
Scalable and Efficient Comparison-based Search without Features

Daniyar Chumbalov¹ Lucas Maystre² Matthias Grossglauser¹

Abstract

We consider the problem of finding a target object t using pairwise comparisons, by asking an oracle questions of the form “Which object from the pair (i, j) is more similar to t ?”. Objects live in a space of latent features, from which the oracle generates noisy answers. First, we consider the *non-blind* setting where these features are accessible. We propose a new Bayesian comparison-based search algorithm with noisy answers; it has low computational complexity yet is efficient in the number of queries. We provide theoretical guarantees, deriving the form of the optimal query and proving almost sure convergence to the target t . Second, we consider the *blind* setting, where the object features are hidden from the search algorithm. In this setting, we combine our search method and a new distributional triplet embedding algorithm into one scalable learning framework called LEARN2SEARCH. We show that the query complexity of our approach on two real-world datasets is on par with the non-blind setting, which is not achievable using any of the current state-of-the-art embedding methods. Finally, we demonstrate the efficacy of our framework by conducting an experiment with users searching for movie actors.

1. Introduction

Finding a target object among a large collection of n objects is the central problem addressed in information retrieval (IR). For example, in web search, one finds relevant web pages through a query expressed as one or several keywords. Of course, the form of the query depends on the object type; other forms of search queries in the literature include finding images similar to a query image (Datta et al., 2008), and finding subgraphs of a large network isomorphic to a query graph (Sun et al., 2012). A common feature of this classic

¹School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland. ²Spotify. Correspondence to: Daniyar Chumbalov <daniyar.chumbalov@epfl.ch>.

formulation of the search problem is the need to express a meaningful query. However, this is often a non-trivial task for a human user. For example, most people would struggle to draw the face of a friend accurately enough that it could be used as a query image. But they would be able to confirm with near-total certainty whether a photograph is of their friend—we have no real doppelgängers in the world. In other words, comparing is cognitively much easier than formulating a query. This is true for many complex or unstructured types of data, such as music, abstract concepts, images, videos, flavors, etc.

In this work, we develop models and algorithms for searching a large database by comparing a sequence of sample objects relative to the target, thereby obviating the need for any explicit query. The central problem concerns the choice of the sequence of samples (e.g., pairs of faces that the user compares with a target face she remembers). These query objects have to be chosen in such a way that we learn as much as possible about the target, i.e., shrink the set of potential targets as quickly as possible. This is closely related to a classic problem in active learning (Settles, 2012; MacKay, 1992), assuming we have a meaningful set of features available for each object in the database.

It is natural to assume that the universe of objects lives in some low-dimensional latent feature space, even though the raw objects might be high-dimensional (images, videos, sequences of musical notes, etc.). 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 (Chang & Tsao, 2017).

The key component of our framework is a noisy oracle model that describes how a user responds to comparison queries relative to a target, based on the objects’ embedding in the latent feature space. The noise model implicitly determines which queries are likely to extract useful information from the user; it is similar to pairwise comparison models such as those of Thurstone (1927) or Bradley & Terry (1952), but here the comparison is always relative to a target. The compared samples and the target are all represented as points in the feature space. Our contributions in this context are threefold. First, we develop an efficient search algorithm, which (a) generates query pairs and (b) navigates towards the target. This algorithm is provably guaranteed

to converge to the target, due to the specific form of our comparison 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 also consider the more difficult, but realistic, scenario where the objects still live in a low-dimensional feature space, but the search algorithm does not have access to their embedding. In this *blind* setting, the latent features of an object manifest themselves only through the comparison queries. Although this assumption clearly makes the problem challenging, it also has the advantage of yielding a completely generic approach. We develop a method to generate a latent embedding of the objects from past searches, in order to make future searches more efficient. The goal of the method is to minimize the number of queries to locate a target; this is achieved by gradually improving the latent embedding of all the objects based on past searches. Unlike with previous triplet embedding approaches that use low-dimensional point estimates to represent objects, we embed each object as a Gaussian distribution, which captures the uncertainty in the exact position of the object given the data. We show that our approach leads to significant gains in a real-world application, where users search for movie actors by comparing faces: Our framework learns useful low-dimensional representations based on comparison outcomes that lead to decreasing search cost as more data is collected.

The rest of the paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we define the problem of comparison-based search and define our user (oracle) model. In Section 4, we describe our interactive search method in the non-blind setting, establish its favorable scalability, and prove its convergence properties. In Section 5, we focus on the blind setting, and introduce our distributional embedding method GAUSSEMBED and its integration into the combined LEARN2SEARCH framework. We demonstrate the performance of our algorithms on synthetic and real world datasets in Section 6 in both non-blind and blind settings, and also present the results of an experiment involving human oracles searching for movie. Finally, we give our concluding remarks in Section 7. For conciseness, we defer the full proofs of our theorems to the supplementary material.

2. Related Work

Comparison-Based Search. For the general *active learning* problem, where the goal is to identify a target hypothesis by making queries, the Generalized Binary Search method (Dasgupta, 2005; Nowak, 2008) is known to be near-optimal in the number of queries if the observed outcomes are noiseless. In the noisy case, Nowak (2009) and Golovin et al. (2010) propose objectives that are greedily minimized

over all possible queries and outcomes in a Bayesian framework with a full posterior distribution over the hypothesis. We adapt the method suggested by Golovin et al. (2010) to our setting and investigate its performance in comparison to the one proposed in this paper later in Section 6.

