Comparison Based Learning from Weak Oracles

Ehsan Kazemi
Yale University

Lin Chen
Yale University

Sanjoy Dasgupta
University of California San Diego

Amin Karbasi
Yale University

Abstract

There is increasing interest in learning algorithms that involve interaction between human and machine. Comparison-based queries are among the most natural ways to get feedback from humans. A challenge in designing comparison-based interactive learning algorithms is coping with noisy answers. The most common fix is to submit a query several times, but this is not applicable in many situations due to its prohibitive cost and due to the unrealistic assumption of independent noise in different repetitions of the same query.

In this paper, we introduce a new weak oracle model, where a non-malicious user responds to a pairwise comparison query only when she is quite sure about the answer. This model is able to mimic the behavior of a human in noise-prone regions. We also consider the application of this weak oracle model to the problem of content search (a variant of the nearest neighbor search problem) through comparisons. More specifically, we aim at devising efficient algorithms to locate a target object in a database equipped with a dissimilarity metric via invocation of the weak comparison oracle. We propose two algorithms termed WORCS-I and WORCS-II (Weak-Oracle Comparison-based Search), which provably locate the target object in a number of comparisons close to the entropy of the target distribution. While WORCS-I provides better theoretical guarantees, WORCS-II is applicable to more technically challenging scenarios where the algorithm has limited access to the ranking dissimilarity between objects. A series of experiments validate the performance of our proposed algorithms.

1 Introduction

Interactive machine learning refers to many important applications of machine learning that involve collaboration of human and machines. The goal of an interactive learning algorithm is to learn an unknown target hypothesis from input provided by humans[1] in the forms of labels, pairwise comparisons, rankings or numerical evaluations [Huang et al., 2010, Settles, 2010, Yan et al., 2011]. A good algorithm, in both theory and practice, should be able to efficiently deal with inconsistent or noisy data because human feedback can be erroneous. For instance, experimental studies show error rates up to 30% in crowdsourcing platforms such as Amazon Mechanical Turk [Ipeirotis et al., 2010].

Studies conducted by psychologists and sociologists suggest that humans are bad at assigning meaningful numerical values to distances between objects or at rankings [Stewart et al., 2005]. It has been shown that pairwise comparisons are a natural way to collect user feedback [Thurstone, 1927, Salganik and Levy, 2015]: they tend to produce responses that are most consistent across users, and they are less sensitive to noise and manipulation [Maystre and Grossglauser, 2015]. Despite many successful applications of pairwise comparison queries in noise-free environments, little is known on how to design and analyze noise-tolerant algorithms. Furthermore, many active learning methods end up querying points that are the most noise-prone [Balcan et al., 2009].

In the literature, the most common approach to cope with noisy answers from comparison based oracles is to make a query several times and use majority voting [Dalvi et al., 2013]. These methods assume that by repeating a question they can reduce the error probability arbitrarily. As an example, this approach has been considered in the context of classic binary search through different hypotheses and shown to be suboptimal [Karp and Kleinberg, 2007, Nowak, 2009]. In general, the idea of repeated queries for handling noisy oracles suffers from three main disadvantages: (i) In many important applications, such as recommender systems and

---

[1]In this paper, we refer to human, crowd or user interchangeably.

exploratory searches, the algorithm is interacting with only one human and it is not possible to incorporate feedback from several users. (ii) Many queries are inherently difficult, even for human experts, to answer. (iii) Massive redundancy significantly increases the total number of oracle accesses in the learning process. Note that each query information requires significant cost and effort.

In order to efficiently address the problem of non-perfect and noisy answers from the crowd, we introduce a new *weak oracle model*, where a non-malicious user responds to a pairwise comparison query only when she is quite sure about the answer More specifically, we focus on the application of this weak oracle model to queries of the form of *"is object z more similar to object x or to object y?"*. In this model, a weak oracle gives an answer only if one of the two objects x or y is substantially more similar to z than the other. We make this important assumption to cope with difficult and error-prone situations where, to almost the same degree, the two objects are similar to the target. Since the oracle may decline to answer a query, we can refer to it as an *abstention oracle*. This model is one of the very first attempts to somewhat accurately mimic the behavior of crowd in real world problems. To motivate the main model and algorithms of this paper consider the following example.

Example 1. *Assume Alice is a new user to our movie recommender system platform. In order to recommend potentially interesting movies to Alice, we plan to figure out an unknown target movie[2] that she likes the most. Our aim is to find her favorite movie by asking her preference over pairs of movies she has watched already. With each response from Alice, we expect to get closer to the target movie. For this reason, we are interested in algorithms that find Alice's taste by asking a few questions. To achieve this goal, we should (i) present Alice movies that are different enough from each other to make decision easier for her, and (ii) find her interest with the minimum number of pairwise comparisons because each query takes some time and the number of potential queries might be limited.*

To elaborate more on the usefulness of our weak oracle model, as a special case of active learning, we aim at devising an algorithm to locate a target object in a database through comparisons. We model objects in the database as a set of points in a metric space. In this problem, a user (modeled by a weak oracle) has an object t (e.g., her favorite movie or music) in mind which is not revealed to the algorithm. At each time, the user is presented with two candidate objects x and y. Then through the query $\mathcal{O}_t(x, y)$ she indicates which of these is closer to the target object. Previous

works assume a *strong comparison oracle* that always gives the correct answer [Tschopp et al., 2011, Karbasi et al., 2012a,b]. However, as we discussed earlier, it is challenging, in practice, for the user to pinpoint the closer object when the two are similar to the target to almost the same degree.

