Faster Algorithm for Truth Discovery via Range Cover*

Ziyun Huang[1] Hu Ding[2] Jinhui Xu[1]

[1] Department of Computer Science and Engineering
State University of New York at Buffalo
{ziyunhua, jinhui}@buffalo.edu
[2] Department of Computer Science and Engineering
Michigan State University
huding@msu.edu

Abstract. Truth discovery is a key problem in data analytics which has received a great deal of attention in recent years. In this problem, we seek to obtain trustworthy information from data aggregated from multiple (possibly) unreliable sources. Most of the existing approaches for this problem are of heuristic nature and do not provide any quality guarantee. Very recently, the first quality-guaranteed algorithm has been discovered. However, the running time of the algorithm depends on the spread ratio of the input points and is fully polynomial only when the spread ratio is relatively small. This could severely restrict the applicability of the algorithm. To resolve this issue, we propose in this paper a new algorithm which yields a $(1+\epsilon)$-approximation in near quadratic time for any dataset with constant probability. Our algorithm relies on a data structure called range cover, which is interesting in its own right. The data structure provides a general approach for solving some high dimensional optimization problems by breaking them down into a small number of parametrized cases.

1 Introduction

Truth discovery is an important problem arising in data analytics, and has received a great deal of attentions in recent years in the fields of data mining, database, and big data [3, 6–8, 4, 9–11]. Truth discovery seeks to find trustworthy information from a dataset acquired from a number of sources which may contain false or inaccurate information. There are numerous applications for this problem. For example, the latest search engines are able to answer user queries directly, instead of simply listing webpages that might be relevant to the query. This process involves retrieving answers from potentially a large number of related webpages. It is quite common that these webpages may provide inaccurate or inconsistent information. Thus a direct answer to the query needs the search

* The research of the first and third authors was supported in part by NSF through grants CCF-1422324, IIS-1422591, and CNS-1547167. The research of the second author was supported by a start-up fund from Michigan State University.

F. Ellen et al. (Eds.): WADS 2017, LNCS 10389, pp. 461–472, 2017.
DOI: 10.1007/978-3-319-62127-2_39

engine to be able to extract the most trustworthy information from all these webpages, which is exactly the problem of truth discovery.

Truth discovery is an unsupervised learning problem. Besides the input data, no prior knowledge about the reliability of each data source is provided. In such settings, an intuitive approach is to view all data sources equally reliable and obtain the solution by averaging or majority rule. A major issue of this approach is that the yielded answer may be quite far away from the truth. This is because a small number of unreliable data sources could significantly deviate the final solution. To deal with this issue, truth discovery treats data sources differently by estimating the reliability for each of them. This greatly increases the level of challenge for the problem. Moreover, since the truth discovery problem often occurs in big data scenarios, the number of data sources could be quite large and the dimensionality of the data could be rather high, which brings another dimension of challenge to the problem.

A widely accepted geometric modeling of the truth discovery problem is the follows. Data from each source is formulated as a set of real number attributes, and thus can be viewed as a vector in $\mathbb{R}^d$, where d is the number of attributes. Each data source is associated with a positive variable (or weight) representing its reliability. Formally, the truth discovery problem can be defined as follows.

Definition 1. *(Truth Discovery [4, 8]). Let $P = \{p_1, p_2, \ldots p_n\}$ be a set of points in $\mathbb{R}^d$ space, where each p_i represents the data acquired from the i-th source among a set of n sources. The truth discovery problem is to find the truth vector p^* and w_i (*i.e., *reliability) for each i-th source such that the following objective function is minimized.*

$$\min \Sigma_{i=1}^{n} w_i \|p_i - p^*\|^2, \text{ s.t. } \Sigma_{i=1}^{n} e^{-w_i} = 1. \tag{1}$$

The meaning of the above truth discovery formulation was discussed in [1] from an information theory's point of view. It is shown that the constraint on w_i in Definition 1 ensures that the **entropy** is minimized when p^* approaches the truth vector. For this reason, the problem is also called *Entropy based Geometric Variance* problem [1].

Despite extensive studies on this problem, most of the existing techniques are of heuristic nature, and do not provide any guarantee on the quality of solution. It is not until very recently that the truth discovery problem has a theoretically guaranteed solution [1]. This result ensures that a $(1+\epsilon)$-approximation of the problem can be achieved in $O(dn^2 + (n\Delta)^\sigma nd)$ time, where n is the number of input points (*i.e.,* data sources), d is the dimensionality of the space, Δ is the spread ratio of the input points (*i.e.* the ratio of the largest distance between any two input points to the smallest distance), and σ is any fixed small positive number. The result is based on an elegant sampling technique called Simplex Lemma [2] which is capable of handling high dimensional data. A main issue of this method is that its running time depends on the spread ratio of the input points, and is polynomial only when the spread ratio is relatively small (*i.e.,* $\Delta = O(\sqrt{n})$). This could severely restrict its applicability.

