Learning to Search Efficiently Using Comparisons

Daniyar Chumbalov 1  Lucas Maystre 1  Matthias Grossglauser 1

Abstract
We consider the problem of searching in a set of items by using pairwise comparisons. We aim to locate a target item \( t \) by asking an oracle questions of the form “Which item from the pair \((i, j)\) is more similar to \( t \)?”. We assume a blind setting, where no item features are available to guide the search process; only the oracle sees the features in order to generate an answer. Previous approaches for this problem either assume noiseless answers, or they scale poorly in the number of items, both of which preclude practical applications. In this paper, we present a new scalable learning framework called LEARN2SEARCH that performs efficient comparison-based search on a set of items despite the presence of noise in the answers. Items live in a space of latent features, and we posit a probabilistic model for the oracle comparing two items \( i \) and \( j \) with respect to a target \( t \). Our algorithm maintains its own representation of the space of items, which it learns incrementally based on past searches. We evaluate the performance of LEARN2SEARCH on both synthetic and real-world data, and show that it learns to search more and more efficiently, over time matching the performance of a scheme with access to the item features.

1. Introduction
Searching for 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 in the form of a set of keywords. Of course, the form of the query depends very much on the type of object being searched; examples in the literature include finding images similar to a query image, or subgraphs of a large network isomorphic to a query graph.

A common feature of the classical formulation of the search problem is the need to express a meaningful query. However, for some important types of information, this is no trivial task for a human querier. For example, most people would struggle to draw a face they have seen; at the same time, they could confirm with near-total certainty whether two images of faces belong to the same person—we have no real doppelgängers in the world. In other words, comparing is cognitively much easier than querying. This is true for many “complex” types of data, such as pieces of music, abstract concepts, images, and the like.

In this work, we attempt to develop models and algorithms to query a large database by answering a sequence of comparison queries, thereby obviating the need for any explicit query. The central problem concerns the choice of the sequence of query objects (e.g., pairs of images 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 plausible set of potential targets. This is closely related to a classical problem in active learning, 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, can be quite accurately described by a few tens of features, capturing head shape, eye color, fullness of lips, gender, age, and the like.

We assume in this paper that this latent space is difficult to extract from the raw data, e.g., through face recognition techniques.

In this blind setting, the latent features of an object only manifest themselves through the comparison queries. This setting essentially amounts to a reinforcement learning problem, where we have to rely on past search results in order to generate a latent embedding of the objects, 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 a latent embedding of all the objects based on past searches. While assuming that no explicit object features are available clearly makes the problem challenging, it has the advantage of yielding a completely generic approach: the only hyperparameter to know
We combine the search and the embedding problem into a single probabilistic framework. The central component of this framework is a noisy oracle model, describing how a user responds to comparison queries relative to a target. The noise model is central, as it 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 and Terry (1952), but distinctly different. 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 computationally inexpensive but provably greedy, thanks to the specific form of our comparison model. Second, we develop an embedding algorithm that computes a latent embedding of all the objects based on past searches. Third, we show experimentally that blind comparison-based search is tractable: starting from a random embedding, we can drastically reduce the number of queries needed to identify a target after a few hundred searches. Our approach is, to the best of our knowledge, the first complete and scalable approach to blind comparison-based search.

The rest of the paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we describe the formal problem of the comparison based search, define our user model, and present the general formulation of our reinforcement learning algorithm LEARN2SEARCH. Then in Section 4, we propose our new scalable method for finding the target object through pairwise comparisons, if the features of the objects are known. In Section 5, we address the problem of learning object representations based on their relative triplets comparisons. We demonstrate the performance of each piece of our scheme separately and in combination on real world datasets in Section 6. Finally, we give our concluding remarks in Section 7. The proof of our main theorem can be found in the Appendix.

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 will investigate the performance of the method suggested by Golovin et al. (2010) in comparison to the one proposed in this paper later in Section 6.

Finding a target object in a set of objects with hidden features 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, “?” which can be used by a user to indicate that he does not know the answer confidently. With this model they propose several new search techniques under different assumptions on the knowledge of relative objects distances.

When the objects features are known, adaptive search through relevance feedback was studied in the context of image retrieval systems by Cox et al. (2000); Fang and Geman (2005); Ferecatu and Geman (2009); Suditu and Fleurier (2012) under a Bayesian framework and different user answer models.