We therefore propose the problem of comparison-based search via a *weak oracle* that gives an answer only if one of the two objects is substantially more similar to the target than the other. Note that due to the weak oracle assumption, the user cannot choose the closer object if they are in almost the same distance to the target. Our goal is to find the target by making as few queries as possible to the weak comparison oracle. We also consider the case in which the demand over the target objects is heterogeneous, i.e., there are different demands for object in the database.

Analysis of our algorithms relies on a measure of intrinsic dimensionality for the distribution of demand. These notions of dimension imply, roughly, that the number of points in a ball in the metric space increases by only a constant factor when the radius of the ball is doubled. This is known to hold, for instance, under limited curvature and random selection of points from a manifold (two reasonable assumptions in machine learning applications) [Karger and Ruhl, 2002]. Computing the intrinsic dimension is difficult in general, but we do not need to know its value; it merely appears in the query bounds for our algorithms.

Contributions In this paper, we make the following contributions: (i) We introduce the weak oracle model to handle human behavior, when answering comparison queries, in a more realistic way. We believe that this weak oracle model opens new avenues for designing algorithms, which handle noisy situations efficiently, for many machine learning applications. (ii) As an important example of our weak oracle model, we propose a new adaptive algorithm, called WORCS-I, to find an object in a database. This algorithm relies on a priori knowledge of the distance between objects, i.e., it assumes we can compute the distance between any two objects. (iii) We prove that, under the weak oracle model, if the search space satisfies certain mild conditions then our algorithm will find the target in a near-optimal number of queries. (iv) In many realistic and technically challenging scenarios, it is difficult to compute all the pairwise distances and we might have access only to a noisy information about relative distances of points. To address this challenge, we propose WORCS-II. The main assumption in WORCS-II is that for all triplets x, y and z we only know if the the distance of one of x or y is closer to z by a factor of α. Note this is a much less required information than knowing all the pairwise distances. This kind of information could be obtained, for example, from an

---

[2] The target movie of Alice is unknown to the platform.

approximation of the distance function.[3] We prove that WORCS-II, by using only this minimal information, can locate the target in a reasonable number of queries. We also prove that if WORCS-II is allowed to learn about the ranking relationships between objects, it provides better theoretical guarantees. (v) Finally, we evaluate the performance of our proposed algorithms and several other baseline algorithms over real datasets.

The remainder of this paper is organized as follows. We discuss the related work in Section 2. We overview definitions and preliminaries in Section 3. We present the main algorithms of this paper in Section 4. In Section 5, we report our experimental evaluations. We conclude the paper in Section 6. **Proofs for all of theoretical results are deferred to the Supplementary Material.**

2   Related Work

Interactive learning from pairwise comparisons has a range of applications in machine learning, including exploratory search for objects in a database [Tschopp et al., 2011, Karbasi et al., 2011, 2012a,b], nearest neighbor search [Goyal et al., 2008], ranking or finding the top-k ranked items [Chen et al., 2013, Wauthier et al., 2013, Eriksson, 2013, Maystre and Grossglauser, 2015, Heckel et al., 2016, Maystre and Grossglauser, 2017], crowdsourcing entity resolution [Wang et al., 2012, 2013, Firmani et al., 2016], learning user preferences in a recommender system [Wang et al., 2014, Qian et al., 2015], estimating an underlying value function [Balcan et al., 2016], multiclass classification [Hastie and Tibshirani, 1997], embedding visualization [Tamuz et al., 2011] and image retrieval [Wah et al., 2014].

Comparison-based search, as a form of interactive learning, was first introduced by Goyal et al. [2008]. Lifshits and Zhang [2009] and Tschopp et al. [2011] explored this problem in worst-case scenarios, where they assumed demands for the objects are homogeneous.[4] Karbasi et al. [2011] took a Bayesian approach to the problem for heterogeneous demands, i.e, where the distribution of demands are far from the uniform. It is noteworthy that the two problems of content search and finding nearest neighbor are closely related [Clarkson, 2006, Indyk and Motwani, 1998].

Search though comparisons can be formalized as an special case of the Generalized Binary Search (GBS) or splitting algorithm [Dasgupta, 2004, 2005], where the goal is to learn an underlying function from pairwise comparisons. In practice GBS performs very well in terms of required number of queries, but it appears to be difficult to provide tight bounds for its performance. Furthermore, its computational complexity is

prohibitive for large databases. For this reason, Karbasi et al. [2012a,b] introduced faster algorithms that locate the target in a database with near-optimal expected query complexities. In their work, they assume a strong oracle model and a small intrinsic dimension for points in the metric space.

The search process should be robust against erroneous answers. Karp and Kleinberg [2007] and Nowak [2009] explored the idea of repeated queries for the GBS to cope with noisy oracles. Also, for the special case of search though comparisons, Karbasi et al. [2012a] propose similar ideas for handling noise.

The notion of assuming a small intrinsic dimension for points in a metric space is used in many different problems such as finding the nearest neighbor [Clarkson, 2006, Har-Peled and Kumar, 2013], routing [Hildrum et al., 2004, Tschopp et al., 2015], and proximity search [Krauthgamer and Lee, 2004].

3   Definitions and Preliminaries

Let $\mathcal{M}$ be a metric space with distance function $d(\cdot, \cdot)$. Intuitively, $\mathcal{M}$ characterizes the database of objects. An example of a distance function is the Euclidean distance between feature vectors of items in a database. Given any object $x \in \mathcal{M}$, we define the ball centered at x with radius r as $B_x(r) \triangleq \{y \in \mathcal{M} : d(x, y) \leq r\}$. We define the diameter of a set $A \subseteq \mathcal{M}$ as $\text{diam}(A) = \sup_{x,y \in A} d(x, y)$. We will consider a distribution μ over the set $\mathcal{M}$ which we call the *demand*: it is a non-negative function $\mathcal{M} \to [0, 1]$ with $\sum_{x \in \mathcal{M}} \mu(x) = 1$. For any set $A \subseteq \mathcal{M}$, we define $\mu(A) = \sum_{x \in A} \mu(x)$. The *entropy* of μ is given by $H(\mu) = \sum_{x \in \text{supp}(\mathcal{M})} \mu(x) \log \frac{1}{\mu(x)}$, where $\text{supp}(\mu) \triangleq \{x \in \mathcal{M} : \mu(x) > 0\}$ is the support of μ. Next, we define an ϵ-cover and an ϵ-net for a set $\mathcal{A} \subseteq \mathcal{M}$.

