1}{b}$. □

Proof of Theorem 4. Let us first prove the first statement, $\mathbf{P}(\boldsymbol{x}_t \in u(X_s, k)) > 1 - \delta$. We show that for any $k > 0$

$$\mathbf{P}[z(\boldsymbol{x}_t, X_s) > k] \leq \left(\frac{1-b}{1+b}\right)^k$$

Under the assumption (A1) we have shown $z(\boldsymbol{x}_t, X_s) \preceq_{st.} Z_s$. The definition of stochastic ordering $\mathbf{P}[z(\boldsymbol{x}_t, X_s) \geq k] \leq \mathbf{P}[Z_s \geq k]$ is equivalent to $\mathbf{P}[z(\boldsymbol{x}_t, X_s) \leq k] \geq \mathbf{P}[Z_s \leq k]$.

We will now show the claim of the lemma for Z_s, the same statement for $z(\boldsymbol{x}_t, X_s)$ follows immediately. Our proof is an induction for $\mathbf{P}[Z_s > k] \leq (\frac{1-b}{1+b})^k$. The property holds trivially for $s = 0, k \geq 0$ and $k = 0, s \geq 0$. Assume that the property holds for a given s and for all k. For any $k \geq 1$, we have

$$\begin{aligned}\mathbf{P}[Z_{s+1} > k] &= \frac{1+b}{2}\mathbf{P}[Z_s > k+1] + \frac{1-b}{2}\mathbf{P}[Z_s > k-1] \\ &\leq \frac{1+b}{2}(\frac{1-b}{1+b})^{k+1} + \frac{1-b}{2}(\frac{1-b}{1+b})^{k-1} = (\frac{1-b}{1+b})^k\end{aligned}$$

Hence, at any time s, the probability of needing more than k backtracks until we reach a green region from X_s is less than $(\frac{1-b}{1+b})^k$. We solve $\delta = (\frac{1-b}{1+b})^k$ for k. Then we know that with probability of at least $1-\delta$, the target must be in the k-th ancestor of X_s. This is the region that we propose as the result of our search process.

We now need to show that the expected depth of this region increases at a constant rate. Since k is a constant that only depends on the desired rate of error δ and does not change over time, it suffices to show that the expected depth of X_s increases at a constant rate.

We make use of the Markovian property of X_s. Without loss of generality, we assume that we are currently at time $s = 0$. Additionally we assume that the current region X_0 is green. When the execution of the algorithm begins, this is true since $\boldsymbol{x}_t \in \Omega$.

We now define a stopping time $s' = \inf\{s > 0 \mid \boldsymbol{x}_t \in X_s, X_0\}$, as the next time at which our algorithm visits a green region. We will show that this stopping time is finite and that this next green node is, in expectation, at a higher depth. The analysis then becomes recursive. Specifically, we will show:

- There is a constant $C_d > 0$, such that $\mathbb{E}[D(X_{s'}) - D(X_0)] > C_d$
- There is a constant $C_s < \infty$, such that $\mathbb{E}[s'] < C_s$

Starting from the green region X_0, the following transitions are possible:

- With probability $p_d(X_0, \boldsymbol{x}_t)$, the search proceeds to a green child. In this case we stop immediately, $s' = 1$ and $D(X_1) - D(X_0) = 1$.
- With probability $q_u(X_0, \boldsymbol{x}_t)$, the search backtracks. Since the parent of a green region must be green as well, we also stop immediately, $s' = 1$. Since backtracking looses two levels of depth, we have $D(X_1) - D(X_0) = -2$.
- With probability $q_s(X_0, \boldsymbol{x}_t)$, the search strays.

The last case requires further analysis. Following Lemma 3, we know that the expected stopping time after straying is upper bounded by $\mathbb{E}[\tau_{Z=1}] = \frac{1}{b}$. Every backtracking decision must always undo at least one proceed decision. This means that, in the worst case scenario, exactly half the steps until s' are proceed and half are backtrack decisions. A pair of proceed and backtrack decisions first gains one level of depth and then looses two. Therefore, conditioned on the assumption that we have left X_0 by straying, the expected new depth is bounded by $\mathbb{E}[D(X_{s'}) - D(X_0) | \text{we strayed from } X_0] < -1\frac{1}{2}(1 + \mathbb{E}[\tau_{Z=1}]) = -\frac{1}{2}(1 + \frac{1}{b}) = -\frac{b+1}{2b}$.

In expectation, the number of timesteps that passes between consecutive green regions is $\mathbb{E}[s'] \leq q_u + p_d + q_s(1 + \mathbb{E}[\tau_{Z=1}]) = q_u + p_d + q_s\frac{b+1}{b}$. This means at time s we have, in expectation, visited $\frac{s}{q_u + p_d + q_s\frac{b+1}{b}}$ green nodes.

The expected depth of each consecutive green node is $p_d - 2q_u - q_s\frac{b+1}{2b}$ levels higher than its predecessor. Due to Assumption (A1) we know that this is strictly positive.

In expectation, the last green node that we have visited is at a depth of $\frac{s}{q_u + p_d + q_s\frac{b+1}{b}}\left(p_d - 2q_u - q_s\frac{b+1}{2b}\right)$. We also know an upper bound on the expected number of steps between green nodes: For any given state X_s of the search algorithm, we know that in expectation, we have taken at most $\mathbb{E}[s'] \leq q_u + p_d + q_s\frac{b+1}{b}$ steps since the last green region. We are interested in a bound of the depth of the current region. In the worst case scenario, all of these steps were backtracks. This leads to $\mathbb{E}[D(X_s)] \geq \frac{s}{q_u + p_d + q_s\frac{b+1}{b}}\left(p_d - 2q_u - q_s\frac{b+1}{2b}\right) - 2(q_u + p_d + q_s\frac{b+1}{b})$ (which is a linear function of s).

