A Probabilistic Framework for Estimating Pairwise Distances Through Crowdsourcing

Habibur Rahman
UT Arlington
habibur.rahman@mavs.uta.edu

Senjuti Basu Roy
NJIT
senjutib@njit.edu

Gautam Das
UT Arlington
gdas@uta.edu

ABSTRACT

Estimating all pairs of distances among a set of objects has wide applicability in various computational problems in databases, machine learning, and statistics. This work presents a *probabilistic framework for estimating all pair distances through crowdsourcing, where the human workers are involved to provide distance between some object pairs.* Since the workers are subject to error, their responses are considered with a probabilistic interpretation. In particular, the framework comprises of three problems : (1) Given multiple feedback on an object pair, how do we combine and aggregate those feedback and create a probability distribution of the distance? (2) Since the number of possible pairs is quadratic in the number of objects, how do we estimate, from the known feedback for a small numbers of object pairs, the unknown distances among all other object pairs? For this problem, we leverage the metric property of distance, in particular, the triangle inequality property in a probabilistic settings. (3) Finally, how do we improve our estimate by soliciting additional feedback from the crowd? For all three problems, we present principled modeling and solutions. We experimentally evaluate our proposed framework by involving multiple real-world and large scale synthetic data, by enlisting workers from a crowdsourcing platform.

1. INTRODUCTION

In this paper we investigate the following problem: *how to obtain pairwise distance values between a given set of objects by using feedback from a crowdsourcing platform?* This problem lies at the core of a plethora of computational problems in databases, machine learning, and statistics, such as top-k query processing, indexing, clustering, and classification problems. We consider an approach where feedback from the crowd is solicited in the form of simple pair-wise comparison questions. As an example, given two images (a, b), workers are asked to rate (in a scale of $[0, 1]$) how dissimilar these two images are. The worker response may be interpreted as the *distance* between the two images. Although the number of pairwise questions is quadratic in the number of objects, the main idea in this paper is to only involve the workers in answering a small number of key pair-wise questions, and to estimate the remaining pair-wise distances using the metric properties of the distance function, in particular the *triangle inequality property* [19] - a property that is true for distance functions that arise in many common applications.

Our iterative crowdsourcing distance estimation framework has three key probabilistic components. When we solicit distance information for a specific object pair from multiple workers, we recognize that due to the subjectivity of the task involved, workers may disagree on their feedback, or may even be uncertain about their own estimate. Thus we develop a probabilistic model for *aggregating* multiple workers feedback to create a single probability distribution of the distance learned about that object pair. Next, given that we have learned the distance distributions of several object pairs from the crowd, we *estimate* the probability distributions of the remaining pairwise distances by leveraging the triangle inequality property of the distances. Finally, if there is still considerable "uncertainty" in the learned/estimated distances and we have an opportunity to solicit additional feedback, we investigate *which object pair* should we choose to solicit the next feedback on. This iterative procedure is continued until all pair-wise distances have been learned/estimated with a desired target certainty (or alternatively, the budget for soliciting feedback from the crowd has been exhausted).

Novelty: There have been a few prior works that have studied computational problems using crowdsourcing that require distance computations. For example, entity resolution [25, 26, 5] problems investigate entity disambiguation, and [22] study top-k and clustering problems in a crowdsourced settings. However, these works have developed their formalism and solutions tightly knit to their specific applications of interest, and do not offer any obvious extension to solve other distance-based applications. For example,the work in [24] is focused on determining whether two objects are the same or not, and not on the broader notion of quantifying the amount of distance between them. In contrast, our proposed framework offers a unified solution to all these computational problems, as they all can leverage our distance estimation framework to obtain the distance between any pair of objects. Please note that once all pair distances are computed, finding the top-k objects, or finding the clusters of the objects is easier to compute. Hence, our problem is more general than the above mentioned body of works. We discuss related work more thoroughly in Section 7.