Definition 1 (ϵ-cover)**.** *An ϵ-cover of a subset $\mathcal{A} \subseteq \mathcal{M}$ is a collection of points $\{x_1, \cdots, x_k\}$ of $\mathcal{A}$ such that $\mathcal{A} \subseteq \bigcup_{i=1}^{k} B_{x_i}(\epsilon)$.*

Definition 2 (ϵ-net)**.** *An ϵ-net of a subset $\mathcal{A} \subseteq \mathcal{M}$ is a maximal (a set to which no more objects can be added) collection of points $\{x_1, \cdots, x_k\}$ of $\mathcal{A}$ such that for any $i \neq j$ we have $d(x_i, x_j) > \epsilon$.*

An ϵ-net can be constructed efficiently via a greedy algorithm in $O(k^2|\mathcal{A}|)$ time, where k is the size of the ϵ-net. We refer interested readers to [Clarkson, 2006]. Our algorithms are analyzed under natural assumptions on the expansion rate and underlying dimension of datasets, where we define them next.

Doubling Measures   The *doubling constant* captures the embedding topology of the set of objects in a metric space when we are generalizing analysis from Euclidean spaces to more general metric spaces. Such restrictions imply that the volume of a closed

---

[3]Finding the partial relative distances between points is not the focus of this paper.

[4]Their distributions are close to uniform.

ball in the space does not increase drastically when its radius is increased by a certain factor. Formally, the doubling constant of the measure μ [Karger and Ruhl, 2002] is given by

$$c \triangleq \sup_{x \in \mathrm{supp}(\mu), R \geq 0} \frac{\mu(B_x(2R))}{\mu(B_x(R))}.$$

A measure μ is c-*doubling* if its doubling constant is $c < \infty$. We define $\dim_m(\mathcal{M}) = \log c$. To provide guarantees for the performance of algorithms in Sections 4.3 and 4.4, a stronger notion of doubling measure is required. Intuitively, we need bounds on the expansion rate of all the possible subsets of support of μ. The *strong doubling constant* of the measure μ [Ram and Gray, 2013, Haghiri et al., 2017] is given by

$$c_{\mathrm{strong}} \triangleq \sup_{A \subseteq \mathcal{M}} \sup_{x \in \mathrm{supp}(\mu) \cap A, R \geq 0} \frac{\mu(B_x(2R) \cap A)}{\mu(B_x(R) \cap A)}.$$

We also say that measure μ is c_{strong}-*doubling* if its strong doubling constant is $c_{\mathrm{strong}} < \infty$. Note that for a finite space $\mathcal{M}$ both c and c_{strong} are bounded.

Doubling Spaces The doubling dimension of a metric space $(\mathcal{M}, d)$ is the smallest k such that every subset $B_x(2R)$ of $\mathcal{M}$ can be covered by at most k balls of radius R. We define the doubling dimension $\dim_s(\mathcal{M}) = \log_2 k$. Lemma 1 states that a metric with a bounded doubling constant has a bounded doubling dimension, i.e., every doubling measure carries a doubling space.

Lemma 1 (Proposition 1.2. [Gupta et al., 2003]). *For any finite metric $(\mathcal{M}, d)$, we have $\dim_s(\mathcal{M}) \leq 4\dim_m(\mathcal{M})$, where $\dim_m(\mathcal{M}) \triangleq \log_2 c$.*

Weak Comparison Oracle The *strong comparison oracle* $\mathcal{O}_z^{\mathrm{strong}}(x, y)$ [Goyal et al., 2008] takes three objects x, y, z as input and returns which one of x and y is closer to z; formally,

$$\mathcal{O}_z^{\mathrm{strong}}(x, y) = \begin{cases} x, & \text{if } d(x, z) \leq d(y, z), \\ y, & \text{otherwise.} \end{cases}$$

In this paper, we propose the *weak comparison oracle*[5] as a more realistic and practical model. This oracle gives an answer for sure only if one of $d(x, z)$ and $d(y, z)$ is substantially smaller than the other. Formally,

$$\mathcal{O}_z(x, y) = \begin{cases} x, & \text{if } \alpha d(x, z) \leq d(y, z), \\ y, & \text{if } \alpha d(y, z) \leq d(x, z), \\ ? \text{ or } x, & \text{if } d(x, z) \leq d(y, z) < \alpha d(x, z), \\ ? \text{ or } y, & \text{if } d(y, z) < d(x, z) < \alpha d(y, z). \end{cases}$$

where $\alpha \geq 1$ is a constant that characterizes the approximate equidistance of x and y. More precisely, the

answers to a query $\mathcal{O}_z(x, y)$ has the following interpretation: (i) if it is x, then $d(x, z) \leq d(y, z)$. (ii) if it is y, $d(x, z) > d(y, z)$. (iii) if it is "?", $d(x, z)$ and $d(y, z)$ are within a multiplicative factor α of each other. In our model, an answer of x or y does not provide any information about the relative distances of x and y from the target other than stating that one of them is closer to t. Given $x, y \in A \subseteq \mathcal{M}$, the α-weighted Voronoi cell of x is defined as $Vor(x, y, A) = \{v \in A : \alpha d(x, v) \leq d(y, v)\}$, where $\alpha \geq 1$. Similarly, we can define the α-weighted Voronoi cell of y as $Vor(y, x, A)$. Note that the two Voronoi cells $Vor(x, y, A)$ and $Vor(y, x, A)$ contain the set of points which we are sure about the answers we get from the oracle.