□

5.3.4 Lemma 4 and Theorem 5

Proof of Lemma 4. In the query Q we ask the oracle which $\vec{\mathbf{0}}$ or $\boldsymbol{x}_q = (1+d)\mathbf{e}$ is closer to target $\boldsymbol{x}_t$. We denote the probability of $\vec{\mathbf{0}}$ inside of X being chosen by the oracle as $\mathbf{P}[\vec{\mathbf{0}} \succ \boldsymbol{x}_q | \boldsymbol{x}_t] = \mathbf{P}[X \succ F | \boldsymbol{x}_t]$

We will now show that there are two probabilities $p_X > p_F > 0$ such that:

- $\boldsymbol{x}_t \in X \implies \mathbf{P}[X \succ F | \boldsymbol{x}_t] \geq p_X$
- $\boldsymbol{x}_t \in F \implies \mathbf{P}[X \succ F | \boldsymbol{x}_t] \leq p_F$

This immediately allows the use of a binomial test for the hypothesis (H). Repeating the query Q enough times, we can achieve level of accuracy $1 - \delta$ of the test.

The possible target location $\boldsymbol{x}_c$ inside X for which $\mathbf{P}[X \succ F | \boldsymbol{x}_t = \boldsymbol{x}_c]$ is smallest if

$\mathbf{x}_c = \arg\min_{\boldsymbol{x}_t \in X} \mathbf{P}[X \succ F | \boldsymbol{x}_t]$ and it lies in a corner of the hypercube. For any parameterization of γ-CKL we can now explicitly calculate the lower bound: $p_X = \mathbf{P}[X \succ F | \boldsymbol{x}_t = \boldsymbol{x}_c]$.

We define the following distances:

$$\begin{aligned} d_c &= ||\vec{\mathbf{0}} - \mathbf{x}_c|| = \sqrt{d} \\ d_{qc} &= ||\mathbf{x}_q - \mathbf{x}_c|| = \sqrt{d^2 + d - 1} \\ d_q &= ||\vec{\mathbf{0}} - \mathbf{x}_q|| = d + 1 \end{aligned}$$

The ratio of distances between $\mathbf{x}_c$ and the two query points is $\frac{||\vec{\mathbf{0}} - \mathbf{x}_c||}{||\mathbf{x}_q - \mathbf{x}_c||} = \frac{d_c}{d_{qc}}$. We know that any point $\mathbf{x}'$ that induces the same outcome probability must have the same ratio of distances: $\mathbf{P}[\vec{\mathbf{0}} \succ \mathbf{x}_q | \boldsymbol{x}_t = \mathbf{x}'] = \mathbf{P}[\vec{\mathbf{0}} \succ \mathbf{x}_q | \boldsymbol{x}_t = \mathbf{x}_c] \iff \frac{||\vec{\mathbf{0}} - \mathbf{x}'||}{||\mathbf{x}_q - \mathbf{x}'||} = \frac{d_c}{d_{qc}}$. The points with equal ratio of distances define the surface of a hypersphere, as previously discussed.

Out of these points, the one with the smallest distance to $\vec{\mathbf{0}}$, i.e. to the center of X, lies on the line segment between $\vec{\mathbf{0}}$ and $\mathbf{x}_q$. The point with the largest distance to $\vec{\mathbf{0}}$ lies on the ray from $\mathbf{x}_q$ to $\vec{\mathbf{0}}$, at $\vec{\mathbf{0}} - \mathbf{e}\frac{d + \sqrt{d^3 + d^2 - d}}{d - 1}$. This can be easily verified by solving the condition of equal ratio for the x coordinate. We know that all points $\mathbf{x}'$ with a ratio that is strictly larger than $\frac{d_c}{d_{qc}}$ must induce a smaller probability $\mathbf{P}[\vec{\mathbf{0}} \succ \mathbf{x}_q | \boldsymbol{x}_t = \mathbf{x}'] < p_X$. The point farthest away from $\vec{\mathbf{0}}$ which still has this ratio lies at $\vec{\mathbf{0}} - \mathbf{e}\hat{r}$, with $\hat{r} = \frac{d + \sqrt{d^3 + d^2 - d}}{d - 1}$. This allows us to specify an uncertainty region. Let $r_u = \hat{r} + 1$. The probability p_F can now be explicitly computed (for any parametrization of γ-CKL) as $p_F = \mathbf{P}[\vec{\mathbf{0}} \succ \mathbf{x}_q | \boldsymbol{x}_t = \vec{\mathbf{0}} - \mathbf{e}(\hat{r} + 1)] < p_X$.

□

Proof of Theorem 5. The tiling $\mathcal{T}(S, r_c)$ contains K cells, this is also the number of hypothesis tests that we conduct. Conditional on $\boldsymbol{x}_t$, the oracle replies are independent, and therefore the test outcomes are independent. We assume that the probability of error for any one of the tests is δ. The probability of no error occurring across all tests is therefore $(1 - \delta)^K$. We make a decision based on the test results assuming each test outcome is correct. This way, the error probability of our decision is $\hat{\delta} = 1 - (1 - \delta)^K$. Lemma 4 ensures that we can adjust the hypothesis test for any desired probability of error δ. It is therefore always possible to choose a number of query observations (depending on the dimensionality and the parameters of the choice model) that leads to any desired $\hat{\delta}$.

In the following we assume that all hypothesis tests have provided correct information. This means, that (H) has not been rejected for the cells in class (A) and it has been rejected for the cells in class (C). We create a bounding box $\mathcal{B}$ around all cells for which hypothesis (H) has not been rejected. From Lemma 5 we know that this bounding box has an edge length of at most

$\frac{1}{4}$. We now look at all possible locations for $\boldsymbol{x}_t$ and verify that the decision criterion must lead to a correct decision.