To overcome this main issue, we present in this paper a faster algorithm for the truth discovery problem. With constant probability, our algorithm achieves a $(1+\epsilon)$-approximation in $O(dn^2(\log n + \log d))$ time, and is completely independent of the spread ratio. Our algorithm is also space efficient, using only near linear space, while the space complexity of [1] also depends on the spread ratio. Our algorithm relies on a new data structure called *range cover*, which is interesting in its own right. Roughly speaking, range cover is a data structure designed for a class of optimization problems (in high dimensional space) which are decomposable into a number of "easier" cases, where each case can be characterized by a parameterized assumption. For example, truth discovery can be formulated as a problem of finding a truth vector $p^* \in \mathbb{R}^d$ from a given set P of points in $\mathbb{R}^d$ so that a certain objective function (the exact formulation will be discussed later) is minimized. We are able to show that although directly optimizing the objective function is challenging, the problem is much easier to solve if some additional information (*e.g.,* the distance r between p^* and P) is known. Thus, by viewing the additional information as a parameterized assumption, we can solve the truth discovery problem by searching for the best assumption. The range cover data structure shows that even though the number of parameterized assumptions could be very large (or even infinite), it is sufficient to sample only a small number of assumptions to ensure an approximate solution. This leads to a small-size data structure (*i.e.,* $O(n \log n)$ space) and a faster algorithm for truth discovery. Since the idea of decomposing problem into cases is not restricted only to the truth discovery problem, we expect that this data structure will provide new approaches to other problems.

2 Range Cover Data Structure

In this section, we present the aforementioned range cover data structure.

Range cover is motivated by several high dimensional optimization problems (such as truth discovery). In these problems, an input point set P is given in $\mathbb{R}^d$ space, and the objective is to find a point q in $\mathbb{R}^d$ so that a certain objective function is optimized. A commonly used approach for such problems is to examine a number of candidate points selected by some algorithms. But directly applying such an approach could require too many (*e.g.*, exponential in d) points to be examined in high dimensional space. A possible way to overcome this difficulty is to characterize all possibilities of q into a small number of cases so that in each case q is associated with a certain parametrized assumption which could help solve the problem more efficiently. For instance, in some optimization problem, q could be much easier to obtain if we know in advance the nearest neighbor (say p) of q in P and its distance r to q (*i.e.,* $\|p-q\| = r$) for some parameter r. We expect that these parameterized assumptions form a space with much lower dimensionality than d, and thus the overall time complexity can be significantly reduced.

From the above discussion we know that for the range cover data structure to be efficient, the problem needs to be decomposable into a small number

of "easier" cases. For this purpose, we will take advantage of the distribution of the points in P, such as their locality and point aggregation properties. To understand how point aggregation can be useful, consider the following parameterized assumption on q: Assume that p is the nearest neighbor of q in P and r is their distance. Denote this assumption by $\mathcal{NN}_q(p, r)$. If a subset of points, $v = \{p_1, p_2, \ldots, p_m\}$, are close to each other compared to r, *i.e.* their diameter $D(v)$ is no larger than λr for some predefined small constant $\lambda > 0$, then points in v can be viewed as a single 'heavy' point (simply denoted by v for convenience), and assumptions $\mathcal{NN}_q(p_1, r), \mathcal{NN}_q(p_2, r), \ldots, \mathcal{NN}_q(p_m, r)$ can be covered (or replaced) by a single assumption $\mathcal{NN}_q(v, r)$ without losing much quality. We formally define $\mathcal{NN}_q(v, r)$ for aggregated subset v as follows.

Assumption 1 *$\mathcal{NN}_q(v, r)$: For a subset v of P, $\mathcal{NN}_q(v, r)$ is an assumption made about q which says: $D(v) \leq \lambda r$ for some small constant $\lambda > 0$, where $D(v)$ is the diameter of v, and $r \leq \|p' - q\| \leq (1 + \lambda)r$ holds for p' which denotes the nearest neighbor of q in v.*

Another property of P which can be made use of is the *domination* relation. If q is very close to an aggregated subset of points $v \subseteq P$ compared to points in $P \setminus v$, it is often a degenerated case for the problem and relatively easy to solve. To cover such cases, we define the following assumption $\mathcal{DOM}_q(v)$ for predefined constants $\xi > 0$ and $\lambda > 0$.

Assumption 2 *$\mathcal{DOM}_q(v)$: For a subset v of P, $\mathcal{DOM}_q(v)$ is an assumption made about q, which says: there exists a point $p_v \in v$ such that $D(v) \leq \lambda \|q - p_v\|$ and $\|p_v - q\| \leq \xi \|p_{-v} - q\|$ for any point $p_{-v} \in P \setminus v$, where $D(v)$ is the diameter of v.*

With the above definitions of assumption, we know that the goal of the range cover data structure is to generate a small number of assumptions $\mathcal{DOM}_q(v_1)$, $\mathcal{DOM}_q(v_2), \ldots, \mathcal{DOM}_q(v_h)$ and $\mathcal{NN}_q(v'_1, r_1), \mathcal{NN}_q(v'_2, r_2), \ldots, \mathcal{NN}_q(v'_g, r_g)$, so that for any $q \in \mathbb{R}^d$, at least one of these assumptions holds. We call such a collection of assumptions an **assumption coverage**.