Adaptive search through relevance feedback by using pre-processed features was studied in the context of image retrieval systems by Cox et al. (2000); Fang & Geman (2005); Ferecatu & Geman (2009); Suditu & Fleuret (2012) in a Bayesian framework and by using different user answer models.

When the features are hidden, finding a target in a set of objects by making similarity queries is explored by Tschopp et al. (2011) and Karbasi et al. (2012) in the noiseless case. To deal with erroneous answers, Kazemi et al. (2018) consider an augmented user model that allows a third outcome to the query, “?”; it can be used 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. (2012) also briefly discussed the case of a noisy oracle with a constant mistake probability and proposed to treat it with repeated queries. Haghiri et al. (2017) consider the problem of nearest-neighbor search using comparisons in the noiseless setting.

Brochu et al. (2008) study the comparison-based search problem in continuous space. Their approach relies on maximizing a surrogate GP model of the valuation function. In contrast to their work, our approach is restricted to a specific valuation function (based on the distance to the target), but our search algorithm is backed by theoretical guarantees, its running time is independent of the number of queries and scales better with the number of objects. Chu & Ghahramani (2005) and Houlshby et al. (2011) study a similar problem where the entire valuation function needs to be estimated. Our approach is inspired from this line of work and also uses the expected information gain to drive the search algorithm. However, our exact setup enables finding a near-optimal query pair simply by evaluating a closed-form expression, as opposed to searching exhaustively over all pairs of objects. Comparison-based search under noise is also explored in Canal et al. (2019); the authors independently propose an oracle model similar to ours.

Triplet Embedding. The problem of learning objects embedding based on triplet comparisons has been studied extensively in recent years. Jamieson & Nowak (2011) provides a lower bound on the number of triplets needed to uniquely determine an embedding that satisfies all $O(n^3)$ relational constraints and describe an adaptive scheme that builds an embedding by sequentially selecting triplets of objects to compare.

More general algorithms for constructing ordinal embed-

dings from a *given* set of similarity triplets, under different triplet probability models, are proposed by Agarwal et al. (2007), Tamuz et al. (2011) and Van Der Maaten & Weinberger (2012). Amid & Ukkonen (2015) modify the optimization problem of Van Der Maaten & Weinberger (2012) to learn at once multiple low-dimensional embeddings, each corresponding to a single latent feature of an object. Theoretical properties for a general maximum likelihood estimator with an assumed probabilistic generative data model are studied by Jain et al. (2016).

An alternative approach to learning ordinal data embedding is suggested in Kleindessner & von Luxburg (2017), where the authors explicitly construct kernel functions to measure similarities between pairs of objects based on the set of triplet comparisons. Finally, Heim et al. (2015) adapt the kernel version of Van Der Maaten & Weinberger (2012) 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 and to exploit sparsity properties of the gradient. Although this work is closely related to our scenario, the kernel decomposition, which is $O(n^3)$ in time, would be too computationally expensive to perform in the regimes of interest to us.

Finally, Ghosh et al. (2019) and Anderton & Aslam (2019) both 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.

3. Preliminaries

Let us consider n objects denoted by integers $[n] = \{1, 2, \dots, n\}$. Our goal is to build an interactive comparison-based search engine that would assist users in finding their desired *target* object $t \in [n]$ by sequentially asking questions of the form “among the given pair of objects (i, j) , which is closer to your target?” and observing their answers $y \in \{i, j\}$. Formally, at each step of this search, we collect an answer y to the query (i, j) . We then use this information to decide on the next pair of objects (i', j') to show to the user, until one of the elements in the query is recognized as the desired target.

We assume that the objects have associated latent feature vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$, that reflect their individual properties and that are intuitively used by users when answering the queries. In fact, each user response y can be viewed as an outcome of a perceptual judgement on the similarity between objects that is based on these representations. A natural choice of quantifying similarities between objects is the Euclidean distance between hidden vectors in \mathbb{R}^d :

$$y(i, j | t) = i \iff \|\mathbf{x}_i - \mathbf{x}_t\| < \|\mathbf{x}_j - \mathbf{x}_t\|.$$

Different users might have slightly different perceptions of

similarity. Even for a single user, it might be difficult to answer every comparison confidently, and as such we might observe inconsistencies across their replies. We postulate that this happens when two objects \mathbf{x}_i and \mathbf{x}_j are roughly equidistant from the target \mathbf{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 \mathbf{x}_i and \mathbf{x}_j , and less noisy when it is far away from it.

We consider the following probabilistic model that captures the possible uncertainty in users’ answers (*probit model*):

$$\begin{aligned} p(y(i, j | t) = i) &= p(\mathbf{x}_t^T \mathbf{w}_{ij} + b_{ij} + \varepsilon > 0) \\ &= \Phi\left(\frac{\mathbf{x}_t^T \mathbf{w}_{ij} + b_{ij}}{\sigma_\varepsilon}\right), \end{aligned} \quad (1)$$

where $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_t \in \mathbb{R}^d$ are the coordinates of the query points and the target, respectively,

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

is the bisecting normal hyperplane to the segment between \mathbf{x}_i and \mathbf{x}_j , $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is additive Gaussian noise, and Φ is the Standard Normal CDF. Indeed, if \mathbf{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 \mathbf{x}_t to the hyperplane h_{ij} .