Brochu et al. (2008) study the comparison-based search problem in continuous spaces. 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 simpler, backed by theoretical guarantees, and its running time is independent of the number queries. [Chu and Ghahramani (2005) and Houlsby 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 the optimal query pair simply by evaluating a closed-form expression.

Triplet Embedding The problem of learning object embedding based on triplet comparisons has been studied by many researchers in recent years. Jamieson and Nowak (2011) provide 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 embeddings 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 and Weinberger (2012). Amid and Ukkonen (2015) modify the optimization problem of Van Der Maaten and 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 and 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 and Weinberger.
We consider the following probabilistic model that captures the possible uncertainty in user’s answers:

\[
p(y(i, j \mid t) = i) = p(y = i \mid x_i, x_j, x_t) \\
= p(y = i \mid x_i, h_{i,j}) \\
= p(x_i^T w_{i,j} + b_{i,j} + \varepsilon > 0) \\
= \Phi \left( \frac{x_i^T w_{i,j} + b_{i,j}}{\sigma_\varepsilon} \right),
\]

where \(x_i, x_j, x_t \in \mathbb{R}^d\) are the coordinates of the query points and the target respectively, \(h_{i,j} = (w_{i,j}, b_{i,j}) = (x_i - x_j, \|x_i\|^2 - \|x_j\|^2)\) is the bisecting normal hyperplane to the segment between \(x_i\) and \(x_j\), \(\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)\) is additive Gaussian noise, and \(\Phi\) is the Standard Normal CDF. Indeed, if \(x_t\) is on the hyperplane \(h_{i,j}\), the feedback becomes Gaussian noise, as the target point is equally far from both query points. When \(x_t\) is close to, but not necessarily on the hyperplane, the effect of random noise still plays a major role in the user answer decisions. Finally, when \(x_t\) starts to get farther from the decision bound, probability of the correct outcome increases.

This model is reminiscent of pairwise comparison models such as those of Thurstone (1927) or Bradley and Terry (1952), e.g., where \(p(y(i, j \mid t) = i) = \Phi(s(x_i) - s(x_j))\) and \(s(x) = -\|x - x_0\|^2\) (Agarwal et al., 2007; Jain et al., 2016). These models have the undesirable property of favoring distant query points: given any \(x_i, x_j \neq 0\), it is easy to verify that the pair \((2x_i, 2x_j)\) is strictly more discriminative for any target that does not lie on the bisecting hyperplane.

We want to stress, however, that our comparison model is different: in equation (1) the outcome probability depends on \(x_i\) and \(x_j\) only through the bisecting hyperplane that they generate.

### 3. Preliminaries

Let us consider \(n\) objects denoted by integers \([n] = \{1, 2, \ldots, n\}\). 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 \(m\) of this search, we collect an answer \(y_m\) to the query \(i) m, j_m)\). Then we show the user the next pair of objects \((i_{m+1}, j_{m+1})\), until one of the elements in the query is recognized as the desired target. We assume that except for the comparison data collected during past searches, we do not have access to any other type of representational information about the objects (such as images, text descriptions, etc.).

Nevertheless, we assume that the objects have associated latent feature vectors \(X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times d}\) that reflect their individual properties and that are intuitively utilized 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 internal representations \(X\). A natural choice of quantifying similarities between objects would be the Euclidean distance between hidden vectors in \(\mathbb{R}^d\):

\[
y(i, j \mid t) = i \Rightarrow \|x_i - x_t\| < \|x_j - x_t\|.
\]

However, since it might not be necessarily easy for a person to always describe his answers quantitatively, it is possible that some of the feedback from a user is inconsistent with his previous replies or replies of other users. The inaccuracy in answers could occur when two objects \(x_i\) and \(x_j\) are roughly equally distant from the target \(x_t\), and it is hard for a user to state confidently which object from the given query is closer. Intuitively, this difficulty arises when the target vector \(x_t\) is close to the “decision boundary”, i.e., the bisecting normal hyperplane to the segment between \(x_i\) and \(x_j\). For example, when \(x_j\) is perfectly equally distant from both objects in the query, i.e., \(x_t\) lies on this hyperplane, both answers \(i\) and \(j\) could be expected to be equally probable to be chosen by a user. Moreover, even if \(x_j\) is far from the decision boundary, the possibility of a mistake in the answer still can not be completely eliminated (however one would expect this probability to diminish the farther \(x_t\) moves from the decision boundary, as the user becomes more certain about his answer).