Case 1, $\boldsymbol{x}_t \notin (S \cup X)$:

The bounding box can't overlap with X. Therefore we backtrack. This is the correct decision.

Case 2, $\boldsymbol{x}_t \in X$:

There is a cell in class (A), which overlaps with X. For this cell, the hypothesis (H) has not been rejected. This means that the bounding box must overlap with X. Also, we know that the bounding box has an edge length of less than $\frac{1}{4}$. This means that it can overlap with at most 2 of the tiles in $\mathcal{T}(S, 1/4)$. This means that there is a child region in $D(X)$ which fully contains the bounding box. Our decision criterion proceeds to this child. And we know that the bounding box must contain the target (since we're assuming that all hypothesis tests have returned correctly). This ensures that we are proceeding to a green region.

Case 3, $\boldsymbol{x}_t \in S$:

The target is not in the current region, i.e. X is a red region. If the bounding box happens to not overlap with X, we backtrack, which is considered a correct decision. If the bounding box happens to overlap with X, then we know that there must be a child which fully contains the bounding box. Our decision criterion proceeds to this child region. We also know that the bounding box contains the target. So we are proceeding to a green region. This is a recovery transition, and it is also considered a correct decision.

We have shown that, under the assumption that all hypothesis tests have provided correct information, the decision criterion leads to a correct transition. Our assumption on the hypothesis tests holds with probability $1 - \hat{\delta}$.

If some hypothesis tests are erroneous, then we can see inconsistent behaviour. For example, it is possible that the bounding box is too large, and overlaps with multiple child regions, or overlaps with both X and $\Omega \setminus S$. We assume that in this case, we backtrack. This can be the wrong decision, but it will happen with at most probability $\hat{\delta}$. $\square$

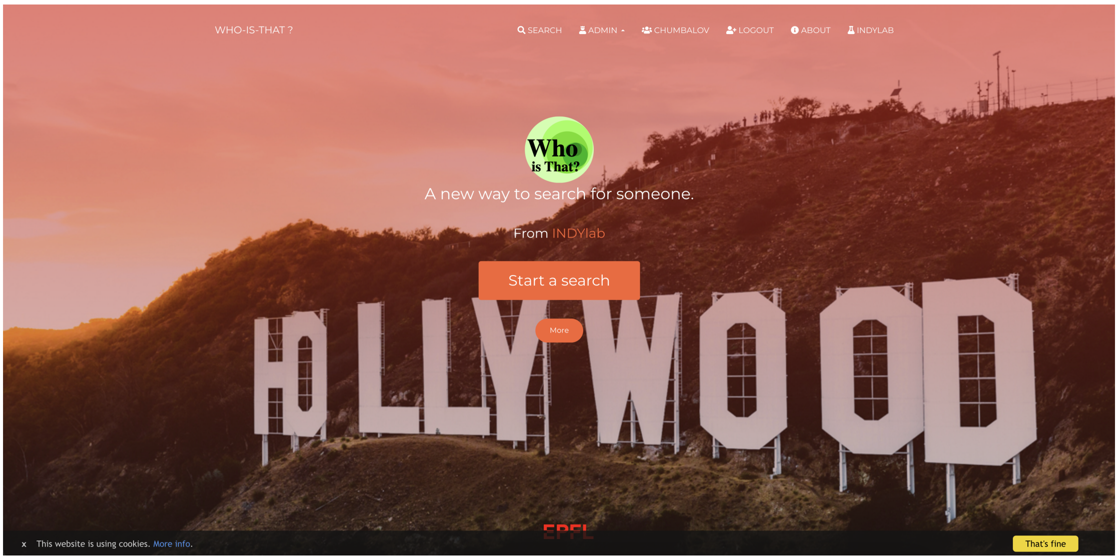
6 Experiments with Human Oracles

In this Chapter, we present the results of the search experiments involving human oracles. First, we describe the results of applying LEARN2SEARCH in an experiment on searching for movie actors involving real users. Then, we present the results of comparing the performances of γ-CKLSEARCH versus GAUSSSEARCH.

For all the experiments in this Chapter we built a website http://who-is-th.at/, with a convenient UI for searching using comparisons, see Fig. 6.1-6.3. In the scope of experiments, Fig. 6.2, a user is presented with a target actor, which is pre-selected by the system but is kept unknown to the search engine. A user can also search for any actor he wants using another webpage, Fig. 6.3.

In this system, a user searches for an actor by comparing faces: at each search iteration, the user is shown 4 pictures of faces to chose from, and can then click on the face that looks the most like the target. Once the user makes his choice, 3 unordered triplet comparisons are recorded in the system logs. This process repeats until the target is among the 4 samples. Once the user's target appears on the screen, the user clicks on a button to see details about the target, which ends the search.

All algorithms presented in this thesis were implemented on the website. Our search algorithms, GAUSSSEARCH and γ-CKLSEARCH are designed to output queries in a form of pairs of objects and were slightly modified in order to output two pairs of objects: GAUSSSEARCH considered first two main hyperplanes instead of just one, and, similarly, in γ-CKLSEARCH we choose two points on the first main axis and two points on the second main axis.



A new way to search for someone.

From INDYlab

x This website is using cookies. More info.

Figure 6.1: Main website.