This model is reminiscent of pairwise comparison models such as those of Thurstone (1927) or Bradley & Terry (1952), e.g., where $p(y(i, j | t) = i) = \Phi[s(\mathbf{x}_i) - s(\mathbf{x}_j)]$ and $s(\mathbf{x}) = -\|\mathbf{x} - \mathbf{x}_t\|_2^2$ (Agarwal et al., 2007; Jain et al., 2016). These models have the undesirable property of favoring distant query points: given any $\mathbf{x}_i, \mathbf{x}_j \neq \mathbf{0}$, it is easy to verify that the pair $(2\mathbf{x}_i, 2\mathbf{x}_j)$ is strictly more discriminative for any target that does not lie on the bisecting hyperplane. Our comparison model is different: in (1), the outcome probability depends on \mathbf{x}_i and \mathbf{x}_j *only* through their bisecting hyperplane.

4. Search with Known Features

In this section, we describe our algorithm for interactive search in a set of objects $[n]$ when we have access to the features \mathcal{X} . We are interested in methods that are (a) *efficient* in the average number of queries per search, and (b) *scalable* in the number of objects. Scalability requires *low computational complexity* of the procedure for choosing the next query. As we expect users to make mistakes in their answers, we also require our algorithms to be *robust* to noise in 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 $\mathcal{N}(\hat{\mathbf{x}}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ that reflects our current belief about the position of the target point \mathbf{x}_t in \mathbb{R}^d . We model user answers with the probit likelihood (1), 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 (a) the size of the model does not depend on n , guaranteeing scalability, (b) one can characterize a general pair of points in \mathbb{R}^d that maximizes the expected information gain, and (c) the sampling scheme that chooses the next pair of query points informed by (b) is simple and works extremely well in practice.

4.1. Choosing the Next Query

To generate the next query, we follow a classical approach from information-theoretic active learning (MacKay, 1992): find the query that minimizes the expected posterior uncertainty of our estimate $\hat{\mathbf{x}}$, as given by its entropy. More specifically, we find a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ that maximizes the expected *information gain* for our current estimation of \mathbf{x}_t at step m :

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

where $\hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the current belief about the target position and $y \sim p(y \mid \hat{\mathbf{x}}, \mathbf{x}_i, \mathbf{x}_j)$ is the current belief over the answer to the comparison between \mathbf{x}_i and \mathbf{x}_j .

Exhaustively evaluating the expected information gain over all $O(n^2)$ pairs for each query (e.g., as done in Chu & Ghahramani, 2005) is prohibitively expensive. Instead, we propose a more efficient approach. Recall from equation (1) that the comparison outcome probability $p(y \mid \hat{\mathbf{x}}, \mathbf{x}_i, \mathbf{x}_j)$ depends on $(\mathbf{x}_i, \mathbf{x}_j)$ 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 Houlshby et al. (2011), it can be rewritten as:

$$\begin{aligned} I[\hat{\mathbf{x}}; y \mid h] &= H(\hat{\mathbf{x}}) - \mathbb{E}_{y \mid \hat{\mathbf{x}}, h} [H(\hat{\mathbf{x}} \mid y)] \\ &= H(y \mid \hat{\mathbf{x}}, h) - \mathbb{E}_{\hat{\mathbf{x}}} [H(y \mid \hat{\mathbf{x}}, h)]. \end{aligned} \quad (2)$$

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

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

$$\arg \max_{h \in \mathcal{H}} I[\hat{\mathbf{x}}; y \mid h] = \arg \max_{(\mathbf{w}, b) \in \mathcal{H}} \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}.$$

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

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

where $z \sim \mathcal{N}(0, \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w})$. 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. \square

In other words, Theorem 1 states that any optimal hyperplane is orthogonal to a direction that maximizes the variance of $\hat{\mathbf{x}}$.

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) to form the next query. We propose a sampling strategy based on Theorem 1, which we describe in Algorithm 1. After computing the optimal hyperplane h^* , we sample a point \mathbf{z}_1 from the current Gaussian belief over the location of the target, and reflect it across h^* to give the second point \mathbf{z}_2 . Then our query is the closest pair of (distinct) unused points $(\mathbf{x}_i, \mathbf{x}_j)$ from the dataset to $(\mathbf{z}_1, \mathbf{z}_2)$.

Assuming that the feature vectors \mathcal{X} are organized into a k -d tree (Bentley, 1975) at a one-time computational cost of $O(n \log n)$ for a given \mathcal{X} , Algorithm 1 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 1 to $O(\log n + d^2)$.

Algorithm 1 SAMPLEMIRROR

Input: current belief $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, \mathcal{X} , \mathcal{U}

Output: query pair $(\mathbf{x}_i, \mathbf{x}_j)$

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

Note that the actual hyperplane defined by the obtained pair $(\mathbf{x}_i, \mathbf{x}_j)$ in Algorithm 1 does not, in general, coincide with the optimal one. Nevertheless, the hyperplane h_{ij} approximates h^* increasingly better as n grows.