The main idea of range cover is to build a series of *views* of P formed by aggregated subsets from different scales of r, which is a controlling factor and can be interpreted as the distance of observation. Range cover identifies, for each r, a collection of disjoint aggregated subsets v of P with diameter no larger than λr for some predefined small constant $\lambda > 0$. The collection could be used as a sketch of P observed from distance r, which takes much less space than P. These views (from different distances r) jointly provide an easy way to access the "skeleton" information of P, which allow us to produce a smaller size assumption coverage. Particularly, for a given r, instead of generating assumptions $\mathcal{NN}_q(\{p\}, r)$ for each point $p \in P$, we produce coarse-grained assumptions $\mathcal{NN}_q(v, r)$ for every v in this view. Furthermore, by utilizing domination relation, we do not need to consider small values of r, and thus can further reduce the size of the assumption coverage. This is because the aggregation-based views of P from small enough r's correspond to situations where q is very close to some point and the domination

relation holds. Note that when determining point aggregation, we need not to consider too large r as well, since for large enough r the whole point set P is an aggregated set.

To generate the assumption coverage, an obvious challenge is how to reduce the number of possible values for r for which we need to build a view of P. Even though there is no need to consider too large and too small values for r, the gap between the maximum and minimum values often depends on the spread ratio of P, which could lead to pseudo-polynomial running time for some algorithms using the range cover data structure. Below we will show how to overcome this challenge and obtain a small size range cover.

2.1 Range Cover and Assumption Coverage

The range cover data structure uses the aggregation tree as an ingredient. The aggregation tree is a version of Hierarchical Well-Separated Tree (HST)[5] which is defined conveniently for point aggregation in a well-behaved manner. The definition is as follows.

1. Every node v (called *aggregation node*) represents a subset $P(v)$ of P, and the root represents P.
2. Every aggregation node v is associated with a representative point $l(v) \in P(v)$ and a size $s(v)$ which is an upper bound on the diameter of $P(v)$.
3. Every leaf node corresponds to one point in P with size $s(v) = 0$, and each point appears in exactly one leaf node.
4. The two children v_1 and v_2 of any internal node v form a partition of v with $\max\{s(v_1), s(v_2)\} < s(v)$.
5. For every aggregation node v with parent v_p, $\frac{s(v_p)}{r_{out}}$ is bounded by a polynomial function $\mathcal{P}(n, d) \geq 1$ (called *distortion polynomial*), where r_{out} is the minimum distance between any point in $P(v)$ and any point in $P \setminus P(v)$.

The following theorem shows that an HST with polynomial distortion (therefore, the aggregation tree also) can be built within near linear time.

Theorem 1. *[5] An HST with distortion polynomial $O(\sqrt{d}n^5)$ can be built in $O(dn \log n)$ time with success probability $1 - 1/n$.*

Below we will show how to build a range cover data structure from a given aggregation tree T_p which ensures to form an assumption coverage.

Consider an aggregation node v from distance r. If the diameter of v is not larger than λr for a predefined constant $\lambda > 0$, all points in v can be viewed as an aggregated subset and thus is part of the view from r. If r is so large that even the parent v' of v in T_p is an aggregated subset, v can be replaced by v' in the view. This means that an aggregation node v should not appear in the view from a far enough distance r. Also if r is small, either v has a too large diameter and thus cannot be an aggregated subset or v dominates q (*i.e.* the solution point). In the former case, v should be replaced by one of its descendant in the view. In the latter case, we do not include v in the view from distance r, with

the belief (which will be proved later) that the absence of v can be compensated by including the $\mathcal{DOM}_q(v)$ assumption in the assumption coverage.

The above observation implies that for any aggregation node v, there exists a range (r_L, r_H) of the value of r, such that v is only "visible" when r lies in the range. This immediately suggests the following scheme. Divide the set of all positive real numbers into intervals $((1+\lambda)^t, (1+\lambda)^{t+1}], t = \ldots, -2, -1, 0, 1, \ldots$, and associate each of them with a bucket. If an interval $(a, b]$ lies within the interval (r_L, r_H) of a aggregation node v, then insert v into the bucket of $(a, b]$. The collection of these buckets is then the desired range cover data structure.

Algorithm 1 RangeCover(T_p, λ, ξ)

Input: A aggregation tree T_p built over a set P of points in $\mathbb{R}^d$; an approximation factor $0 < \lambda < \frac{1}{4}$, a controlling factor $0 < \xi < 1$.
Output: A number of sets of aggregation nodes, each of which is associated with an interval $((1+\lambda)^t, (1+\lambda)^{t+1}]$ for some integer t.