We consider the following probabilistic model that captures
choosing which next query to make. Finally, since 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 interactive nature of the problem, we take a Bayesian approach to model the uncertainty about the location of the target. In particular, at each step \( m \) of the search, we use a \( d \)-dimensional Gaussian distribution \( \mathcal{N}(\hat{x}^m, \Sigma_m) \) to reflect our current belief about the position of the target point \( x_t \) in \( \mathbb{R}^d \). We model user answers with probit likelihood, and apply approximate inference techniques for updating \( \mu \) and \( \Sigma \) every time we observe a query outcome. The total memory complexity of this model is \( O(d^2) \).

The motivation behind such a choice of parametrization is that a) parameters do not depend on \( n \), which leads to great scalability in practice, b) one can characterize a general pair of points in \( \mathbb{R}^d \) that is provably optimal for optimization the expected information gain, and c) the next query sampling scheme inspired from the theoretical results is simple and works extremely well in practice.

### 4.1. Choosing The Next Query

To decide on the next query, it is important first to understand what value of information each query potentially carries. We follow a classic approach from information-theoretic active learning [MacKay 1992], and find the query that minimizes the expected posterior uncertainty of our estimate \( \hat{x}_t \) given by its Shannon’s entropy. More specifically, we seek a pair of points \((x_i, x_j)\) that maximizes the expected information gain (IG) for our current estimation of \( x_t \) at step \( m \):

\[
U((x_i, x_j), \hat{x}_t^m) = H(\hat{x}_t^m) - \mathbb{E}_y[H(\hat{x}_t^m | y, x_i, x_j)],
\]

where \( \hat{x}_t^m \sim N(\hat{x}_t^m; \mu_m, \Sigma_m) \) is the current prior for the target position on the \( m \)-th step of the search, \( y \sim p(y | x_i, x_j, \hat{x}_t^m) \) is the answer distribution and \( \hat{x}_t^m | y, x_i, x_j \) is the posterior distribution after observing outcome \( y \) for the query \((x_i, x_j)\).

Evaluating the expected information gain exhaustively over all \( O(n^2) \) pairs for each query (as done, e.g., in [Chu and Ghahramani 2005]) is prohibitively expensive. Instead, we propose a simple, yet very effective heuristic that runs in time \( O(n \log n) \) with \( O(n \log n) \) preprocessing using a kd-tree, that is done only once for an embedding.

First, let us note that for our user model \( p(y | x_i, x_j, x_t) \) the query outcome distribution depends on \((x_i, x_j)\) only through the corresponding bisecting normal hyperplane \( h_{i,j} \). Thus instead of looking for the optimal pair of points from \( X \), one could look for a hyperplane \( h \) that maximizes the expected IG:

\[
U(h, \hat{x}_t^m) = H(\hat{x}_t^m) - \mathbb{E}_y[H(\hat{x}_t^m | y, h)].
\]

This utility function is equivalent to the mutual information \( I[\hat{x}_t^m; y | h] \), as discussed in [Houlsby et al. 2011]. It can be rewritten as

\[
U(h, \hat{x}_t^m) = H(y | \hat{x}_t^m, h) - \mathbb{E}_x[H(y | x, h)],
\]

where \( x \sim p(\hat{x}_t^m) \). The following theorem gives us the key insight about the general form of a hyperplane \( h = (w, b) \) that optimizes this utility function by splitting the Gaussian distribution into two equal “halves”:

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

\[
\arg\max_{h \in H} U(h, x) = \arg\max_{(w, b) \in H} w^T\Sigma w.
\]

In other words, any optimal hyperplane is orthogonal to a direction that maximizes the variance of \( x \).

Thus one could find a hyperplane that maximizes the expected IG by simply performing eigenvalue decomposition on \( \Sigma_m \). This can be done in time \( O(d^3) \), which does not depend on \( n \). However, we still need to find a pair of objects \((i, j)\) to construct the next query. We propose the following sampling heuristic that exploits the results of Theorem 1:

**Algorithm 1** **SAMPLEMIRROR**

1. Compute optimal hyperplane \( \hat{h} \) for \((\mu_m, \Sigma_m)\).
2. Sample a point \( z_1 \) from \( \mathcal{N}(\hat{x}_t; \mu_m, \Sigma_m) \).
3. Obtain \( z_2 \), mirror reflection of \( z_1 \) across \( h \).
4. Find objects \( i \) and \( j \), that were not shown yet, with the closest representations \( x_i \) and \( x_j \) to \( z_1 \) and \( z_2 \), respectively.

The total computational complexity of these four steps is \( O(d^3 + d^2 + d + \log n) = O(\log n) \).