4 Main Results

Our main goal is to design an algorithm that can locate an unknown target $t \in \mathcal{M}$ by making only $f(c)H(\mu)$ queries in expectation, where $f(c)$ is polynomial in the doubling constant c. In addition, we are interested in algorithms with a low computational complexity for deciding on the next query which we are going to make. Note that identifying a target having access to the very powerful *membership oracle*[6] in average needs at least $H(\mu)$ queries [Cover and Thomas, 2012]. Also, Karbasi et al. [2011] proved that a strong oracle needs $\Omega(cH(\mu))$ queries to find an object via a strong oracle. To sum-up, we conclude that an algorithm with a query complexity $f(c)H(\mu)$ is near-optimal (in terms of number of queries) for a weak oracle model.

4.1 Generalized Ternary Search

In this section, we first take a greedy approach to the content search problem as a baseline. Let the version space $\mathcal{V}_i$ be the set of points (hypotheses) consistent with the answers received from the oracle so far (at time i). After each query $\mathcal{O}_t(x, y)$ there are three possible outcomes of "x", "y" or "?". The Generalized Ternary Search algorithm (GTS), as a generalization of GBS [Dasgupta, 2004], greedily selects the query $\mathcal{O}_t(x, y)$ that results in the largest reduction in mass of the version space based on the potential outcomes. More rigorously, the next query to be asked by GTS is found through

$$\arg\min_{(x,y) \in \mathcal{V}_i^2} \max \Big(\sum_{z \in \mathcal{V}_i, \mathcal{O}_z(x,y)=x} \mu(z),$$
$$\sum_{z \in \mathcal{V}_i, \mathcal{O}_z(x,y)=y} \mu(z), \sum_{z \in \mathcal{V}_i, \mathcal{O}_z(x,y)=?} \mu(z) \Big). \quad (1)$$

The computational complexity of GTS is $\Theta(n^3)$ and thus makes it prohibitive for large databases. Also, while this type of greedy algorithms performs well in practice (in terms of number of queries), it is difficult

---

[5]We refer to it as an *abstention oracle* interchangeably.

[6]An oracle such that for any $A \subseteq \mathcal{M}$ identifies if $t \in A$ or not.

(or even impossible) to provide tight bounds for their performances [Dasgupta, 2004, Golovin and Krause, 2011]. Furthermore, the noisy oracle model makes analysis even harder. For this reason, we need more efficient algorithms with provable guarantees. The key ingredient of our methods is how to choose queries in order to find the target in a near optimal number of queries.

4.2 Full Knowledge of the Metric Space
The first algorithm we present (WORCS-I) relies on a priori knowledge of the distribution μ and the pairwise distances between all the points. We prove that WORCS-I (see Algorithm 1) locates the target t, where its query complexity is within a constant factor from the optimum. WORCS-I uses two main ideas: (i) it guarantees that, by carefully covering the version space, in each iteration the target t remains in the ball (which is found in line 11 of Algorithm 1), and (ii) the volume of version space decreases by a constant factor at each iteration. These two facts ensure that in a few iterations we can locate the target. We should mention that while WORCS-I relies on knowing all the pairwise distances, but we guarantee bounds for a c-doubling measure μ in the metric space $\mathcal{M}$.[7] Later in this section, we devise two other algorithms that relax the assumption of having access to the pairwise distances.

Theorem 1. *The average number of queries required by* WORCS-I *to locate an object is bounded from above by* $f(c) \cdot \left[1 + \frac{H(\mu)}{\log(1/(1-c^{-2}))}\right]$, *where* c *is the doubling constant of the measure* μ *in the metric space* $\mathcal{M}$ *and* $f(c)$ *is a polynomial function in* c.

In order to find a $\frac{\Delta}{8(\alpha+1)}$-cover in line 8 of Algorithm 1, we can generate a $(\Delta/2^l)$-net of $\mathcal{V}_i$, where $l = \lceil \log_2 8(\alpha+1) \rceil$. The size of such a net is at most $c^{\lceil \log_2 8(\alpha+1) \rceil + 3}$ by Lemma 1 in [Karbasi et al., 2012b]. Also, since we assume the value of c is constant, $f(c)$ is independent of the size of database; it remains constant and our bound depends on the size of database only through $H(\mu)$, which is growing with a factor of at most $O(\log|M|)$.

4.3 Partial Knowledge of the Metric Space
Despite the theoretical importance of WORCS-I, it suffers from two disadvantages: (i) First, it needs to know the pairwise distances between all points a priori, which makes it impractical in many applications. (ii) Second, experimental evaluations show that, due to the high degree of the polynomial function $f(c)$ (see proof of Theorem 1), the query complexity of WORCS-I is worse than baseline algorithms for small datasets. To overcome these limitations, we present another algorithm called WORCS-II.

---
[7]This is a fairly loose condition for a metric space.