1: For every interval $((1+\lambda)^t, (1+\lambda)^{t+1}]$, create an empty bucket B_t. (Note that B_t will not be actually created until some aggregation node v is inserted into it.)
2: For every non-root node v of T_p, let v_p be its parent in T_p, r_H be $s(v_p)/\lambda$, and r_L be $\max\{s(v)/\lambda, \xi s(v_p)/(16\mathcal{P}(n,d))\}$. Do
   – For every integer t satisfying the condition of $r_L \leq (1+\lambda)^t < r_H$, insert v into bucket B_t.

Given input P, for any constant factors $0 < \lambda < 1/4$ and $\xi > 0$ in **Assumption 1** and **Assumption 2**, we build the aggregation tree T_p and the corresponding range cover data structure $\mathcal{R}$ by calling RangeCover(T_p, λ, ξ), and let the assumption coverage $\mathcal{A}_{\lambda,\xi}$ (or simply $\mathcal{A}$ for convenience) contain the following assumptions:

1. $\mathcal{DOM}_q(v)$, for every aggregation node v of T_p
2. $\mathcal{NN}_q(v, r)$, for every aggregation node v of T_p and r such that interval $(r, (1+\lambda)r]$ is one of the nonempty bucket in $\mathcal{R}$ and v is a aggregation node in this bucket.

Clearly obtaining $\mathcal{A}$ from $\mathcal{R}$ is quite straightforward, and $|\mathcal{A}|$ has a size no larger than that of $\mathcal{R}$.

The following theorem shows that $\mathcal{A}$ is indeed an assumption coverage.

Theorem 2. *For any q in $\mathbb{R}^d$, at least one of the assumptions in $\mathcal{A}$ holds.*

Proof. Let p' be the nearest neighbor of q in P. If $\|q - p'\| = 0$, $\mathcal{DOM}_q(\{p'\})$ holds. In the following we assume that $\|q - p'\| > 0$. Let t' be the integer such that $(1+\lambda)^{t'} < \|q - p'\| \leq (1+\lambda)^{t'+1}$. Let v' be a aggregation node of T_p which is the highest ancestor of $\{p'\}$ in T_p such that $s(v') \leq \lambda(1+\lambda)^{t'}$. Since $\{p'\}$ is a leaf of T_p and $s(\{p'\}) = 0 \leq \lambda(1+\lambda)^{t'}$, such a v' always exists.

Based on the relationship between v', t' and the range cover data structure, we have 4 cases to consider. (a) v' is the root of T_p, (b) $(1+\lambda)^{t'} <$

$\max\{s(v')/\lambda, \xi s(v'_p)/(16\mathcal{P}(n,d))\}$, where v'_p is the parent of v' in T_p, (c) $(1+\lambda)^{t'} \geq s(v'_p)/\lambda$, and (d) $\max\{s(v')/\lambda, \xi s(v'_p)/(16\mathcal{P}(n,d))\} \leq (1+\lambda)^{t'} < s(v'_p)/\lambda$. Below we analyze each of them.

Case (a): Since $s(v') \leq \lambda(1+\lambda)^{t'} \leq \lambda\|q-p'\|$ and v' represents the whole point set P (as it is the root of T_p), we have $P \setminus v'$ is empty. This means that the assumption $\mathcal{DOM}_q(v')$ holds for q.

Case (b): Note that by the definition of t', we know that $(1+\lambda)^{t'} \geq s(v')/\lambda$. Therefore if case (b) occurs, we have $(1+\lambda)^{t'} \leq \xi s(v'_p)/(16\mathcal{P}(n,d))$. By $(1+\lambda)^{t'} < \|q-p'\| \leq (1+\lambda)^{t'+1}$ and $\lambda < 1$, it follows that $\|q-p'\| \leq \xi s(v'_p)/(8\mathcal{P}(n,d))$. Let p_o be any point in $P \setminus v'$. Then $\|p_o - p'\| \geq s(v'_p)/\mathcal{P}(n,d)$ by the property of aggregation tree. Therefore, $\xi\|p_o-p'\| \geq 8\|q-p'\|$. Thus, $\|p_o-q\| \geq \|p_o-p'\| - \|q-p'\| \geq (8/\xi-1)\|q-p'\|$. By the fact $\xi < 1$, we have $\|q-p'\| \leq \xi\|p_o-q\|$. Also since $(1+\lambda)^{t'} \geq s(v')/\lambda$ and $(1+\lambda)^{t'} < \|q-p'\| \leq (1+\lambda)^{t'+1}$, we have $\|q-p'\| \geq s(v')/\lambda$. This indicates that $\mathcal{DOM}_q(v')$ holds for case (b).

Case (c): This case actually never occurs. This is because, by the definition of v', $s(v'_p) > \lambda(1+\lambda)^{t'}$, since otherwise v' cannot be the highest ancestor of $\{p'\}$ satisfying the inequality $s(v') \leq \lambda(1+\lambda)^{t'}$.