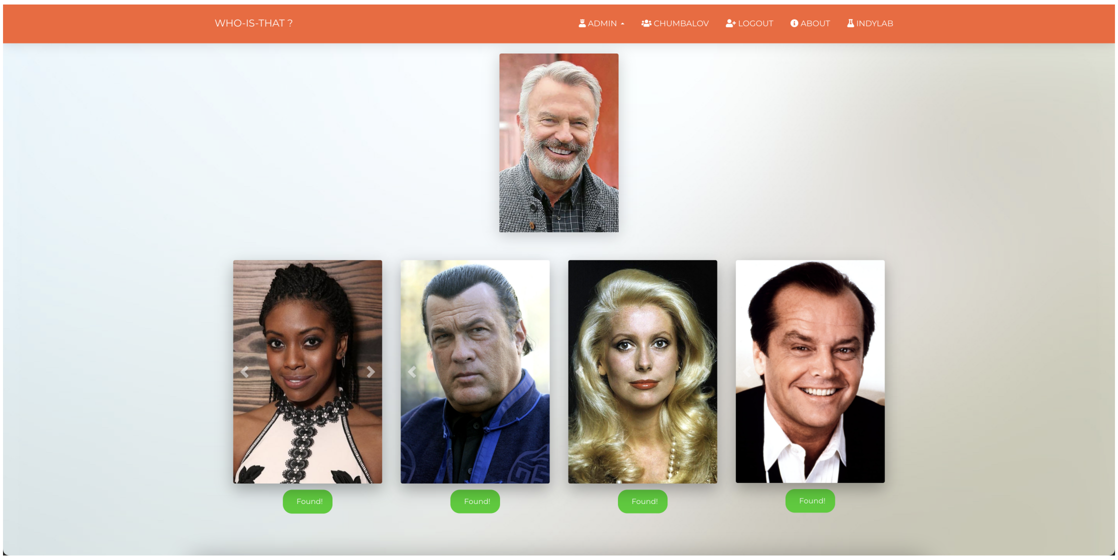
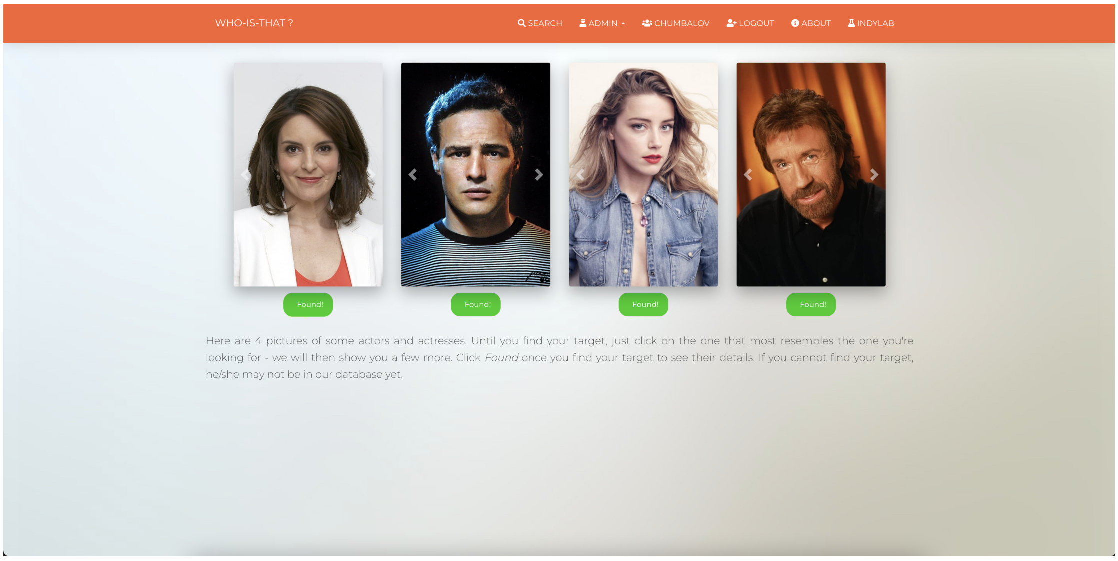


Figure 6.2: An example of the UI in the experiments; target actor is randomly sampled and is the participant has to find him.



Here are 4 pictures of some actors and actresses. Until you find your target, just click on the one that most resembles the one you're looking for - we will then show you a few more. Click *Found* once you find your target to see their details. If you cannot find your target, he/she may not be in our database yet.

Figure 6.3: An example of the UI for searching for any actor; target actor is only mentally known by the user. A showcase of our system that can be used in practice.

6.1 LEARN2SEARCH with Probit Model

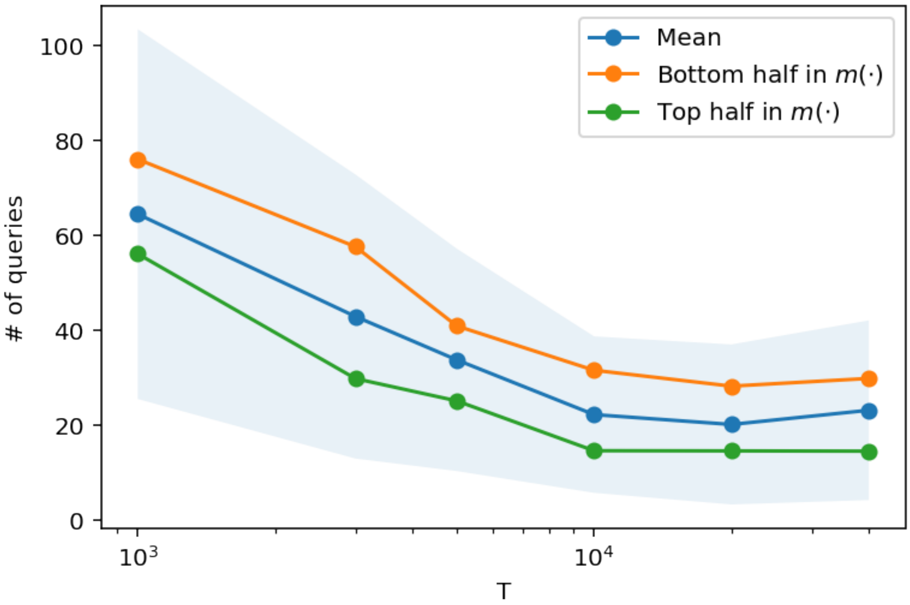


Figure 6.4: Empirical search cost, averaged over 5 human subjects, for a database with $n = 513$ faces of movie actors as a function of the number of triplets T available in the database from the previous searches. The averages in the top and bottom groups in the number of triplets per target, m(actor) of targets are shown individually as well: clearly, the denser group (green) has lower search cost. The blue shaded area shows one std around the average search cost. Note that the y-axis is the number of queries (each query features four pictures). Each choice is broken down into 3 pairwise comparison outcomes.