Algorithm 1 WORCS-I

Input: Metric space $\mathcal{M}$ and oracle $\mathcal{O}_t(\cdot, \cdot)$
Output: Target t
1: $\mathcal{V}_0 \leftarrow \mathcal{M}$
2: $i \leftarrow 0$
3: **loop**
4:     **if** $|\mathcal{V}_i| \leq 2$ **then**
5:         Invoke $\mathcal{O}_t(x, y)$ and halt
6:     **end if**
7:     $\Delta \leftarrow$ diameter of $\mathcal{V}_i$
8:     Let C be a $\frac{\Delta}{8(\alpha+1)}$-cover of $\mathcal{V}_i$
9:     **for** each $c_i \in C$ **do**
10:         **if** for all c_j such that $d(c_i, c_j) > \frac{\Delta}{8}$: $\mathcal{O}_t(c_i, c_j) = c_i$ **then**
11:             $\mathcal{V}_{i+1} \leftarrow B_{c_i}(\frac{\Delta(\alpha+2)}{8(\alpha+1)})$
12:         **end if**
13:     **end for**
14:     $i \leftarrow i + 1$
15: **end loop**

WORCS-II allows us to relax the assumption of having access to all the pairwise distances. Indeed, the WORCS-II algorithm can work by knowing only relative distances of points, i.e., for each triplets x, y and z, it needs to know only if $\alpha d(x, y) \leq d(x, z)$ or $\alpha d(x, z) \leq d(x, y)$. Note that this is enough to find $Vor(x, y, \mathcal{V}_i)$ and $Vor(y, x, \mathcal{V}_i)$ for all points x and y. We refer to this information as $\mathcal{F}_1$. Information of $\mathcal{F}_1$ is applied to find the resulting version space after each query. Theorem 2 guarantees that WORCS-II, by using $\mathcal{F}_1$ and choosing a pair with a β approximation of the largest distance in each iteration, will lead to a very competitive algorithm.

Theorem 2. *Let* $l \triangleq \left\lceil \log_2\left(\frac{\alpha+1}{\beta}\right)\right\rceil$. *The average number of required queries by* WORCS-II *to locate an object is bounded from above by* $1 + \frac{H(\mu)}{\log(1/(1-c_{\text{strong}}^{-l}))}$, *where* c_{strong} *is the strong doubling constant of the measure* μ *in the metric space* $\mathcal{M}$.

From Theorem 2, we observe that the performance of WORCS-II improves by larger values of β. Generally, we can assume there is auxiliary information (called $\mathcal{F}_2$) that could be applied to find points with larger distances in each iteration. For example, WORCS-II-RANK refers to a version of WORCS-II where the ranking relationships between objects is provided as side information (it is used for $\mathcal{F}_2$). By knowing the rankings, we can easily find the farthest points from x. From the triangle inequality, we can ensure this results in a β of at least $1/2$. Also, WORCS-II by knowing all the pairwise distances (similar to WORCS-I), can always find a query with $\beta = 1$. For the detailed explanation of WORCS-II see Algorithm 2.

Algorithm 2 Worcs-II

Input: Oracle $\mathcal{O}_t(\cdot, \cdot)$, $\mathcal{F}_1$ and $\mathcal{F}_2$ ▶ $\mathcal{F}_1$ is used
    for finding the next version space after each query.
    Auxiliary information $\mathcal{F}_2$ is used for finding points
    with larger β.
Output: Target t
1: $\mathcal{V}_0 \leftarrow \mathcal{M}$ and $i \leftarrow 0$
2: **loop**
3:     **if** $|\mathcal{V}_i| \leq 2$ **then**
4:         Invoke $\mathcal{O}_t(x, y)$ and halt
5:     **end if**
6:     Find $x, y \in \mathcal{V}_i$ such that $d(x, y) \geq \beta \operatorname{diam}(\mathcal{V}_i)$
7:     **if** $\mathcal{O}_t(x, y) = x$ **then**
8:         $\mathcal{V}_{i+1} = \mathcal{V}_i \setminus Vor(y, x, \mathcal{V}_i)$
9:     **else if** $\mathcal{O}_t(x, y) = y$ **then**
10:        $\mathcal{V}_{i+1} = \mathcal{V}_i \setminus Vor(x, y, \mathcal{V}_i)\}$
11:     **else**
12:        $\mathcal{V}_{i+1} = \mathcal{V}_i \setminus (Vor(x, y, \mathcal{V}_i) \cup Vor(y, x, \mathcal{V}_i))$
13:     **end if**
14:     $i \leftarrow i + 1$
15: **end loop**

Although the theoretical guarantee for Worcs-II depends on the assumption that the underlying metric space is c_{strong}-doubling, the dependence in c_{strong} is through a polynomial function with a degree much smaller than $f(c)$. Note that the Worcs-II algorithm finds the next query much faster than Gts. Indeed, Gts needs $O(|V_i|^3)$ operations in order to find the next query at each iteration. It is easy to see, for example, Worcs-II-Rank finds the next query in only $O(|V_i|)$ steps. In the following, we present a version of Worcs-II that needs much less information in order to find an acceptable query.

4.4 Algorithm with Minimalistic Information

In this section, we present Worcs-II-Weak which needs only partial information about the relative distances. This is exactly the information from $\mathcal{F}_1$. Worcs-II-Weak, in each iteration, instead of picking two points with distance $\operatorname{diam}(\mathcal{V}_i)$ or $\operatorname{diam}(\mathcal{V}_i)/2$ (which is possible only with information provided from $\mathcal{F}_2$), uses Algorithm 3 to find the next query. From the result of Lemma 2, we guarantee that Algorithm 3 finds a pair of points with distance at least $\frac{\Delta}{2\alpha}$. The computational complexity to find such a pair is $O(|\mathcal{V}_i|^2)$.

Lemma 2. *Assume for x and y there exist no $z \neq y$ such that $\mathcal{O}_x(y, z) = y$, then $d(x, y) \geq \frac{\Delta}{2\alpha}$.*

Corollary 1. *From the results of Theorem 2 and Lemma 2, we conclude that the expected query complexity of* Worcs-II-Weak *is bounded from above by* $1 + \frac{H(\mu)}{\log(1/(1-c_{\text{strong}}^{-l}))}$ *with* $l = 1 + \lceil \log_2 \alpha(\alpha + 1) \rceil$.