Case (d): Note that this case means that v' is placed in bucket $((1+\lambda)^{t'}, (1+\lambda)^{t'+1}]$. Thus $\mathcal{NN}_q(v', (1+\lambda)^{t'})$ is in $\mathcal{A}$. We show that $\mathcal{NN}_q(v', (1+\lambda)^{t'})$ holds for q. Indeed, this follows immediately from previous discussion on v': $s(v') \leq \lambda(1+\lambda)^{t'}$ and $(1+\lambda)(1+\lambda)^{t'} \geq \|p'-q\| > (1+\lambda)^{t'}$.

Since in all cases at least one assumption in $\mathcal{A}$ holds for q, the theorem follows. □

The following theorem indicates that the size of the assumption coverage is small.

Theorem 3. *Given a aggregation tree T_p and factors $0 < \lambda < 1/4$ and $0 < \xi < 1$, the range cover data structure can be built in $O(1/\lambda \log(1/\xi) n(\log n + \log d))$ time and takes $O(1/\lambda \log(1/\xi) n(\log n + \log d))$ space. Consequently, $|\mathcal{A}| = O(1/\lambda \log(1/\xi) n(\log n + \log d))$.*

Proof. From **Algorithm 1**, we know that every aggregation node v is inserted into $O(\log_{1+\lambda} r_H/r_L)$ buckets (see Step 2 of the algorithm). Note that $\log_{1+\lambda} r_H/r_L$ is no larger than $\log_{1+\lambda}((s(v_p)/\lambda)/(\xi s(v_p)/16\mathcal{P}(n,d))) = O(1/\lambda \log(1/\xi)(\log n + \log d))$. Since the total number of aggregation node is $O(n)$, the theorem follows. □

3 Solving Truth Discovery with Assumption Coverage

In this section, we show how to use the assumption coverage to solve the truth discovery problem. Given any point set P in $\mathbb{R}^d$ and a small constant $0 < \epsilon < 1$, we first build an assumption coverage $\mathcal{A}$ with factors λ and ξ whose values depend on ϵ only and will be determined later. We then show how to obtain a $(1+\epsilon)$-approximation of the problem in polynomial time. Let p^* be the truth vector (*i.e.,* optimal solution) of the problem.

We first borrow a useful lemma from [6]. It shows that once p^* is determined, the weights w_i can also be determined. Thus we only need to find an approximate truth vector p^*.

Lemma 1. *[6] If the truth vector p^* is fixed, the following value for each weight w_l minimizes the the objective function (1) (**in Definition 1**),*

$$w_l = \log(\frac{\sum_{i=1}^{n} \|p^* - p_i\|^2}{\|p^* - p_l\|^2}). \tag{2}$$

There are two types of assumptions about p^* in $\mathcal{A}$ which covers all possibilities of p^*: $\mathcal{NN}_{p*}(v, r)$ and $\mathcal{DOM}_{p*}(v)$. Below we discuss each of them.

The following lemma shows that $\mathcal{DOM}_{p*}(v)$ is easy to solve.

Lemma 2. *By setting $\lambda \leq 1/4$ and $\xi \leq \epsilon/4$, if $\mathcal{DOM}_{p*}(v)$ holds for the truth vector p^*, there exists a point $p' \in v \subseteq P$ such that p' is a $(1+\epsilon)$-approximation of the truth discovery problem (using the objective function (1) in **Definition 1**).*

From the above lemma, we know that if $\mathcal{DOM}_{p*}(v)$ holds for some v, then one of the input point in P will be a $(1+\epsilon)$-approximation. This means that we can handle all such cases by trying every input point as p^* by computing the objective function (1) in equation (2), and choosing the one with the minimum objective value as the solution. This takes $O(dn^2)$ time.

The following lemma shows that $\mathcal{NN}_{p*}(v, r)$ can also be handled efficiently. We leave the proof to the next subsection.

Lemma 3. *If $\mathcal{NN}_{p*}(v, r)$ holds for any factor $0 < \lambda < 1/4$, then a $(1+\epsilon)$-approximation can be computed in time $O(dn)$ with constant probability, where ϵ is a small constant in $(0, 1)$.*

The above lemmas suggest that we can compute an approximate p^* by the following algorithm.

1. Compute an aggregation tree from P.
2. Set $\xi = \epsilon/4$, $\lambda = 1/5$, compute a range cover from the aggregation tree.
3. Compute $\mathcal{A}$ from the range cover.
4. Try every $p \in P$ as a candidate for the truth vector. Choose the one, say p_1, that minimizes the objective function.
5. For every $\mathcal{NN}_{p*}(v, r)$ in $\mathcal{A}$, compute a candidate for p^*. Choose the one, say p_2, that minimizes the objective function.
6. Choose from p_1 and p_2 the one that minimizes the objective function

In the above algorithm, Step 1 takes $O(dn \log n)$ time. Step 2 needs $O(n(\log n + \log d))$ time (where ϵ is hidden in the $O(\cdot)$ notion). Step 3 costs $O(n(\log n + \log d))$ time. Step 4 can be done in $O(dn^2)$ time. Step 5 takes $O(dn^2(\log n + \log d))$ time, since we test at most $O(n(\log n + \log d))$ assumptions in $\mathcal{A}$. Step 6 requires only $O(1)$ time. For the space usage, it can be computed $O(dn \log n) + O(n(\log n + \log d)) + O(n(\log n + \log d)) + O(dn) + O(dn) + O(1) = O(dn(\log n + \log d))$. Thus we have the following main theorem.