We present the results of an experiment involving human oracles, in order to validate the practical relevance of Probit and GAUSSSEARCH. We collected $n = 513$ photographs of the faces of well-known actors and actresses, and used our web-based search interface, Fig. 6.1, implementing the LEARN2SEARCH method over this dataset.

We have collected slightly more than 40’000 comparison triplets from users of the site (which include both project participants as well as outside users, who explored the service out of curiosity). The system is *blind*, in that we do not extract any explicit features of the faces (shape, eye color, etc.)

We try to answer the following two questions: (i) how efficient are searches? and (ii) how does the search cost depend on the size of the training set? For this, we recruited 5 subjects on the university campus on a voluntary basis, whose task it was to find the face of an actor, which has been sampled uniformly from the n possible targets. For each search, we first sampled $T \in \{1000, 3000, 5000, 10k, 20k, 40k\}$ triplets, uniformly from the source $40k+$ triplets, and then generated the embedding as described in Chapter 4. For each value of T, each subject performed 10 searches using our search system with the objects embedding fixed and shared across individuals. The results are shown in Fig. 6.4. Observe that for the smallest T, the average search cost is essentially equal to the random strategy (which has expected cost $\frac{1}{2} \cdot \frac{n}{4}$). As T increases, the embedding becomes more meaningful, and the search cost drops significantly. In our experiments, the median time-to-answer is around 5 seconds. Thus, once we have ~10k triplets, the time until the target is found is ~2 minutes. We observe small differences in the time-to-answer as a function of the quality of the embedding: with random embeddings, users tend to answer slightly faster (probably because the images are very different to the target).

An additional comment on the variance in search cost is in order. In Chapter 4, we defined $m(i)$ to be the number of triplets in T that object i is part of (in any position). In our full experimental dataset, $m(i)$ is heavily skewed, see Fig. 4.1 for Movie Actors in Chapter 4, because

1. objects were added to the system gradually over time, and
2. the distribution over targets follows popularity, which is heavily skewed itself.

Concretely, this ranges from m(Halle Berry) = 1567 to m(Kumail Nanjiani) = 16. It is natural to suspect that a target t with larger $m(t)$ is easier to find, because its embedding is more precise relative to other objects. Indeed, our results bear this out: in Fig. 6.4, we break out the search cost for the top and bottom half of targets separately. This suggests that if we could collect further data, including for the sparse targets (low $m(t)$), the asymptotic search cost would be reduced further. In summary, our experiment shows that LEARN2SEARCH is able to extract an embedding in the latent setting that appears to align with visual features that human oracles rely on to find a face, and is able to navigate through this embedding to locate a target efficiently.

6.2 γ-CKLSEARCH vs GAUSSSEARCH

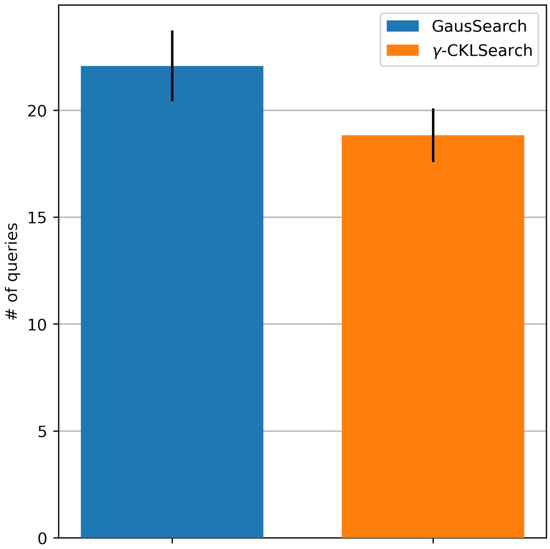


Figure 6.5: Face search experiment with real users: Comparison of the average number of queries a user needs to answer until he finds his target using GAUSSSEARCH and γ-CKLSEARCH.

Next we conducted a search experiment with real users in order to compare the actual performance of γ-CKLSEARCH against GAUSSSEARCH in a real world application setting. As previously, we used our web application in which a user can search for movie actors by comparing faces, Fig. 6.1. Our database again consisted of $n = 513$ face pictures of famous movie actors. At each step of a search, the user is presented with 4 pictures of faces of yet unseen actors and is asked to choose among them the one that resembles his target the most. The search completes once the user finds his target, i.e. when the picture of the target's face appears in one of the 4 displayed pictures. An embedding of actors faces has been learned individually for each algorithm from the triplets collected prior to the experiment.

To ensure fair payment, we have estimated the duration of our study in trial runs. Participants were paid the equivalent of 20USD per hour. Our study is designed with controlled randomization. Each user sees a target at most once. Each target is searched for twice, once with algorithm γ-CKLSEARCH and once with GAUSSSEARCH. This corresponds to an across subject design and reduces object-related bias. To reduce user-related bias, we also use a within subject design, each user performs the same amount of searches with the two algorithms. The order in which searches are seen is random. Users are not aware of the algorithm they are testing.