Challenges and Technical Highlights: There are substantial challenges in formalizing and solving the key prob-

lems that arise in our three probabilistic components. Perhaps the most straightforward is the first component, i.e., how to aggregate the feedbacks received from multiple workers into a single pdf that describes the distance between two objects. There has been several prior works on reconciling the answers from multiple workers, which range from simple majority voting to sophisticated matrix factorization techniques [7, 14] on binary data, or opinion pooling [12, 8, 4] on categorical data. However these methods are largely focused on aggregating Boolean/categorical feedback (e.g., "are these two entities the same?"), whereas in our case we need to merge the potentially diverging and uncertain numeric (distance) feedback from multiple workers into a single probability distribution.

The most challenging aspect of our framework is the second component. Knowing distance distributions of some of the object pairs from the crowd, we have to estimate the probability distributions of the distances of the remaining object pairs, by leveraging the metric property of the distance. While the intuitive idea is simple (e.g., "if a is close to b, and b is close to c, then a and c cannot be too far apart"), the problem is challenging because (a) the known distances themselves are distributions rather than deterministic quantities, and (b) the metric property imposes interdependence between all the pairwise distances in a complex manner. In fact, since there are $n(n-1)/2$ pair-wise distances (where n is the number of objects), each such distance can be assumed to be a random variable such that all distances are jointly distributed in a high dimensional $(n(n-1)/2)$ space with interdependencies governed by the triangle inequality. In principle, this joint distribution must be first computed, and then the (marginal) pdfs computed as estimates of the unknown distances. The unknown pairs cannot be estimated in isolation, as a small change in one pdf is likely to disrupt the joint distribution and the triangle inequality property impacting the other pdfs.

We argue that in certain cases, computing the joint distribution may require us to solve a mixture of *over and under-constrained* nonlinear optimization system, whereas in other cases it may reduce to solving an under-constrained system with many feasible solutions ([2]). For the former cases, we formalize the optimization problem as a combination of *least squares and maximum entropy* formulation and present algorithm `LS-MaxEnt-CG` that adopts a conjugate gradient approach [27, 10] to iteratively compute the joint distribution. For the latter cases, the problem reduces to that of maximizing entropy, and we present an algorithm `MaxEnt-IPS` that leverages the idea of *iterative proportional scaling* [23, 21] to efficiently converge to an optimal solution. Both of these solutions, while ideal, only work for small to moderate problem instances since they are exponential in the dimensionality of the joint distribution being estimated. Consequently, we also present a heuristic solution `Tri-Exp` that scales much better and can handle larger problem instances.

In the third component, our task is to decide, from among the remaining unknown object pairs, which one to select for soliciting distance feedback from the crowd. Intuitively, the selected object pair should be the one whose distance (after being learned from the crowd) is likely to reduce the "overall" uncertainty of the remaining unknown distance pdfs the most, i.e., minimize the *aggregated variance* of the remaining pdfs. To solve this problem in a meaningful way, it is critical to be able to model how workers are likely to respond to a solicitation, because their anticipated feedback needs to be taken into account for selecting the most effective pair. Finally, we also recognize that this approach of resolving one object pair at a time by the crowd may be sub-optimal and slow to converge. Thus, we also describe an extension where we "look ahead" and select multiple promising unresolved object pairs, and engage the crowd in simultaneously providing feedback for these pairs.

Summary of Contributions: In summary, we make the following contributions in this paper:

- We consider the novel problem of all-pairs distance estimation via crowdsourcing in a probabilistic settings.
- We identify three key sub-components of our iterative framework, and present formal definitions of problems and the solutions for each of the component (Sections 2,3,4,5).
- We experimentally evaluate our framework using both real world and synthetic datasets to demonstrate its effectiveness (Section 6).

2. DATA MODEL AND PROBLEM FORMULATIONS

We first describe the data model and then formalize the problems considered in this paper.