The actual hyperplane defined by the obtained pair \((x_i, x_j)\) does not in general coincide with the optimal one, nevertheless, for big enough \( n \) one could make a reasonable assumption of high density of points \( \{x_i\}_{i=1}^n \) in \( \mathbb{R}^d \), and therefore expect \( h_{i,j} \) to be a good enough approximation for \( \hat{h} \). This was empirically confirmed in our experiments (see Figure 1a).

### 4.2. Model Update

Finally, after querying the pair \((i_m, j_m)\) and observing answer \( y_m \), we update our belief on the location of target using Bayes’ formula. Ideally, we would like to compute

\[
p(\hat{x}_t^{m+1} | y = y_m, x_{i_m}, x_{j_m}, \hat{x}_t^m) \propto p(y = y_m | x_{i_m}, x_{j_m}, \hat{x}_t^m) p(\hat{x}_t^m),
\]

However this distribution is no longer Gaussian. Therefore we seek to approximate it by the “closest” Gaussian distribution \( q(\hat{x}_t^{m+1}) = \mathcal{N}(\hat{x}_t^{m+1}; \mu, \Sigma) \). Formally, we use a method known as assumed density filtering [Minka 2001].
Algorithm 2 summarizes our active search procedure:

**Algorithm 2** **GAUSSMIRROR**

Require: Set of coordinates \( \mathcal{X} = \{x_i\}_{i=1}^{n} \subset \mathbb{R}^d \).

1. Set prior \( x_{i}^0 \sim N(\mu_0, \Sigma_0) \).
2. \( m \leftarrow 0 \)
3. repeat
   4. \( h_m \leftarrow \arg \max_{h} U(h, \mu_m, \Sigma_m) \)
   5. \( x_{i,m}, x_{j,m} \leftarrow \text{SAMPLEMIRROR}(h_m, \mu_m, \Sigma_m, \mathcal{X}) \)
   6. \( y_m \leftarrow \text{ANSWER}(x_{i,m}, x_{j,m}) \)
   7. \( x_{t}^{m+1} \leftarrow \text{UPDATE}(x_{i,m}, x_{j,m}, y_m, \mu_m, \Sigma_m) \)
   8. \( m \leftarrow m + 1 \)
   9. until target is found

5. **Robust Embedding**

As the second part of our scheme, we address the problem of object embedding based on their triplet comparisons data. Suppose we are given a set of objects \([n] = \{1, 2, \ldots, n\}\), known to have representations \( \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d \) in a hidden feature space. Although we do not have access to \( \mathcal{X} \), we observe a set of triplet-based relative similarities of these objects:

\[ \mathcal{T} = \{(i, j, k) \mid \text{object } i \text{ is closer to } k \text{ than } j \} \]

The goal is to learn a \( d \)-dimensional embedding \( \hat{\mathcal{X}} \subset \mathbb{R}^{n \times d} \) of \([n] \) from the data \( \mathcal{T} \) such that the Euclidean distances between obtained representations \( \{\hat{x}_i\}_{i=1}^{n} \) reflect the corresponding triplet distance relations of their “true” representations \( \{x_i\}_{i=1}^{n} \) well. In other words, we require

\[ ||x_i - x_k|| < ||x_j - x_k|| \iff ||\hat{x}_i - \hat{x}_k|| < ||\hat{x}_j - \hat{x}_k|| \]

We are interested in the situation when the triplets are generated under a probabilistic noise assumption. Particularly, following the probit noise model \([1]\), we assume that any correct triplet \((i, j; k)\), i.e., for which \( ||x_i - x_k|| < ||x_j - x_k|| \) holds, can be flipped with probability \( p(y = j \mid x_i, x_j, x_k) \). In this case an incorrect (w.r.t. the ground-truth) triplet \((j, i; k)\) is added to \( \mathcal{T} \). We then estimate the embedding \( \hat{\mathcal{X}} \) by minimizing the log-loss with an additional \( \ell_2 \) regularization term:

\[ \min_{\hat{\mathcal{X}}} \sum_{(i, j; k) \in \mathcal{T}} \log p_{i, j, k} + \lambda \text{trace}(\hat{\mathcal{X}}^T \hat{\mathcal{X}}), \]

where the triplet probability \( p_{i, j, k} \) is defined as in \([1]\). Here the entries of \( \hat{\mathcal{X}} \) are treated as parameters to be optimized, and the probability \( p_{i, j, k} \) reflects how well a triplet \((i, j; k)\) is satisfied in the estimated embedding. The added regularization term prevents points from growing arbitrarily large in norm and enables a better generalization to unseen triplets.