4.2. 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{x} . Suppose that we are at the m -th step of the search, and denote the belief before observing \bar{y} by $p_{m-1}(\hat{x}) = \mathcal{N}(\hat{x}; \boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1})$. Ideally, we would like to update it using

$$p_m^*(\hat{x}) \propto p(y = \bar{y} \mid \mathbf{x}_i, \mathbf{x}_j, \hat{x}) \cdot p_{m-1}(\hat{x}).$$

However, this distribution is no longer Gaussian. Therefore, we approximate it by the ‘‘closest’’ Gaussian distribution $p_m(\hat{x}) \doteq \mathcal{N}(\hat{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. We use a method known as *assumed density filtering* (Minka, 2001), which solves the following program:

$$\min_{\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m} D_{\text{KL}}[p_m^*(\hat{x}) \parallel p_m(\hat{x})],$$

where $D_{\text{KL}}[p \parallel q]$ is the Kullback-Leibler divergence from p to q . This can be done in closed form by computing the first two moments of the distribution p_m^* , with running time $O(d^2)$. Formally, at each step of the search, we perform the following computations (referred to further as the UPDATE routine):

$$\begin{aligned} \boldsymbol{\Sigma}_m &= (\boldsymbol{\Sigma}_{m-1}^{-1} + \tau \mathbf{w} \mathbf{w}^T)^{-1}, \\ \boldsymbol{\mu}_m &= \boldsymbol{\Sigma}_m [\boldsymbol{\Sigma}_{m-1}^{-1} \boldsymbol{\mu}_{m-1} + (\nu - b\tau) \mathbf{w}], \\ \tau &= \frac{-\beta}{1 + \beta \mathbf{w}^T \boldsymbol{\Sigma}_{m-1} \mathbf{w}}, \quad \nu = \frac{\alpha - \beta (\boldsymbol{\mu}_{m-1}^T \mathbf{w} + b)}{1 + \beta \mathbf{w}^T \boldsymbol{\Sigma}_{m-1} \mathbf{w}}, \end{aligned}$$

where α and β are the first and second derivatives with respect to $\mu = \boldsymbol{\mu}_{m-1}^T \mathbf{w}_{ij}$ of the log-marginal likelihood

$$\log \int_{\mathbb{R}^d} p(\bar{y}) p(\hat{x}) d\hat{x} = \log \Phi \left(\frac{\boldsymbol{\mu}_{m-1}^T \mathbf{w} + b}{\sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_{m-1} \mathbf{w} + \sigma_\varepsilon^2}} \right).$$

We summarize the full active search procedure in Algorithm 2.

Algorithm 2 GAUSSSEARCH

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

Output: target object t

- 1: Initialize $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_1$
 - 2: $\mathcal{U} \leftarrow \emptyset$
 - 3: $m \leftarrow 1$
 - 4: **repeat**
 - 5: $\mathbf{x}_i, \mathbf{x}_j \leftarrow \text{SAMPLEMIRROR}(\boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1}, \mathcal{X}, \mathcal{U})$
 - 6: $\mathcal{U} \leftarrow \mathcal{U} \cup \{i, j\}$
 - 7: $\bar{y} \leftarrow$ noisy comparison outcome as per (1)
 - 8: $\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m \leftarrow \text{UPDATE}(\mathbf{x}_i, \mathbf{x}_j, \bar{y}, \boldsymbol{\mu}_{m-1}, \boldsymbol{\Sigma}_{m-1})$
 - 9: $m \leftarrow m + 1$
 - 10: **until** $t \in \{i, j\}$
-

4.3. Convergence of GAUSSSEARCH

For finite n , GAUSSSEARCH is guaranteed to terminate because samples are used 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 2. *If the answers follow (1), then for any initial $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0 \succ 0$, as $m \rightarrow \infty$ we have*

- (i) $\text{Tr}(\boldsymbol{\Sigma}_m) \rightarrow 0$,
- (ii) $\boldsymbol{\mu}_m \rightarrow \mathbf{x}_t$

almost surely.

Proof sketch. We obtain crucial insights on the asymptotic behavior by studying the case $d = 1$, when \mathbf{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 (Robbins & Monro, 1951; Blum et al., 1954), we can show that $\mu_m \rightarrow x_t$ 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. \square

This result is not a priori obvious, especially in light of the model for the oracle: answers become coin flips as the sample points \mathbf{x}_i and \mathbf{x}_j get closer. Nevertheless, Theorem 2 guarantees that the algorithm continues making progress towards the target.

5. Search with Latent Features

In this section, we relax the assumption that the feature vectors \mathcal{X} are available; instead, we consider the blind scenario, where \mathcal{X} is only used implicitly by the oracle to generate answers to queries. As a consequence, we need to generate an estimated feature embedding $\hat{\mathcal{X}} = \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n\}$ from comparison data collected from past searches. As we collect more data, the embedding $\hat{\mathcal{X}}$ should gradually approximate the true embedding \mathcal{X} (up to rotation and reflection), improving the performance of future searches.