In total, we recruited 24 human participants through ad flyers on university campus. We presented 10 different target actors to each participant and asked to search for them. We performed an A/B testing by privily using γ-CKLSEARCH in the backend of the search interface for one half of the searches and GAUSSSEARCH for the other half. The target actors were chosen uniformly at random from a filtered set of 387 actors that had at least 100 associated triplets in $\mathcal{T}$. The A/B testing assignments were designed such that almost all of the chosen targets were paired exactly once with γ-CKLSEARCH and exactly once with GAUSSSEARCH. Overall the participants did 207 searches with 129 unique targets, 104 searches using GaussSearch and 103 searches using γ-CKLSEARCH. 19 participants completed all 10 searches, 1 participant completed 7 searches, 1 participant completed 5 searches, 1 participant completed 3 searches, and 2 participants completed only 1 search. We did not observe any particularly strong outliers in the number of search rounds in the searches done by participants who did not finish all 10 searches. Based on the initial trial runs we ended up with the following choice of hyperparameters: $D = 5$, $\gamma = 3$, and $\sigma_e = 0.1$.

A bar plot on the performance of the search algorithms is presented in Fig. 6.5. Our search method based on the γ-CKL model outperform GAUSSSEARCH in terms of query complexity: with γ-CKLSEARCH a user needs on average **18.83** queries to find the target, while with GAUSSSEARCH he needs on average **22.08** queries. Moreover, γ-CKLSEARCH algorithm tend to ask queries that are cognitively easier for humans to answer: on average participants were spending 11.62 seconds to decide on a query during a search with γ-CKLSEARCH versus 13.19 seconds for a query from GAUSSSEARCH.

7 Conclusion

In this thesis we address a task of searching in a database of objects using comparisons. It is a powerful way of searching for information when the target object is problematic to describe explicitly. We are interested in search schemes that interact with the user via asking comparison queries, and which refine their set of proposed search target candidates at each round, depending on the user answers.

- The central component of any comparison-based search algorithm is the oracle model. In Chapter 2 we introduce two new oracle models. The first one, Probit, is based on the hyperplane between the query points. The second one, γ-CKL, is a generalization of a well-known comparison model CKL. We show how the parameter γ helps to deal with the curse of dimensionality without breaking the scale-free property of the original model.
- An oracle model is useful only when it adequately reflects the reality. In Chapter 4 we empirically demonstrate that our new models predict human choices very well, they are either on par or outperform the existing state-of-the-art models across different real world datasets. The introduction of the γ parameter becomes the key improvement over original CKL model. The movie actors embeddings learned using these models give an interesting insight on the potential order of the importance of the face features people mentally go in, when comparing faces. Finally, we introduce a method to learn a variational embedding of objects using comparisons that helps to account for the uneven distribution of the amount of the comparison data among the objects that is useful for the search algorithm in a setting when objects features are not accessible.
- In Chapter 5, we develop search algorithms for our new oracle models. For the Probit model, its hyperplane nature gives a rise to a very efficient search scheme with a Gaussian prior that asks the most informative queries. We showed that the scheme

is guaranteed to converge to the target under the Probit noise model. This scheme is suitable for large databases, because it has logarithmic computational complexity and is not inferior to the state-of-the-art methods in query complexity. We also explore an important scenario when the objects features, that are mentally used by the user when answering system's queries, are not accessible by the system. In that case we propose a self-learning framework that combines both search and embedding methods. We empirically show that the system is able to learn objects representations that are as useful as if the system could have had access to the hidden representations. For the γ-CKL model, due to its scale-invariance, we study schemes with an exponential convergence rate to the target. We introduce a search scheme with backtracking, allowing asking global queries when necessary via backtracking mechanism, that provably converges to the target at an exponential rate. Then we propose a practical implementation of this scheme and validate its exponential convergence empirically through a number of experiments. Finally, we propose a scheme with γ-CKL model that can be used for moderate number of objects in a database.

- Finally, we show that our models and algorithms fit well in a live application with real users. First, we demonstrate practical significance of the proposed LEARN2SEARCH framework, involving methods developed for the probit model, in a scenario where no objects features are known to the search system: we observe that the search cost successfully decreases as the number of search episodes. This is achieved by iteratively refining the low-dimensional representation of the objects. The framework is scalable in the number of objects n, tolerates noisy answers, and performs well on real-world experiments of searching for movie actors. Next, for moderate n, our user study shows that under human oracles, a search based on γ-CKL can outperform GAUSSSEARCH. This suggests that a scale-free oracle could adequately models human behaviour thus enabling search algorithms with extremely favourable scaling. Our results show the potential of the scale-free oracle models, which should lead to future work in this direction.

Finding the next query to ask to the user during the search is one of the most important parts of a search algorithm. Exploiting the form of the proposed Probit model, GAUSSSEARCH finds the optimal query and updates posterior with much lower computational complexity that the state-of-the-art algorithms. Future work should consider more sophisticated optimality criterias other than the immediate expected information gain. Particularly, we wonder if defining the comparison-based search problem as a reinforcement learning problem could lead to alternative solutions that potentially might improve over the active learning paradigm. The main challenge would be in the choice of the reward function, which is not immediately obvious.

We demonstrated that the scale-free models enable a new class of efficient search schemes. First, we believe a further improvement over the exponential scheme is possible, particularly in terms of the query complexity of the algorithm. Second, future work should explore other variants of scale-free models that would bring us closer to the understanding of how humans make choices.

Our proposed LEARN2SEARCH framework is the first principle attempt to address the problem of searching with noisy comparisons when the objects features are unavailable. By learning its internal objects embedding, we approximate the "mental" objects embedding humans implicitly use when making comparisons. Future research should focus on combining recent advances in Computer Vision field, typically involving using black-box methods such as Convolutional Neural Networks, with the model-based embedding methods proposed in this work. This could result into higher quality embeddings, which will help to run more effective searches.