5.1. **Dimensionality Estimation**

Another important problem of object embedding is to choose the dimensionality \( d \) of the estimated representations. In the original triplet embedding task the dimensionality \( d \) of the hidden feature space \( \mathcal{X} \) is assumed to be known and is explicitly used in \( \mathcal{X} \). However, if \( d \) is not known \textit{a priori}, overshooting the number of dimensions \( D > d \) leads to potential overfitting, while choosing a smaller \( D < d \) simplifies the model too much, and does not allow it to capture the training signal well. The added regularization term in \([1]\) brings some light on the problem of choosing \( D \) in the absence of knowledge of the actual \( d \). We will demonstrate how by varying the regularization constant \( \lambda \), one could estimate the “true” dimensionality \( d \) by looking at the spectrum of covariance matrix \( \frac{1}{n} \mathcal{X}^T \mathcal{X} \) during the learning process starting from a initial, large value for the dimensionality parameter \( D \).

6. **Experiments**

6.1. **Search: Efficiency In Entropy Reduction**

In order to see how well our sampling technique for picking the next query approximates the result from Theorem \([1]\), we compared **GAUSSMIRROR** with two baselines **GAUSSDENSE** and **GAUSSUNIFORM**. **GAUSSDENSE** computes the optimal hyperplane from Theorem \([1]\) for the current Gaussian posterior, which is then used in the answer model directly. This case captures the limiting behavior, where objects are dense in the space, and the optimal hyperplane, that brings the maximum expected information gain, can always be queried exactly, e.g. when the set of points is the whole \([0, 1]^d\). The second baseline **GAUSSUNIFORM** chooses two distinct points for the query from \([n]\) uniformly at random.

We ran 100 queries length searches on two datasets with different number of objects: \( n = 1000 \) and \( n = 10000 \). Both datasets are generated by uniformly sampling points from \([0,1]^5\) and the probit user model is used for answering the queries with \( \sigma = 0.1 \). After each query we measured the entropy of the updated Gaussian posterior \( H(\mu, \Sigma) = \frac{1}{2} \ln \det(2\pi\sigma \Sigma) \). The results are averaged over 1000 experiments and are presented in Fig. \([1a]\).

We observe that, as the number of points \( n \) increases, our strategy **GAUSSMIRROR** converges towards the perfect entropy reduction rate of **GAUSSDENSE**. Note that both values of \( n \), **GAUSSDENSE** gives identical results, as it does not choose a pair of points, but instead always picks the hyperplane that cuts the Gaussian posterior precisely the optimal way. The uniform **GAUSSUNIFORM** random stra-
Figure 1. (a) Entropy reduction rate during the search under Gaussian posterior model for different next query picking techniques. Average (b) query and (c) computational complexity of four search algorithms on 100 size subsets of each dataset with \( P(x_i) \geq 0.5 \) stopping criteria and assuming accessibility of the objects features.

6.2. Search: Query And Computational Complexities

The main difference between our proposed search techniques and previously known methods is the way the uncertainty about the target object is modeled. As outlined in Section 2, most of the previous approaches consider a discrete posterior distribution over the objects in \([n]\), and use full Bayesian update after each search step. We compared GAUSSMIRROR to the following baselines operating on the discrete distribution \( P = \{p_1, p_2, \ldots, p_n\} \): (1) IG: the next query \((i, j)\) is chosen to maximize the expected information gain, (2) EFF: the fast approximation of EC\(^2\) (Golovin et al., 2010) and (3) RAND: the query pair of points is chosen uniformly from \([n]\).