Theorem 4. *Given any set P of n points in $\mathbb{R}^d$, with constant probability, it is possible to compute a $(1+\epsilon)$-approximate solution for the truth discovery problem in $O(dn^2(\log n + \log d))$ time. The space usage can be made to $O(dn(\log n + \log d))$.*

3.1 Solving $\mathcal{NN}_{p*}(v, r)$

In this section we prove Lemma 3. We assume that $\mathcal{NN}_{p*}(v, r)$ holds for p^*, where $v \subseteq P$ and $r > 0$.

Lemma 1 reveals how the weight w_i of every $p_i \in P$ is related to p^*. It is clear from the objective function (1) and Lemma 1 that p^* is the weighted mean of P. Since we do not know p^* in advance, w_i is also unknown for every $p_i \in P$. The truth discovery problem can be viewed as a problem of finding the weighted mean of a point set with unknown weights. Our strategy for solving this problem consists of two main steps: (1) we partition P into a number of subsets (or sub-clusters), with each having some nice property. The weights of the points in some clusters are approximately known, while the weights of the points in other clusters are unknown, but have an upper and lower bound; (2) we apply a technique in [1] to find the approximate weighted mean point of each subset, and combine them to estimate p^*.

Partitioning P for Estimating Weights We first show how to estimate the weights of some points by $\mathcal{NN}_{p*}(v, r)$ without knowing p^*. This is crucial for our algorithm to be efficient for any point set P.

Let $p_1 \in v$ denotes the representative point $l(v)$ of v. We label the rest of points in P as $p_2, p_3, \ldots, p_n$. For each point $p_i \in P$, define $r'_i = \max(\|p_1 - p_i\|, r)$ and $r_i = \|p^* - p_i\|$. For $\mathcal{NN}_{p*}(v, r)$, let $p_{i_s} \in v$ be the nearest neighbor of p^* in P. Below we derive the relationship between r_i and r'_i.

First, we consider the case that $\max(\|p_1 - p_i\|, r) = r$. In this case, we have $r_i \geq \|p_{i_s} - p^*\| \geq r = r'_i$ by assumption $\mathcal{NN}_{p*}(v, r)$ and the fact that p_{i_s} is the nearest neighbor of p^*. Also we have $r_i \leq \|p_1 - p^*\| + \|p_1 - p_i\| \leq \|p_1 - p^*\| + r$, and

$$\|p_1 - p^*\| \leq \|p_1 - p_{i_s}\| + \|p^* - p_{i_s}\| \leq D(v) + (1 + \lambda)r \leq (1 + 2\lambda)r.$$

Thus, $r_i \leq (2 + 2\lambda)r = (2 + 2\lambda)r'_i$. Putting all together, we have $r'_i \leq r_i \leq (2 + 2\lambda)r'_i$.

Then, we consider the case that $\max(\|p_1 - p_i\|, r) = \|p_1 - p_i\|$. In this case, $r'_i = \|p_1 - p_i\| \geq r$. Again, we have $\|p_1 - p^*\| \leq \|p_1 - p_{i_s}\| + \|p^* - p_{i_s}\| \leq D(v) + (1 + \lambda)r \leq (1 + 2\lambda)r$. Therefore, $(1 + 2\lambda)r'_i \geq \|p_1 - p^*\|$. Thus,

$$r_i = \|p_i - p^*\| \leq \|p_1 - p_i\| + \|p_1 - p^*\| \leq \|p_1 - p_i\| + (1 + 2\lambda)r'_i = (2 + 2\lambda)r'_i.$$

Next, we consider 2 subcases, $r'_i \geq 2r$ and $r'_i < 2r$. If $r'_i < 2r$, since $r_i \geq r$, we have $r_i > r'_i/2$. If $r'_i \geq 2r$, since $\|p_1 - p^*\| \leq (1+2\lambda)r$, we have $\|p_1 - p^*\| \leq (1+2\lambda)r'_i/2$. This means that

$$r_i = \|p_i - p^*\| \geq \|p_1 - p_i\| - \|p_1 - p^*\| \geq r'_i - (1 + 2\lambda)r'_i/2 = (1 - 2\lambda)r'_i/2.$$

To conclude, we have $(1-2\lambda)r_i'/2 \le r_i \le (2+2\lambda)r_i'$.

From the above analysis and the fact that $\lambda < 1/4$, we can obtain the following.

$$r_i/4 \le r_i' \le 4r_i. \tag{3}$$

For each $p_i \in P$, let $w_i = \log((\sum_{p_j \in P} r_j^2)/(r_i^2))$, *i.e.,* w_i is the optimal weight determined by Lemma 1. Let $w_i' = \log((\sum_{p_j \in P} r_j'^2)/(r_i'^2))$. From inequality (3), we obtain the following:

$$w_i - \log 256 \le w_i' \le w_i + \log 256. \tag{4}$$

This means that w_i' can be used as an approximation of w_i if w_i is large enough.