The bounds in Theorem 2 and Corollary 1 depend on

the value of c_{strong} through the required number of iterations because the reduction in the size of version space is lower bounded by a function of c_{strong}. In Worcs-II at each iteration we only ask one question, therefore we do not have any dependence on a polynomial function of c_{strong}. Also, note that our algorithms do need to know the value of α.

Algorithm 3 Find x and y

Input: $\mathcal{V}_i$ and $\mathcal{F}_1$
Output: (x, y) with $d(x, y) \geq \frac{1}{2\alpha} \operatorname{diam}(\mathcal{V}_i)$
1: Uniformly at random pick $x \in \mathcal{V}_i$
2: **for** $y \in \mathcal{V}_i \setminus \{x\}$ **do**
3:     **if** $\forall z \in \mathcal{V}_i \setminus \{x, y\} : x \notin Vor(y, z, \mathcal{V}_i)$ **then**
4:         **return** (x, y)
5:     **end if**
6: **end for**

In this section, we presented four different algorithms to locate an object in a database by using a weak oracle model. These algorithms use different types of information as input. To provide guarantees for their performances, we need to make different assumptions on the structure of the underlying metric space (see Table 1). Worcs-I provides better theoretical results; The guarantee for Worcs-I depends on constant values of c which is a looser assumption than having a constant c_{strong}. In the next section, we will show that although the theoretical guarantee of Worcs-II is based on a stronger assumption over the metric space (c_{strong} vs c), in all of our experiments it shows better performances in comparison to Worcs-I. It finds the target with fewer queries and the computational complexity of choosing the next query is much less. We believe that for most real datasets both c and c_{strong} are small and close to each other. Therefore, the polynomial term $f(c)$ (defined in Theorem 1) plays a very important role in practice. Finally, providing guarantees for the performance of Gts or similar algorithms seems impossible [Dasgupta, 2004, Golovin and Krause, 2011].

5 Experiments

We compare Worcs-I and Worcs-II (Worcs-II-Weak and Worcs-II-Rank)[8] with several baseline methods. Recall that Worcs-I needs to know the pairwise distances between objects, and Worcs-II-Rank and Worcs-II-Weak need only information about ranking and partial ordering obtained from the weak oracle, respectively. For choosing the baselines, we followed the same approach as Karbasi et al. [2012a]. Our baseline methods are: (i) Gts. (ii) Random: The general framework of Random is the same as Al-

---

[8]In this section, Worcs-II-R and Worcs-II-W stand for Worcs-II-Rank and Worcs-II-Weak.

Table 1: Comparison between algorithms

Algorithm	Input information	Constraint
GTS	Weak rank information	No guarantees
WORCS-I	Pairwise distances	Doubling constant
WORCS-II-RANK	Rank information	Strong doubling constant
WORCS-II-WEAK	Weak rank information	Strong doubling constant

gorithm 2 except that in Line 6, RANDOM randomly samples a pair of points x and y in the current version space. (iii) FAST-GTS: In light of the computationally prohibitive nature of GTS, FAST-GTS is an approximate alternative to GTS. Rather than finding the exact minimizer, it randomly samples k pairs of points and finds the minimizer of Eq. (1) only in the sampled pairs. RANDOM and GTS can be viewed as a special case of FAST-GTS with parameter $k = 1$ and $k = \binom{|\mathcal{V}_i|}{2}$ (sampling without repetition), respectively. We take $k = 10$ in the experiments. (iv) MINDIST: In contrast to WORCS-II, MINDIST selects a pair of points that reside closest to each other. We use this baseline to highlight the importance of choosing a pair that attain the approximate maximum distance.

We evaluate the performance of algorithms through two metrics: (i) query complexity: the expected number of queries to locate an object, and (ii) computational complexity: the total time-complexity of determining which queries to submit to the oracle. We conduct experiments over the following datasets: *MovieLens* [Harper and Konstan, 2016], *Wine* [Forina et al., 1991], *Music* [Zhou et al., 2014], and Fisher's *Iris* dataset. The demand distribution μ is set to the power law with exponent 0.4. In order to model uncertain answers from the oracle $\mathcal{O}_t(x, y)$, i.e., for points t that are almost equidistant from both x and y, we use the following model: if $d(x, t) \leq d(y, t) < \alpha d(x, t)$, the oracle outputs x with probability $\frac{\log(d(y,t)/d(x,t))}{\log \alpha}$ and outputs "?" otherwise; similarly if $d(y, t) \leq d(x, t) < \alpha d(y, t)$, the oracle outputs y with probability $\frac{\log(d(x,t)/d(y,t))}{\log \alpha}$ and outputs "?" otherwise.

We present expected query complexity of all seven algorithms on each dataset in Table 2 (we have to use a table rather than a bar chart since the data are highly skewed). The corresponding computational complexity is shown in Fig. 1(a). It can be observed that WORCS-II-WEAK outperforms all other algorithms in terms of query complexity. WORCS-II-RANK is only second to WORCS-II-WEAK. Intuitively, selecting two distant items for a query leads into two large Voronoi cells around them, which partially explains the good performance of WORCS-II-RANK. However, it could leave the rest of the version space relatively small; therefore, it may not reduce the version space substantially when a

"?" response is received from the oracle. Unlike WORCS-II-RANK, WORCS-II-WEAK is only guaranteed to find a pair that attain a β-approximation of the diameter of the current version space, where $\beta \geq \frac{1}{2\alpha}$ as shown in Lemma 2, thereby resulting in a relatively more balanced division of the version space. This explains the slightly better performance of WORCS-II-WEAK versus WORCS-II-RANK. With respect to the computational complexity, RANDOM costs the smallest amount of computational resources since it does not need to algorithmically decide which pair of items to query. WORCS-II-WEAK is only second to RANDOM.