The experiments were conducted on three datasets: 5-dimensional hypercube, 11-dimensional red wine (Cortez et al., 2009) and 68-dimensional music (Zhou et al., 2014), and the actual features of the objects \(X\) were accessible during the searches. For each dataset we sample 100 random points and initialize each search with a randomly chosen target. During the search, the probit model generates the answers to queries with \( \sigma^2 \) chosen s.t. around 10% of the queries’ outcomes are flipped, and each algorithm is run until its stopping criteria are met. As expected, in terms of query complexity the full posterior algorithms IG and EFF show better results with about 9 and 13 queries per search on average across all datasets, however demonstrating significantly worse running time performance with more than 1 minute on average needed to choose a query and perform posterior update. GAUSSMIRROR comes with excellent query/computational complexity tradeoff—on hypercube, red wine, and music it requires only 6, 9, and 11 queries more than IG to determine the target whereas achieving computational speedup of 78, 40, and 19 times on the respective datasets, compared to IG. Note that the complexity of the Gaussian posterior update depends only on the dimensionality \(d\) of the data, which is \(d = 68\) for the music dataset, and thus for the low-dimensional data embeddings one should expect further acceleration. Finally, RAND has the worst performance among other strategies in the average number of queries, and has constant running time for the query choice and posterior update on all datasets (as \(n\) is the same). Indeed, both IG and EFF require \(O(n^3)\) operations to choose the next query, as they perform greedy search over all pair of points. Moreover, as all three baselines store full points distribution, they all have \(O(n)\) posterior update time complexity and \(O(n)\) space complexity. This is in contrast...
to GaussMirror that finds query in $O(\log n)$ time, does posterior update in $O(d^2)$ and has $O(d^3)$ space complexity.

We close this section by exploring the search cost as a function of $n$. Figure 2 shows the average number of queries until the target is found, as a function of $n$, for different $\sigma^2 \varepsilon$. Now the search stops as soon as one of the objects in the query is indicated as the target. Our model implies that as the queries get closer to the target, the error probability of the oracle increases, tending to a perfect coin flip in the limit. This might suggest that we can decompose the search into roughly two phases: in the first phase, when the noise variance $\sigma^2 \varepsilon$ is still negligible with respect to $\Sigma$, the search progresses rapidly, because the oracle reveals almost one bit per query; once $\Sigma$ is small with respect to $\sigma^2 \varepsilon$, queries become very noisy, behaving almost like coin flips. In the latter phase, the number of queries to find the target would be approximately $n'/4$, where $n'$ is the number of objects within a radius of $\propto \sigma \varepsilon$ of the target. We would therefore expect the search cost to grow linearly in $n$ beyond a critical size. The simulations in Figure 2 suggest that this approximation is too crude, and that even beyond the “noise horizon” $\sigma \varepsilon$, while progress towards the target is slower, it remains markedly faster than by unbiased coin flips. While we have not found a way to characterize this benign scaling of the search cost in $n$ analytically, it is encouraging that the remaining information in queries close to the target are sufficient to ensure a clearly sublinear evoluation of this cost.

![Figure 2](image)

**Figure 2.** The search cost (average number of queries until the target is found) as a function of $n$, for GaussMirror with three different values of $\sigma \varepsilon$, and for Random, which simply samples two different objects (without replacement) at each step. The search cost for GaussMirror scales much more benignly that the purely random strategy, even for large values of $\sigma \varepsilon$.

### 6.3. Embedding: Performance

To make sure our user model reflects reality reasonably well, we evaluated the quality of the object embedding learned by our embedding technique Prob (optimizing (3)) on two real world datasets with crowdsourced triplet comparisons: Music artists (Ellis et al., 2002) and Food (Wilber et al., 2014).

We compared Prob to two recent embedding algorithms that consider other user models: Crowd Kernel Learning (CKL, (Tamuz et al., 2011)) and t-Stochastic Triplet Embedding (t-STE, (Van Der Maaten and Weinberger, 2012)). We measured generalization error—the percentage of satisfied triplets in the learned embeddings on a holdout set using 10-fold cross validation. Since the “true” dimensionality of the feature space is not known a priori, we also vary the dimensionality $D$ of the estimated embedding between 2 and 30.

On both datasets all three algorithms showed very similar performance between each other. Already with $D = 10$ all Prob, CKL, and t-STE managed to correctly model between 83% and 85% of triplets in the holdout set, and increasing the dimensionality had no significant impact on the performance of any of them. From that we can conclude that the noise model considered in this paper reflects the real user behaviour on the comparison-like tasks well.

### 6.4. Combined Search And Embedding

Finally, we combine the two components of our framework. We assume that the latent design matrix $X$ is unknown, and we maintain our own object representations $\hat{X} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n] \in \mathbb{R}^{n \times d}$. This embedding is initialized at random, and is periodically updated after collecting comparison data from users through searches. Our algorithm consists of two steps that are executed in a loop: a) an interactive search procedure that runs effectively on approximated $\hat{X}$, and b) embedding $\hat{X}$ that approximates the hidden attributes of objects in a way that preserves the relative objects similarities, obtained during previous searches. We sum up the general scheme of our search engine mechanism in Algorithm 3.