This framework together with the search and embedding methods developed in this thesis can be already directly used in a real world practical application. We have demonstrated that by building a web service that helps to find movie actors by comparing pictures of their faces. Another application could be a search of criminals by comparing faces of people from police databases that could be used on a national level.

Finally, we believe that the algorithms presented in here will be also particularly suitable for searching for procedurally generated content. In such a setting, the space of the objects is continuous, which is perfectly handled by all our search methods. One possible application could be a creation of a perfect avatar in online games.

Bibliography

[1] Sameer Agarwal et al. “Generalized non-metric multidimensional scaling”. In: *Artificial Intelligence and Statistics*. PMLR. 2007, pp. 11–18.

[2] Ehsan Amid and Antti Ukkonen. “Multiview triplet embedding: Learning attributes in multiple maps”. In: *International Conference on Machine Learning*. 2015, pp. 1472–1480.

[3] Jesse Anderton and Javed Aslam. “Scaling Up Ordinal Embedding: A Landmark Approach”. In: *International Conference on Machine Learning*. 2019, pp. 282–290.

[4] F Gregory Ashby and Nancy A Perrin. “Toward a unified theory of similarity and recognition.” In: *Psychological review* 95.1 (1988), p. 124.

[5] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[6] Julius R Blum et al. “Approximation methods which converge with probability one”. In: *The Annals of Mathematical Statistics* 25.2 (1954), pp. 382–386.

[7] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[8] Ralph Allan Bradley and Milton E. Terry. “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons”. In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[9] Eric Brochu, Nando de Freitas, and Abhijeet Ghosh. “Active Preference Learning with Discrete Choice Data”. In: *Advances in neural information processing systems*. 2008, pp. 409–416.

[10] Gregory Canal et al. “Active Embedding Search via Noisy Paired Comparisons”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019.

[11] Le Chang and Doris Y Tsao. “The code for facial identity in the primate brain”. In: *Cell* 169.6 (2017), pp. 1013–1028.

[12] Nick Chater and Gordon DA Brown. “Scale-invariance as a unifying psychological principle”. In: *Cognition* 69.3 (1999), B17–B24.

[13] Wei Chu and Zoubin Ghahramani. “Extensions of Gaussian Processes for Ranking: Semi-supervised and Active Learning”. In: *Proceedings of the NIPS 2005 Workshop on Learning to Rank*. Whistler, BC, Canada, Dec. 2005.

[14] Daniyar Chumbalov, Lucas Maystre, and Matthias Grossglauser. “Scalable and efficient comparison-based search without features”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1995–2005.

[15] Paulo Cortez et al. “Modeling wine preferences by data mining from physicochemical properties”. In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[16] Ingemar J Cox et al. “The Bayesian image retrieval system, PicHunter: theory, implementation, and psychophysical experiments”. In: *IEEE transactions on image processing* 9.1 (2000), pp. 20–37.

[17] Sanjoy Dasgupta. “Analysis of a greedy active learning strategy”. In: *Advances in neural information processing systems*. 2005, pp. 337–344.

[18] Daniel PW Ellis et al. “The Quest for Ground Truth in Musical Artist Similarity.” In: *ISMIR*. Paris, France. 2002.

[19] Yuchun Fang and Donald Geman. “Experiments in mental face retrieval”. In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. Springer. 2005, pp. 637–646.

[20] Marin Ferecatu and Donald Geman. “A statistical framework for image category search from a mental picture”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.6 (2009), pp. 1087–1101.

[21] Nikhil Ghosh, Yuxin Chen, and Yisong Yue. “Landmark Ordinal Embedding”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11506–11515.

[22] Daniel Golovin, Andreas Krause, and Debajyoti Ray. “Near-optimal bayesian active learning with noisy observations”. In: *Advances in Neural Information Processing Systems*. 2010, pp. 766–774.

[23] Siavash Haghiri, Debarghya Ghoshdastidar, and Ulrike von Luxburg. “Comparison-Based Nearest Neighbor Search”. In: *Artificial Intelligence and Statistics*. 2017, pp. 851–859.

[24] Eric Heim et al. “Efficient online relative comparison kernel learning”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 271–279.

[25] Neil Houlsby et al. “Bayesian active learning for classification and preference learning”. In: *arXiv preprint arXiv:1112.5745* (2011).

[26] Lalit Jain, Kevin G Jamieson, and Rob Nowak. “Finite sample prediction and recovery bounds for ordinal embedding”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 2711–2719.

[27] Kevin G Jamieson and Robert D Nowak. “Low-dimensional embedding using adaptively selected ordinal data”. In: *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*. IEEE. 2011, pp. 1077–1084.

[28] Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. “Comparison-Based Learning with Rank Nets”. In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*. EPFL-CONF-181754. 2012.

[29] Ehsan Kazemi et al. “Comparison Based Learning from Weak Oracles”. In: *International Conference on Artificial Intelligence and Statistics*. 2018, pp. 1849–1858.

[30] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *Proceedings of ICLR 2014*. Banff, AB, Canada, Apr. 2014.

[31] Matthäus Kleindessner and Ulrike von Luxburg. “Kernel functions based on triplet comparisons”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6807–6817.

[32] Joseph B Kruskal. “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”. In: *Psychometrika* 29.1 (1964), pp. 1–27.

[33] Donald Laming. *Sensory analysis*. Academic Press, 1986.

[34] R Luce, Robert R Bush, and Eugene Ed Galanter. “Handbook of mathematical psychology: I.” In: (1963).