In a given dataset of triplet comparison outcomes, one item 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 item and less information about another. This suggests that when approximating \mathcal{X} , an *uncertainty* about the item’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{x}_i and \hat{x}_j in our estimated embedding space are noisy, the target posterior distribution should have higher entropy than if \hat{x}_i and \hat{x}_j were exact (as we had assumed so far).

We incorporate this dependence by generating a distributional embedding, where each estimate \hat{x}_i is endowed with a distribution, thus capturing the uncertainty about its true position x_i . We also slightly modify the posterior update step (line 8 in Algorithm 2) in the search algorithm, in order to reflect the embedding uncertainty in the belief about the target. We refer to this combined learning framework as LEARN2SEARCH. It combines GAUSSSEARCH, our search method from the previous section, with a new triplet embedding technique called GAUSSEMBED. We now describe our distributional embedding method, then give a brief description of the combined framework.

5.1. Distributional Embedding

As before, suppose we are given a set of objects $[n] = \{1, 2, \dots, n\}$, known to have representations $\mathcal{X} \subset \mathbb{R}^d$ in a hidden feature space. Although we do not have access to \mathcal{X} , we observe a set of noisy triplet-based relative similarities of these objects: $\mathcal{T} = \{(i, j; k) \mid \text{object } i \text{ is closer to } k \text{ than } j \text{ is}\}$, obtained w.r.t. probabilistic model (1). The goal is to learn a d -dimensional Gaussian distribution embedding

$$q(\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n) \doteq \prod_i \mathcal{N}(\hat{x}_i; \nu_i, \Psi_i)$$

where $\nu_i \in \mathbb{R}^d$, $\Psi_i \in \mathbb{R}^{d \times d}$ and diagonal, such that objects that are similar w.r.t. their “true” representations $\{x_i\}_{i=1}^n$ (reflected by \mathcal{T}) are also similar in the learned embedding. Here for each object, ν_i represents the mean guess on the location of object i in \mathbb{R}^d , and Ψ_i represents the uncertainty about its position.

We learn this Gaussian embedding via maximizing the Evidence Lower Bound (ELBO),

$$\mathcal{L} = \mathbb{E}_q [\log p(\mathcal{T}, \hat{x}_1, \dots, \hat{x}_n) - \log q(\hat{x}_1, \dots, \hat{x}_n)],$$

where the joint distribution $p(\mathcal{T}, \hat{x}_1, \dots, \hat{x}_n)$ is the product of the prior $p(\hat{x}_1, \dots, \hat{x}_n) = \prod_i \mathcal{N}(\hat{x}_i; \mathbf{0}, \mathbf{I})$ and the likelihood $p(\mathcal{T} \mid \hat{x}_1, \dots, \hat{x}_n)$ given w.r.t. the probabilistic model (1) with $\sigma_\varepsilon = 1$. We optimize the ELBO by using stochastic backpropagation with the reparametrization trick (Kingma & Welling, 2014; Rezende et al., 2014). The total complexity of executing one epoch is $O(|\mathcal{T}|d)$.

5.2. Learning to Search

We now describe our joint embedding and search framework, LEARN2SEARCH. We begin by initializing $\hat{\mathcal{X}}$ to its

prior distribution, $\prod_i \mathcal{N}(\hat{x}_i; \mathbf{0}, \mathbf{I})$. Then we alternate between searching and updating the embedding, based on all the comparison triplets collected in previous searches, by (a) running GAUSSSEARCH on the estimated embedding $\hat{\mathcal{X}}$, and (b) running GAUSSEMBED to refine $\hat{\mathcal{X}}$ based on all triplets observed so far. The distributional embedding requires the following modifications in GAUSSSEARCH.

1. in SAMPLEMIRROR, we use the Mahalanobis distance $(\nu - z)^T \Psi^{-1} (\nu - z)$ when finding the closest objects i, j to the sampled points z_1, 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{x}_i and \hat{x}_j along the direction perpendicular to the hyperplane, resulting in an effective noise variance $\sigma_{\text{eff}}^2 = \sigma_\varepsilon^2 + \mathbf{w}^T \Psi_i \mathbf{w} + \mathbf{w}^T \Psi_j \mathbf{w}$.

Both modifications results in a natural tradeoff between exploration and exploitation: At first, we tend to sample uncertain points 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.

6. Experiments

In this section, we evaluate our approach empirically and compare it to competing methods. In Section 6.1 we focus on the query complexity and computational performance of comparison-based search algorithms under the assumption that the feature vectors \mathcal{X} are known. We then turn to the blind setting. In Section 6.2 we evaluate our joint search & embedding framework on two real world datasets and compare it against state-of-the-art embedding methods. In Section 6.3 we describe the results of applying our framework in an experiment on searching for movie actors involving real users.