**Algorithm 3 LEARN2SEARCH**

1: Initialize $\hat{X} \in \mathbb{R}^{n \times d}$
2: Initialize $T \leftarrow \emptyset$
3: while true do
4: Run $k$ interactive searches using embedding $\hat{X}$
5: Collect triplets $T_k$ from these searches
6: $T = T \cup T_k$
7: Update $\hat{X}$ on data $T$
8: end while

We considered two settings: when the true dimensionality $d$ of the objects’ feature space is known and when it is not.

The experiments were performed on the full red wine and music datasets. We used GaussMirror as the core search mechanism that ran on the estimated $\hat{X}$ and Prob to learn object representations $\hat{X}$. As baselines we used GaussMirror combined with t-STE, CKL, and the true $X$. 

---

To summarize, our approach to learning an efficient search engine model using comparison data suggests a promising direction for future research in the area of user modeling for comparison-based search.
We ran 9000 searches in total, starting from a randomly initialized $\hat{X}$, and updating the embedding on the $j$-th search for $j \in [1, 2, 4, \ldots, 1024, 2048, \ldots, 8192]$. Each search was initialized with a randomly chosen target object from the dataset. The answers for the queries, produced by GAUSSMIRROR, were given w.r.t. the probit model on the true $X$ corresponding to each dataset. A search was considered finished when the target appeared as one of the objects in the query. As before, the noise level $\sigma^2$ was set to corrupt approximately 10% of the answers on average.

Since we perform searches and learn the embedding simultaneously, we measured the number of queries needed for GAUSSMIRROR to find the target using the current version of $\hat{X}$ for each search, and then averaged 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. 3. As observed from the plots, when $d$ is known, the combination of GAUSSMIRROR and PROB managed to learn object representations that give rise searches as efficient as in the ground-truth $X$, and significantly outperformed the other two baselines with t-STE and CKL embeddings. As the number of searches grows, PROB learns a better embedding $\hat{X}$ that allows GAUSSMIRROR to ask fewer queries to find the target. In the final $\hat{X}$, our search method is as efficient as in the true embedding on Wine dataset and is slightly better than with the true embedding on Music.

Estimating $d$: even if $d$ is not known, one can find an approximation to be used in the embedding algorithm by conservatively setting $D$ to a large value, and then examining the spectrum of the covariance matrix $\hat{X}^T \hat{X}$ once enough triplet samples have been collected. We tried setting an initial $D = 100$, and after collecting around 100000 triplets, we computed the eigen-decomposition of the covariance matrix of the current embedding estimate $\hat{X}$ and the number of its principal components containing 98% of the energy. 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. 3 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 $X$: on average, when the search is run on $\hat{X}$, GAUSSMIRROR needs fewer queries than when it is performed on $X$. 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.

7. Conclusion

We presented a probablistic model and a combined search and embedding algorithm to search a target among a set of alternatives using comparisons. The performance of future searches improves with the number of past searches. This is achieved by learning a latent low-dimensional representation of the objects from the past search results, which in turn improves the information gain of queries asked in future searches. The framework is scalable in the number of objects $n$, and is robust with respect to the dimension $d$ of the latent space. It tolerates noise in the answers, and performs well on synthetic and real-world datasets.
BIBLIOGRAPHY

Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., and Belongie, S. (2007). Generalized non-metric multidimensional scaling. In Artificial Intelligence and Statistics, pages 11–18.

Amid, E. and Ukkonen, A. (2015). Multiview triplet embedding: Learning attributes in multiple maps. In International Conference on Machine Learning, pages 1472–1480.

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39(3/4):324–345.

Brochu, E., de Freitas, N., and Ghosh, A. (2008). Active preference learning with discrete choice data. In Advances in neural information processing systems, pages 409–416.

Chu, W. and Ghahramani, Z. (2005). 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.

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

Cox, I. J., Miller, M. L., Minka, T. P., Papathomas, T. V., and Yianilos, P. N. (2000). The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. IEEE transactions on image processing, 9(1):20–37.

Dasgupta, S. (2005). Analysis of a greedy active learning strategy. In Advances in neural information processing systems, pages 337–344.

Ellis, D. P., Whitman, B., Berenzweig, A., and Lawrence, S. (2002). The quest for ground truth in musical artist similarity. In ISMIR. Paris, France.

Fang, Y. and Geman, D. (2005). Experiments in mental face retrieval. In International Conference on Audio-and Video-Based Biometric Person Authentication, pages 637–646. Springer.

Ferecatu, M. and Geman, D. (2009). A statistical framework for image category search from a mental picture. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(6):1087–1101.