In this set of experiments, in order to study the scalability of algorithms, we use the *Music* dataset and sub-sampled 10 to 1000 items from the dataset. Query and computational complexity of different algorithms is shown in Figs. 1(b) and 1(c). We see the positive correlation between query/computational complexity and the dataset size. Since the curves are approximately linear in a log-log plot, the complexity of algorithms scales approximately polynomially in the dataset size. Specifically, the query complexity of WORCS-II-RANK and WORCS-II-WEAK scales approximately $O(\sqrt{N})$, where N is the dataset size. In Fig. 1(b), the curves of WORCS-II-RANK and WORCS-II-WEAK are the two lowest curves, which represent lowest query complexity.

The performance of algorithms for different values of α is another important aspect to study. We used the *Iris* dataset and varied α from 1 to 100. Query and computational complexity of different algorithms is shown in Figs. 1(d) and 1(e). A general upward tendency in computational complexity as α increases is observed in Fig. 1(e). With respect to the query complexity shown in Fig. 1(d), WORCS-I exhibits a unimodal curve. In fact, while a large α leads to a greater number of balls in the $\frac{\Delta}{8(\alpha+1)}$-cover (increases the query complexity in line 8 of Algorithm 1), it results in smaller balls and therefore a smaller version space in the next iteration (reduces the query complexity as in line 11 of Algorithm 1). This explains the unimodal behavior of the curve of WORCS-I. Other algorithms exhibit a general positive correlation with α, as a larger α tends to induce a more unbalanced division of the version space. Finally, we explore impact of demand distribution. We used the *Iris* dataset and varied the exponent

Table 2: Expected Query Complexity on Each Dataset

	Rand	Worcs-I	W-II-R[a]	W-II-W	MinDist	Gts	Fast-Gts
MovieLens	11.46	594.97	10.86	10.79	49.45	**9.01**	9.79
Wine	7.03	55.68	6.47	**6.11**	35.16	22.82	10.46
Music	14.31	648.12	12.71	**12.67**	143.61	36.90	27.78
Iris	7.86	164.42	7.56	**6.98**	19.85	9.81	9.54

[a]W-II-R and W-II-W are the abbreviation of Worcs-II-Rank and Worcs-II-Weak, respectively.
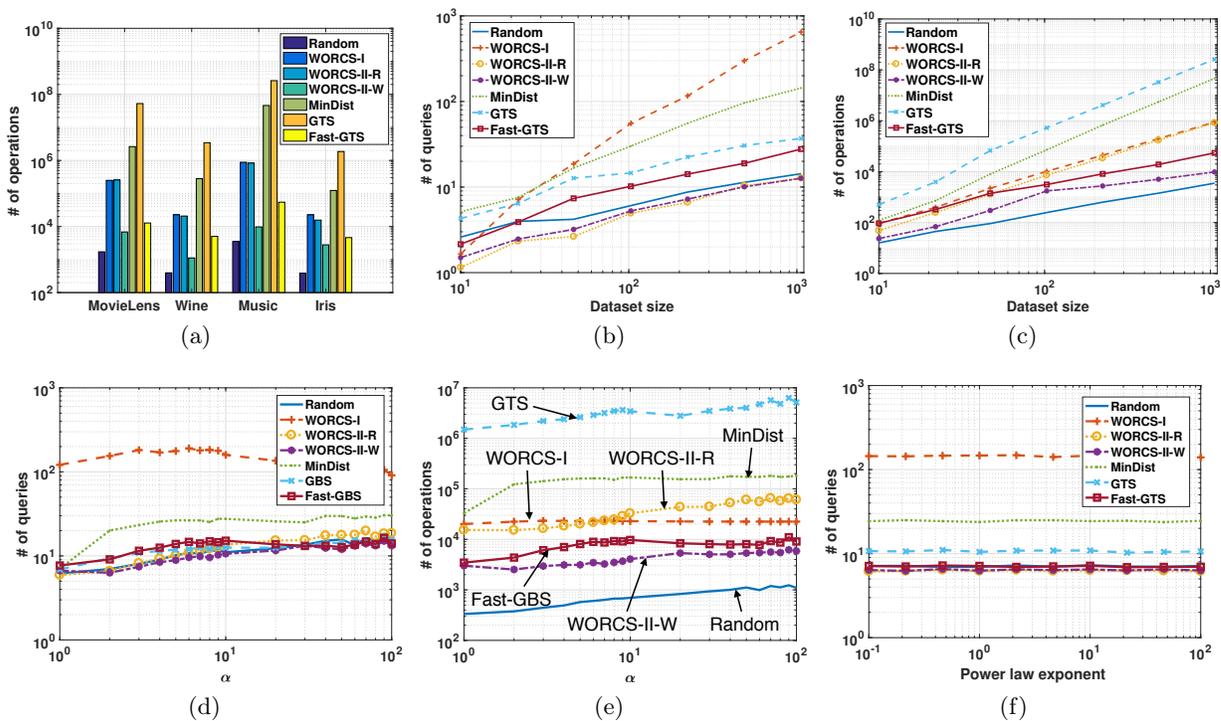
Figure 1: Fig. 1(a) shows the expected computational complexity, per search, of all seven algorithms applied to each dataset. Figs. 1(d) and 1(e) show the expected query and computational complexity as a function of the dataset size. We present how the expected query and computational complexity vary with α in Figs. 1(d) and 1(e). In Fig. 1(f), we present how the query complexity scales as a function of the power law exponent of the demand distribution.

of the power law distribution from 10^{-1} to 10^2. Query complexity of different algorithms is shown in Fig. 1(f). We observe that algorithms are generally robust to the change of the parameter of the power law demand distribution.