6.1. Non-Blind Setting

The main difference between GAUSSSEARCH and previous methods is the way the uncertainty about the target object is modeled. As outlined in Section 2, all previous approaches consider a discrete posterior distribution $P = \{p_1, p_2, \dots, p_n\}$ 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 EC² active learning algorithm (Golovin et al., 2010) 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

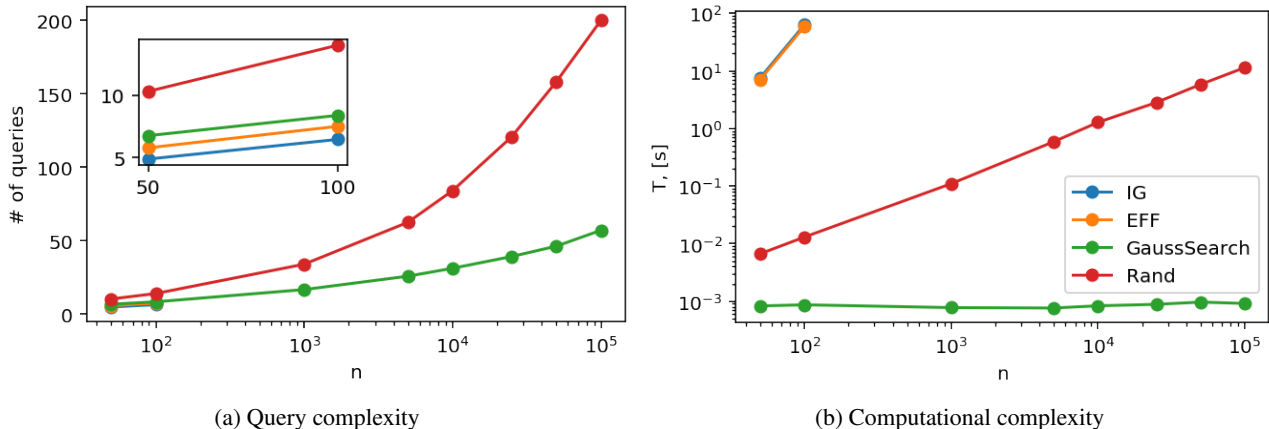


Figure 1. Average (a) query complexity and (b) computational complexity of four search algorithms in the non-blind setting for $d = 5$ and increasing n . Search success is declared when $\arg \max_{i \in [n]} P(\mathbf{x}_i) = t$.

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 (1), 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 $P(\mathbf{x}_i)$ to be proportional to the density of the Gaussian posterior at \mathbf{x}_i . We use two performance metrics.

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

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

Fig 1a and Fig 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. 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).

6.2. Blind Setting

We now turn our attention to the blind setting and study the performance of LEARN2SEARCH on two real world datasets: the *red wine* dataset (Cortez et al., 2009) and the *music* dataset (Zhou et al., 2014). These datasets were studied in the context of comparison-based search by Kazemi et al. (2018). We assume that the feature vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ are latent: they are used to generate comparison outcomes, but they are not available to the search algorithm. We compare different approaches to learning an estimate of the feature vectors from comparison outcomes. In addition to our distributional embedding method GAUSSEMBED, we consider two state-of-the-art embedding algorithms: t -STE (Van Der Maaten & Weinberger, 2012) and CKL (Tamuz et al., 2011) (both of which produce point estimates of the feature vectors), as well as the ground-truth vectors \mathcal{X} .

We run 9000 searches in total, starting from a randomly initialized $\hat{\mathcal{X}}$ and updating the embedding on the k -th search for $k \in [2^0, 2^1, \dots, 2^{13}]$. For each search, the target object

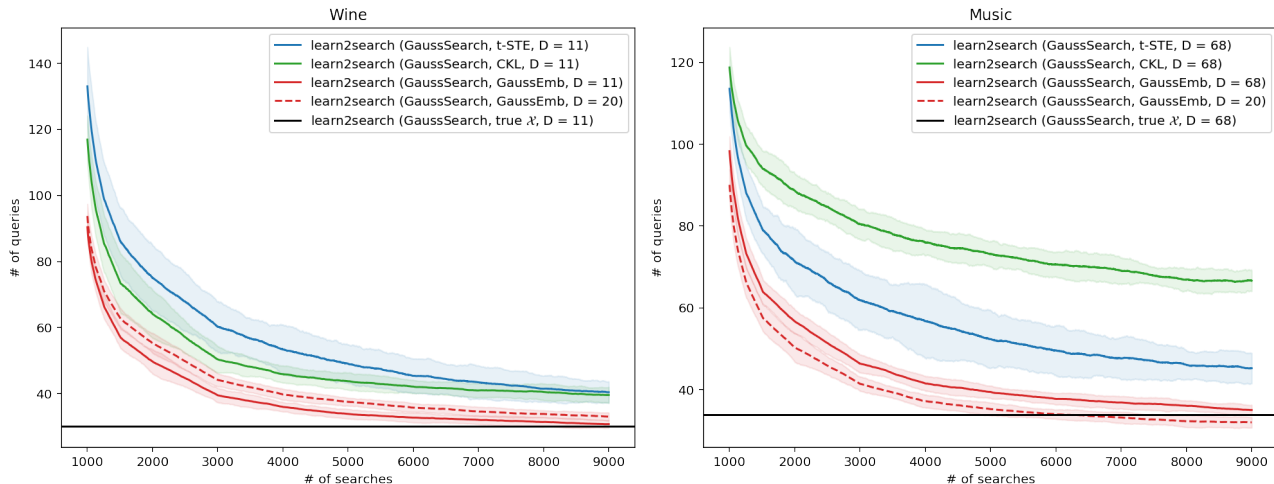


Figure 2. Combined search and embedding framework LEARN2SEARCH on two datasets with different embedding techniques and choices of D . Results are reported over a sliding window of size 1000.

is chosen uniformly at random among all objects. The outcomes of the comparisons queried by GAUSSSEARCH are sampled from the probit model (1) 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.