[35] David JC MacKay. “Information-based objective functions for active data selection”. In: *Neural computation* 4.4 (1992), pp. 590–604.

[36] Thomas Peter Minka. “A family of algorithms for approximate Bayesian inference”. PhD thesis. Massachusetts Institute of Technology, 2001.

[37] Robert M Nosofsky. “Attention, similarity, and the identification–categorization relationship.” In: *Journal of experimental psychology: General* 115.1 (1986), p. 39.

[38] Robert Nowak. “Generalized binary search”. In: *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. IEEE. 2008, pp. 568–574.

[39] Robert Nowak. “Noisy generalized binary search”. In: *Advances in neural information processing systems*. 2009, pp. 1366–1374.

[40] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: (June 2014).

[41] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* 22.3 (1951), pp. 400–407.

[42] Roger N Shepard. “Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space”. In: *Psychometrika* 22.4 (1957), pp. 325–345.

[43] Roger N Shepard. “The analysis of proximities: multidimensional scaling with an unknown distance function. I.” In: *Psychometrika* 27.2 (1962), pp. 125–140.

[44] Roger N Shepard. “Toward a universal law of generalization for psychological science”. In: *Science* 237.4820 (1987), pp. 1317–1323.

[45] Neil Stewart, Gordon DA Brown, and Nick Chater. “Absolute identification by relative judgment.” In: *Psychological review* 112.4 (2005), p. 881.

[46] Nicolae Suditu and François Fleuret. “Iterative relevance feedback with adaptive exploration/exploitation trade-off”. In: *Proceedings of the 21st ACM international conference on Information and knowledge management.* 2012, pp. 1323–1331.

[47] Omer Tamuz et al. “Adaptively learning the crowd kernel”. In: *arXiv preprint arXiv:1105.1033* (2011).

[48] Louis L. Thurstone. “A Law of Comparative Judgment”. In: *Psychological Review* 34.4 (1927), pp. 273–286.

[49] Warren S Torgerson. “Multidimensional scaling: I. Theory and method”. In: *Psychometrika* 17.4 (1952), pp. 401–419.

[50] Dominique Tschopp et al. “Randomized algorithms for comparison-based search”. In: *Advances in Neural Information Processing Systems.* 2011, pp. 2231–2239.

[51] Amos Tversky. “Features of similarity.” In: *Psychological review* 84.4 (1977), p. 327.

[52] Amos Tversky and Itamar Gati. “Similarity, separability, and the triangle inequality.” In: *Psychological review* 89.2 (1982), p. 123.

[53] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[54] Laurens Van Der Maaten and Kilian Weinberger. “Stochastic triplet embedding”. In: *2012 IEEE International Workshop on Machine Learning for Signal Processing.* IEEE. 2012, pp. 1–6.

[55] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. “Cost-effective hits for relative similarity comparisons”. In: *Second AAAI conference on human computation and crowdsourcing.* 2014.

[56] Fang Zhou, Q Claire, and Ross D King. “Predicting the geographical origin of music”. In: *2014 IEEE international conference on data mining (ICDM).* IEEE. 2014, pp. 1115–1120.

Daniyar Chumbalov

✉ daniyar.chumbalov@epfl.ch — ☎ +41 78 834 11 66

Education

École Polytechnique Fédérale de Lausanne (EPFL) Lausanne, Switzerland

PhD in Computer Science 2016 - 2022

- My research focuses on statistical, stochastic and reinforcement learning techniques for comparison-based search and triplet embedding.
- Computer Science Department PhD Fellowship 2016.

MSc in Applied Mathematics 2014 - 2016

- My master thesis was on coding theory. I have obtained some new theoretical results that improved the decoding algorithm for the subspace codes over finite fields.

Moscow Institute of Physics and Technology (MIPT) Moscow, Russia

BSc in Mathematics and Computer Science 2010 - 2014

- Russian State scholarship for research achievements 2013, 2014.
- Phystech Foundation scholarship for academic achievements 2011.

Work Experience

Quantitative Researcher - Qube Research and Technologies, Paris Sept. 2022 - April 2023

Research Intern - Facebook AI, London Sept. 2019 - Dec. 2019

- Developed a brand new machine learning method for automated fact verification.
- Implemented efficient distributed training of reinforcement learning and NLP models on a GPU cluster that significantly decreased GPUs usage.
- Mentored 150 people during Facebook Conversational AI Hackathon.

Quant Intern - Edge Laboratories, Lausanne July 2015 - Jan. 2016

- Created highly optimized multi-threaded C++ library of various mathematical algorithms and tools with applications in Value at Risk computation and Portfolio Optimization.

Professional Skills

- **Proficiency**: Machine Learning, Statistics, Algorithms, Data Structures, Information Theory.
- **Technical tools**: Python (PyTorch, TensorFlow, Keras, Pandas, XGBoost), C++, AWS, Unix.
- **Languages**: Russian (native), English (fluent), French (intermediate).

Selected Publications

- Fast Interactive Search with a Scale-Free Comparison Oracle,
  L. Klein, D. Chumbalov, L. Maystre, M. Grossglauser, *Submitted to* **NeurIPS 2023**.
- Scalable and Efficient Comparison-based Search without Features,
  D. Chumbalov, L. Maystre, M. Grossglauser, **ICML 2020**.
- On the Combinatorial Version of the Slepian–Wolf Problem,
  D. Chumbalov, A. Romashchenko, **IEEE Transactions on Information Theory 2018**.

Competitions

2020	Top 10% at the G-Research NBA Data Challenge on predicting NBA games scores.
2018	4^{th} place at the WWW 2018 Challenge on Musical Genre Recognition.
2015	3^{d} prize winner at the EPFL Combinatorial Problem Solving Contest.
2010	Top 30% in the finals of the Russian National Olympiad in Informatics.