For any $p_i \in P$, if $w_i' \ge 8/\beta \ge \log 256/\beta$ for any $0 < \beta < 1$, we have the following (by (4))

$$(1-\beta)w_i \le w_i' \le (1+\beta)w_i.$$

This means that w_i can be well approximated by w_i' in this case. Let P_β denote the set $\{p_i \in P | w_i' \ge 8/\beta\}$.

Next, we further show that there is at most one point p_i in P with weight $w_i < \log 36/25$ which, if exists, can be identified by a simple procedure. By the definition of w_i, we know that $w_i < \log 36/25$ can happen only when $\|p^* - p_i\| > 5\|p^* - p_j\|$ for any $i \ne j$. This means that for any $j, l \ne i$,

$\|p_j - p_l\| \le \|p^* - p_j\| + \|p^* - p_l\| \le 2\max(\|p^* - p_j\|, \|p^* - p_l\|)$. Thus, we have

$$\begin{aligned}\|p_j - p_i\| &\ge \|p^* - p_i\| - \|p^* - p_j\| \\ &> 5\max(\|p^* - p_j\|, \|p^* - p_l\|) - \max(\|p^* - p_j\|, \|p^* - p_l\|) \\ &= 4\max(\|p^* - p_j\|, \|p^* - p_l\|) \ge 2\|p_j - p_l\|.\end{aligned}$$

Hence, for any $j, l \ne i$, the inequality $\|p_j - p_l\| < \|p_i - p_j\|/2$ holds. In other words, p_i is isolated from the rest of the points in P. It is easy to see that such a p_i is unique, if exists. The following procedure searches for such a p_i.

1. Choose an arbitrary point p from P.
2. Find a point p' in P farthest away from p.
3. Find a farthest point p'' from p' in P.
4. Compare the pairwise distances among the three points in $\{p, p', p''\}$. Throw away the pair of points with the smallest pairwise distance. Output the remaining point as $\hat{p}$.

From the above discussion, it is easy to see that if there is a point p_i with weight $w_i < \log 36/25$, it must be $\hat{p}$. Clearly, this procedure takes only $O(dn)$ time.

For a constant $0 < \beta < 1/2$ (whose value will be determined later), let $P_u = P \setminus (P_\beta \cup \{\hat{p}\})$ and $P_< = \{\hat{p}\} \setminus P_\beta$. Then, $P_u, P_<, P_\beta$ form a partition of P. P_β contains all points p_i in P whose weights w_i have already been roughly determined (*i.e.,* approximated by w_i'); $P_<$ has at most one point, which will be

Algorithm 2 $(1 + O(1)\epsilon)$-approximate Truth Discovery from $\mathcal{NN}_{p*}(v, r)$

Input: A set P of n points in $\mathbb{R}^d$ space. Assumption $\mathcal{NN}_{p*}(v, r)$. $\beta = \epsilon^2$. Constants γ, k solved from $2\gamma\sqrt{k} \leq \epsilon^2$ and $k = \lceil \log_{1+\gamma} \frac{16}{\epsilon^2 \log 36/25} \rceil + 1$. $c_1 = \frac{4k}{\alpha\gamma^2} \log \frac{16k^2}{\gamma^2}$. $c_2 = \frac{4k}{\gamma^2}$. $\alpha = \epsilon^3\beta/48k$.
Output: An approximate truth vector.

1: Identify $P_<, P_\beta, P_u$ by computing w_i' for each $p_i \in P$.
2: Compute the weighted mean o_3' of P_β using weights w_i'.
3: Randomly sample c_1 points from P. Enumerate all subsets of c_2 points from the sample. Compute means of these subsets, and put all the means into a set M.
4: For every k-subset $\{o_1, \ldots o_k\}$ of M, apply SIMPLEX$(\epsilon^2, k, o_1, \ldots, o_k)$ to produce a grid. Put all grid points in into a point set G.
5: For every o_2' in G, if $P_<$ contains a point o_1', then build a grid by applying SIMPLEX$(\epsilon, 3, o_1', o_2', o_3')$; otherwise, build a grid using SIMPLEX$(\epsilon, 2, o_2', o_3')$.
6: Try all the grid points produced above. Output the one that minimized the objective function (1).

the one with weight smaller than $\log 36/25$, if exists; P_u contains all the remaining points whose weights are not known yet. $P_u, P_<, P_\beta$ together with w_i' can be obtained in $O(dn)$ time since it takes a total of $O(n)$ distance computations.

Following a similar idea in [1], we further decompose P_u by using the *log-partition* technique, where $\gamma > 0$ is a constant to be determined later. (Note that the log-partition cannot be explicitly obtained since we do not know the weights w_i. We assume that such a partition exists and will be used in our later analysis.)