Golovin, D., Krause, A., and Ray, D. (2010). Near-optimal bayesian active learning with noisy observations. In Advances in Neural Information Processing Systems, pages 766–774.

Heim, E., Berger, M., Seversky, L. M., and Hauskrecht, M. (2015). Efficient online relative comparison kernel learning. In Proceedings of the 2015 SIAM International Conference on Data Mining, pages 271–279. SIAM.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745.

Jain, L., Jamieson, K. G., and Nowak, R. (2016). Finite sample prediction and recovery bounds for ordinal embedding. In Advances In Neural Information Processing Systems, pages 2711–2719.

Jamieson, K. G. and Nowak, R. D. (2011). Low-dimensional embedding using adaptively selected ordinal data. In Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on, pages 1077–1084. IEEE.

Karbasi, A. et al. (2012). Comparison-based learning with rank nets. In Proceedings of the 29th International Conference on Machine Learning (ICML), number EPFL-CONF-181754.

Kazemi, E., Chen, L., Dasgupta, S., and Karbasi, A. (2018). Comparison based learning from weak oracles. In International Conference on Artificial Intelligence and Statistics, pages 1849–1858.

Klein, E., and van Luxburg, U. (2017). Kernel functions based on triplet comparisons. In Advances in Neural Information Processing Systems, pages 6807–6817.

MacKay, D. J. (1992). Information-based objective functions for active data selection. Neural computation, 4(4):590–604.

Minka, T. P. (2001). A family of algorithms for approximate Bayesian inference. PhD thesis, Massachusetts Institute of Technology.

Nowak, R. (2008). Generalized binary search. In Communication, Control, and Computing, 2008 46th Annual Allerton Conference on, pages 568–574. IEEE.

Nowak, R. (2009). Noisy generalized binary search. In Advances in neural information processing systems, pages 1366–1374.

Suditu, N. and Fleuret, F. (2012). Iterative relevance feedback with adaptive exploration/exploitation trade-off. In Proceedings of the 21st ACM international conference on Information and knowledge management, pages 1323–1331. ACM.

Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. T. (2011). Adaptively learning the crowd kernel. In Proceedings of the 28th International Conference on International Conference on Machine Learning, pages 673–680. Omnipress.
A. Proof of Theorem

Proof. Without loss of generality, assume that \( \sigma_x = 1 \). Let \( y \) be a binary random variable such that \( p(y = 1 | w, b, x) = \Phi(x^T w + b) \). Then,

\[
\arg \max_{(w, b) \in H} U((w, b), x) = \arg \max_{(w, b) \in H} \{1 - \mathbb{E}_x[H(y | w, b, x)]\} = \arg \max_{(w, b) \in H} \int_{\mathbb{R}^d} H[\Phi(x^T w + b)] \mathcal{N}(x; \mu, \Sigma) dx
\]

\[
= \arg \min_{(w, b) \in H} \int_{\mathbb{R}^d} H[\Phi(x^T w + b)] \mathcal{N}(x; \mu, \Sigma) dx = \arg \min_{(w, b) \in H} \int_{\mathbb{R}} H[\Phi(t)] \mathcal{N}(t; 0, w^T \Sigma w) dt = \arg \max_{(w, b) \in H} w^T \Sigma w.
\]

In (4), we use (2) and the fact that, as the hyperplane is passing through \( \mu \),

\[
H \left[ \mathbb{P}_{x \in \mathbb{R}^d} y = 1 | w, b, x, \mathcal{N}(x; \mu, \Sigma) dx \right] = H(1/2) = 1.
\]

In (5), we use the fact that \( x^T w + b \sim \mathcal{N}(0, w^T \Sigma w) \), by properties of the Gaussian distribution. Finally, in (6), we start by noting that, for all \( c_1, c_2 \) such that \( c_1/c_2 > 1 \),

\[
H[\Phi(c_1 t)] \leq H[\Phi(c_2 t)] \quad \text{for all } t \text{ with equality iff } t = 0.
\]

Hence, if \( \tilde{\sigma}^2 > \sigma^2 \), then

\[
\int_{\mathbb{R}} H[\Phi(t)] \mathcal{N}(t; 0, \tilde{\sigma}^2) dt < \int_{\mathbb{R}} H[\Phi(t)] \mathcal{N}(t; 0, \sigma^2) dt.
\]

Therefore, maximizing \( w^T \Sigma w \) minimizes the expected entropy of \( y \).