We present the results, averaged over 100 experiments, in Fig. 2. The combination of 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 blind 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 blind setting is the choice of 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 $\bar{\mathbf{X}}_\mu$. 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. 2 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 \mathbf{X} : on average, when the search is run on $\hat{\mathbf{X}}$, GAUSSSEARCH needs fewer queries than when it is performed on \mathbf{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.

6.3. Movie Actors Face Search Experiment

We present the results of an experiment involving human oracles, in order to validate the practical relevance of the model and search algorithm developed in this paper. We collected $n = 513$ photographs of the faces of well-known actors and actresses, and we built a web-based search interface implementing the LEARN2SEARCH method over this dataset.¹ In this system, a user can look for an actor by comparing faces: at each search iteration, the user is shown 4 pictures of faces, and can then click on the face that looks the most like the target; this process repeats until the target is among the 4 samples. Once it is, the user clicks on a button to see details about the target, which ends the search. We have collected slightly more than 40 000 comparison

¹Movie Actors Search: <http://who-is-th.at/>. The source code for our methods as well as the data collected in the face search experiment will be available at <https://indy.epfl.ch/>.

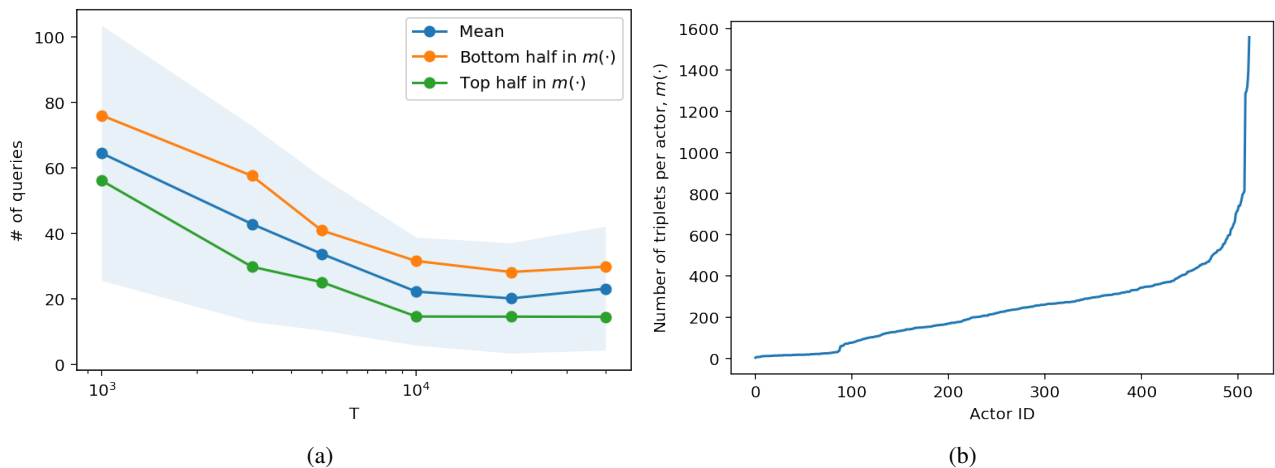


Figure 3. (a) Empirical search cost, averaged over 5 human subjects, for a database with $n = 513$ faces of movie actors. The averages in the top and bottom group of targets (in terms of number of triplets) are shown individually as well: clearly, the denser group (green) has lower search cost. The blue 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. (b) Distribution of the number of collected triplets per actor, $m(\cdot)$. The actor with the highest number of triplets is Halle Berry with $m(\text{Halle Berry}) = 1567$ and the actor with the least number of triplets is Kumail Nanjiani with $m(\text{Kumail Nanjiani}) = 16$. The median number of triplets per actor is 227.

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, 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 Section 5. For each value of T , each subject performed 10 searches. The results are shown in Fig. 3a. 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 $\sim 10k$ 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. Call $m(i)$ 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. 3b, because (a) objects were added to the system gradually over time,

and (b) the distribution over targets follows popularity, which is heavily skewed itself. Concretely, this ranges from $m(\text{Halle Berry}) = 1567$ to $m(\text{Kumail Nanjiani}) = 16$. 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. 3a, 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 blind 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.

7. Conclusion

In this work, we introduce a probabilistic model of triplet comparisons, a search algorithm for finding a target among a set of alternatives, and a Bayesian embedding algorithm for triplet data. Our search algorithm is backed by theoretical guarantees and strikes a favorable computational vs. query complexity trade-off. By combining the search and embedding algorithms, we design a system that learns to search efficiently without a-priori access to object features. We observe that the search cost successfully decreases as the number of search episodes increases. This is achieved by iteratively refining a distributional, latent, and low-dimensional representation of the objects. Our framework is scalable in the number of objects n , tolerates noisy answers, and performs well on synthetic and real-world experiments.