Definition 2. *The log-partition of P_u divides points in P_u into k groups $\mathcal{G}_1, \ldots \mathcal{G}_k$ as follows, where $k = \lceil \log_{1+\gamma} \frac{16/\beta}{\log 36/25} \rceil + 1$: $\mathcal{G}_i = \{p_j \in P_u | (1+\gamma)^{i-1} \log 36/25 \leq w_j \leq (1+\gamma)^i \log 36/25\}$.*

Note that the above partition indeed involves all points in P_u. This is because by the definition of $P_<$ and P_β, and the fact that $(1-\beta)w_i \leq w_i' \leq (1+\beta)w_i$ for all point $p_i \in P_\beta$, we know that $\log 36/25 \leq w_i \leq 16\beta$ for each point $p_i \in P_u$. This implies that $\mathcal{G}_1, \ldots \mathcal{G}_k, P_<, P_\beta$ form a partition of P. Also, we apply log-partition to P_u instead of P as in [1]. In this way the value of k is bounded, making our algorithm efficient for any data.

Applying the Simplex Lemma Roughly speaking, Simplex Lemma in [1] provides a procedure SIMPLEX$(\epsilon, k, o_1, \ldots, o_k)$ to approximate the weighted mean of a partitioned point set $Q = \bigcup Q_i$, where ϵ is an approximate factor, k is an integer and every o_i is a point in $\mathbb{R}^d$. The procedure outputs a grid of size $((8k/\epsilon)^k)$ within $O((8k/\epsilon)^k)$ time which ensures that at least one of the grid points is close to the weighted mean of Q, if o_i is a good approximation of the weighted mean of Q_i.

Algorithm 2 shows how to use SIMPLEX to produce an approximate truth vector, given P partitioned into $P_<, P_\beta, P_u$ as above. The running time and

space usage match those appear in Lemma 3. To obtain a $(1+\epsilon)$-approximation, we only need to do a scaling on the constants without affecting the asymptotic running time.

Below we briefly explain the main steps of Algorithm 2. In Step 1 we partition P into $P_<, P_\beta, P_u$ as mentioned before. In Step 2 an approximate weighted mean of P_β is computed. In Steps 3 and 4, we try to guess k weighted means $\{o_1, \ldots o_k\}$ for the clusters $\mathcal{G}_1, \ldots \mathcal{G}_k$ resulted from the log-partition of P_u by using random sampling. We apply SIMPLEX to these approximate means $\{o_1, \ldots o_k\}$ to produce a small grid. The set G of grid points contains at least one point which is a good approximate weighted mean of P_u. In Steps 5 and 6, we already have approximate weighted means o'_1 and o'_3 of $P_<$ and P_β, respectively, and a set G which contains an approximate weighted mean o'_2 of P_u. We then try all possible o'_2 from G and use SIMPLEX on o'_1, o'_2, o'_3 to produce grids and one of such grids contains the desired approximation of the truth vector.

References

1. Ding, H., Gao, J., and Xu, J.: Finding Global Optimum for Truth Discovery: Entropy Based Geometric Variance. Leibniz International Proceedings in Informatics (LIPIcs), 32nd International Symposium on Computational Geometry (SoCG 2016), Vol. 51, 34:1-34:16(2016).
2. Ding, H. and Xu, J.: A Unified Framework for Clustering Constrained Data without Locality Property. Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA 2015), pp. 1471-1490, January 4-6, 2015, San Diego, California, USA.
3. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: The role of source dependence. PVLDB, 2(1): 550-561(2009).
4. Li, Y., Gao, J., Meng, C., Li, Q., Su, L., Zhao, B., Fan, W., Han, J.: A Survey on Truth Discovery, CoRR abs/1505.02463(2015).
5. Har-Peled, S.: Geometric approximation algorithms. Vol. 173. Boston: American mathematical society(2011).
6. Li, H., Zhao, B., Fuxman, A.: The Wisdom of Minority: Discovering And Targeting The Right Group of Workers for Crowdsourcing. Proc. of the International Conference on World Wide Web (WWW'14), pp. 165-176(2014).
7. Li, Q., Li, Y., Gao, J., Su, L., Zhao, B., Demirbas, M., Fan, W., Han, J.: A Confidence- Aware Approach for Truth Discovery on Long-Tail Data. PVLDB 8(4): 425-436(2014).
8. Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., Han, J.: Resolving Conflicts in Heterogeneous Data by Truth Discovery and Source Reliability Estimation. Proc. the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14), pp. 1187-1198(2014).
9. Pasternack, J., Roth, D.: Knowing what to believe (when you already know something). Proc. of the International Conference on Computational Linguistics (COLING'10), pp. 877-885(2010).
10. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose Vote Should Count More: Optimal Integration of Labelers of Unknown Expertise. Advances in Neural Information Processing Systems (NIPS'09), pp. 2035-2043(2009).
11. Yin, X., Han, J., and Yu, P.S.: Truth discovery with multiple conflicting information providers on the web: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07), pp. 1048-1052(2007).