6   Conclusion

In this paper, we studied the problem of interactive learning though pairwise comparisons. We introduced a new weak oracle model to handle noisy situations, under which the oracle answers only when it is sure of the answer. We also considered the problem of comparison-based search via our weak oracle. We proposed two algorithms where they require different levels of knowledge about the distances or rankings between objects. We guaranteed the performance of these algorithms

based on a measure of intrinsic dimensionality for the distribution of points in the underling metric space. Finally, we compared our algorithms with several baseline algorithms on real datasets. Note that we assumed the oracle is non-malicious and when it is not confident, it answers with "?". Although this model is robust to high levels of uncertainty, but considering the effect of erroneous answers from the weak oracle is interesting for future work.

References

Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *J. Comput. Syst. Sci.*, 75(1):78–89, 2009.

Maria-Florina Balcan, Ellen Vitercik, and Colin White. Learning combinatorial functions from pairwise comparisons. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 310–335, 2016.

Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM, 2013.

Kenneth L Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-neighbor methods for learning and vision: theory and practice*, pages 15–59, 2006.

Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, volume 17, pages 337–344, 2004.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 235–242, 2005.

Brian Eriksson. Learning to top-k search using pairwise comparisons. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pages 265–273, 2013.

Donatella Firmani, Barna Saha, and Divesh Srivastava. Online entity resolution using an oracle. *Proceedings of the VLDB Endowment*, 9(5):384–395, 2016.

M Forina et al. An extendible package for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analisys and Technologies, Via Brigata Salerno*, 16147, 1991.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res. (JAIR)*, 42:427–486, 2011.

Navin Goyal, Yury Lifshits, and Hinrich Schütze. Disorder inequality: A combinatorial approach to nearest neighbor search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining,*

WSDM '08, pages 25–32, New York, NY, USA, 2008. ACM.

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

Siavash Haghiri, Debarghya Ghoshdastidar, and Ulrike von Luxburg. Comparison-based nearest neighbor search. In *Artificial Intelligence and Statistics*, 2017.

Sariel Har-Peled and Nirman Kumar. Approximate nearest neighbor search for low-dimensional queries. *SIAM Journal on Computing*, 42(1):138–159, 2013.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4): 19, 2016.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pages 507–513, 1997.

Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright. Active ranking from pairwise comparisons and the futility of parametric assumptions. *arXiv preprint arXiv:1606.08842*, 2016.

Kirsten Hildrum, John D Kubiatowicz, Satish Rao, and Ben Y Zhao. Distributed object location in a dynamic network. *Theory of Computing Systems*, 37 (3):405–440, 2004.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. Content search through comparisons. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 601–612, 2011.

Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. Comparison-Based Learning with Rank Nets. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012a.

Amin Karbasi, Stratis Ioannidis, and Laurent Massoulié. Hot or not: Interactive content search using comparisons. In *Information Theory and Applications Workshop (ITA), 2012*, pages 291–297. IEEE, 2012b.

David R Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM, 2002.

Richard M Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 881–890. Society for Industrial and Applied Mathematics, 2007.

Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

Yury Lifshits and Shengyu Zhang. Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–326. Society for Industrial and Applied Mathematics, 2009.

Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of plackett–luce models. In *Advances in Neural Information Processing Systems*, pages 172–180, 2015.

Lucas Maystre and Matthias Grossglauser. Just Sort It! A Simple and Effective Approach to Active Preference Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2344–2353, 2017.

Robert D. Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems*, pages 1366–1374, 2009.

Li Qian, Jinyang Gao, and HV Jagadish. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment*, 8(11):1322–1333, 2015.

P. Ram and A. G. Gray. Which space partitioning tree to use for search? In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 656–664, USA, 2013.

Matthew J Salganik and Karen EC Levy. Wiki surveys: Open and quantifiable social data collection. *PloS one*, 10(5):e0123483, 2015.

Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

Neil Stewart, Gordon DA Brown, and Nick Chater. Absolute identification by relative judgment. *Psychological review*, 112(4):881, 2005.

Omer Tamuz, Ce Liu, Serge J. Belongie, Ohad Shamir, and Adam Kalai. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 673–680, 2011.

Louis L Thurstone. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology*, 21(4):384, 1927.

Dominique Tschopp, Suhas Diggavi, Payam Delgosha, and Soheil Mohajer. Randomized algorithms for comparison-based search. In *Advances in Neural Information Processing Systems*, pages 2231–2239, 2011.

Dominique Tschopp, Suhas Diggavi, and Matthias Grossglauser. Hierarchical routing over dynamic wireless networks. *Random Structures & Algorithms*, 47(4):669–709, 2015.

Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity comparisons for interactive fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2014.

Jialei Wang, Nathan Srebro, and James Evans. Active collaborative permutation learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–511. ACM, 2014.

Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

Jiannan Wang, Guoliang Li, Tim Kraska, Michael J Franklin, and Jianhua Feng. Leveraging transitive relations for crowdsourced joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 229–240. ACM, 2013.

Fabian L Wauthier, Michael I Jordan, and Nebojsa Jojic. Efficient ranking from pairwise comparisons. *ICML (3)*, 28:109–117, 2013.

Yan Yan, Glenn M Fung, Rómer Rosales, and Jennifer G Dy. Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168, 2011.

Fang Zhou, Q Claire, and Ross D King. Predicting the geographical origin of music. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 1115–1120. IEEE, 2014.