Acknowledgements

The authors wish to thank Olivier Cloux for the development of the <https://who-is-th.at/> platform, and for his help with running and evaluating the experiments. We are grateful to Holly Cogliati for proof-reading the manuscript.

References

- Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., and Belongie, S. Generalized non-metric multidimensional scaling. In *Artificial Intelligence and Statistics*, pp. 11–18, 2007.
- Amid, E. and Ukkonen, A. Multiview triplet embedding: Learning attributes in multiple maps. In *International Conference on Machine Learning*, pp. 1472–1480, 2015.
- Anderton, J. and Aslam, J. Scaling up ordinal embedding: A landmark approach. In *International Conference on Machine Learning*, pp. 282–290, 2019.
- Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
- Blum, J. R. et al. Approximation methods which converge with probability one. *The Annals of Mathematical Statistics*, 25(2):382–386, 1954.
- Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- Brochu, E., de Freitas, N., and Ghosh, A. Active preference learning with discrete choice data. In *Advances in neural information processing systems*, pp. 409–416, 2008.
- Canal, G., Massimino, A., Davenport, M., and Rozell, C. Active embedding search via noisy paired comparisons. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.
- Chang, L. and Tsao, D. Y. The code for facial identity in the primate brain. *Cell*, 169(6):1013–1028, 2017.
- Chu, W. and Ghahramani, Z. 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, December 2005.
- Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.
- Cox, I. J., Miller, M. L., Minka, T. P., Papathomas, T. V., and Yianilos, P. N. The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. *IEEE transactions on image processing*, 9(1):20–37, 2000.
- Dasgupta, S. Analysis of a greedy active learning strategy. In *Advances in neural information processing systems*, pp. 337–344, 2005.
- Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2):5, 2008.
- Fang, Y. and Geman, D. Experiments in mental face retrieval. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pp. 637–646. Springer, 2005.
- Ferecatu, M. and Geman, D. A statistical framework for image category search from a mental picture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1087–1101, 2009.
- Ghosh, N., Chen, Y., and Yue, Y. Landmark ordinal embedding. In *Advances in Neural Information Processing Systems*, pp. 11506–11515, 2019.
- Golovin, D., Krause, A., and Ray, D. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pp. 766–774, 2010.
- Haghiri, S., Ghoshdastidar, D., and von Luxburg, U. Comparison-based nearest neighbor search. In *Artificial Intelligence and Statistics*, pp. 851–859, 2017.
- Heim, E., Berger, M., Seversky, L. M., and Hauskrecht, M. Efficient online relative comparison kernel learning. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pp. 271–279. SIAM, 2015.
- Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.
- Jain, L., Jamieson, K. G., and Nowak, R. Finite sample prediction and recovery bounds for ordinal embedding. In *Advances In Neural Information Processing Systems*, pp. 2711–2719, 2016.
- Jamieson, K. G. and Nowak, R. D. Low-dimensional embedding using adaptively selected ordinal data. In *Communication, Control, and Computing (Allerton)*, 2011 49th Annual Allerton Conference on, pp. 1077–1084. IEEE, 2011.
- Karbasi, A., Ioannidis, S., and Massoulié, L. Comparison-based learning with rank nets. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, number EPFL-CONF-181754, 2012.

- Kazemi, E., Chen, L., Dasgupta, S., and Karbasi, A. Comparison based learning from weak oracles. In *International Conference on Artificial Intelligence and Statistics*, pp. 1849–1858, 2018.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *Proceedings of ICLR 2014*, Banff, AB, Canada, April 2014.
- Kleindessner, M. and von Luxburg, U. Kernel functions based on triplet comparisons. In *Advances in Neural Information Processing Systems*, pp. 6807–6817, 2017.
- MacKay, D. J. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.
- Minka, T. P. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.
- Nowak, R. Generalized binary search. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pp. 568–574. IEEE, 2008.
- Nowak, R. Noisy generalized binary search. In *Advances in neural information processing systems*, pp. 1366–1374, 2009.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. June 2014.
- Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, 22(3): 400–407, 1951.
- Settles, B. *Active Learning*. Morgan & Claypool Publishers, 2012.
- Suditu, N. and Fleuret, F. Iterative relevance feedback with adaptive exploration/exploitation trade-off. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pp. 1323–1331. ACM, 2012.
- Sun, Z., Wang, H., Wang, H., Shao, B., and Li, J. Efficient subgraph matching on billion node graphs. *Proceedings of the VLDB Endowment*, 5(9):788–799, 2012.
- Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. T. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 673–680. Omnipress, 2011.
- Thurstone, L. L. A law of comparative judgment. *Psychological Review*, 34(4):273–286, 1927.
- Tschopp, D., Diggavi, S., Delgosha, P., and Mohajer, S. Randomized algorithms for comparison-based search. In *Advances in Neural Information Processing Systems*, pp. 2231–2239, 2011.
- Van Der Maaten, L. and Weinberger, K. Stochastic triplet embedding. In *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*, pp. 1–6. IEEE, 2012.
- Zhou, F., Claire, Q., and King, R. D. Predicting the geographical origin of music. In *2014 IEEE international conference on data mining (ICDM)*, pp. 1115–1120. IEEE, 2014.