2.1 Data Model

Objects and Actual Distances: We are given a set $\mathcal{O}$ of n objects, with no two objects being the same. Objects could be images, restaurants, movies, etc. Let $d(i,j)$ be the actual distance between objects i and j. Assume that all distances are normalized within the interval $[0,1]$, where larger values denote larger distances, and that metric properties are satisfied, in particular the *triangle inequality* [19] or *relaxed triangle inequality* [9] property, as we define below. We are interested in using this property for learning all the $\binom{n}{2}$ pairs of distances.

Triangle Inequality Property: For every three objects (i,j,k) that comprise a triangle $\triangle_{i,j,k}$, $d(i,j) \leq d(i,k) + d(k,j)$ and $d(i,j) \geq \big|d(i,k) - d(k,j)\big|$.

To lift the strict notion of triangle inequality, one can consider *relaxed triangle inequality*, that assumes $d(i,j) \leq c.(d(i,k)+d(k,j))$, where c is a known constant that is not too large. Indeed, the *relaxed triangle inequality* [9] property allows us to effectively incorporate subjective human feedback from crowd workers.

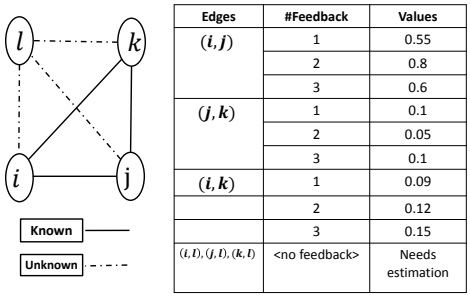
Question: A question $Q(i,j)$ to a worker requests feedback on her estimate of $d(i,j)$. The same question Q is directed to m different workers in the available workers pool, in order to gather multiple feedback.

Feedback: Let $f(i,j)$ represents a worker's feedback for the distance. The worker could either give a single value, or a range/distribution of values (if she is uncertain about the distance).[1] Even if the worker gives a single value, if it is known from past history of her performance that this worker is prone to making errors and is only correct with a certain probability p (say, 80%) (referred to as correctness

[1]The latter type of feedback is common in experts opinion aggregation problems [13], where a worker has partial knowledge on a particular topic and her answer reflects that with a distribution over the possible answers.

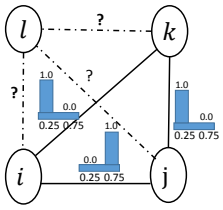
Notation	Interpretation
$\mathcal{O}$	set of n objects
$d(i,j)$	distance between objects i and j
$Q(i,j)$	asking distance on an object pair (i,j) in $[0-1]$ scale
$f(i,j)$	feedback on object pair (i,j)
$\triangle_{i,j,k}$	triangle formed by objects (i,j,k)
$d^k(i,j), d^u(i,j)$	known and unknown distance between an object pair (i,j), respectively
D^k, D^u	known and unknown set of distances, respectively
$\mathbf{D}$	distance vector
$Pr(\mathbf{D})$	joint probability distribution of $\mathbf{D}$
$\mathbf{W}$	vector representing all buckets of the multi-dimensional histogram of $Pr(\mathbf{D})$
m	m different feedbacks on the same question
$\mathbf{A}$	a Boolean matrix of constraints

Table 1: Notations



Edges	#Feedback	Values
(i,j)	1	0.55
	2	0.8
	3	0.6
(j,k)	1	0.1
	2	0.05
	3	0.1
(i,k)	1	0.09
	2	0.12
	3	0.15
$(i,l),(j,l),(k,l)$	<no feedback>	Needs estimation

(a) Illustration of Example 1



(b) Distances as Distributions (Histograms with $\rho = 0.5$)

Figure 1: **Illustrative Example.**

probability), then her single-value feedback can be converted to a more general probability distribution (pdf) over the range $[0,1]$ (e.g.,using techniques described in Section 3). We henceforth assume that the "raw" feedback of the worker has been appropriately processed into a pdf over $[0,1]$.

Known and Unknown Distances: Once a distance question $Q(i,j)$ has been answered by multiple workers, their respective feedbacks needs to be *aggregated* into a single pdf representing how the crowd has estimated the distance between i and j. Exactly how this aggregation is done is the first of the three key challenges of this paper, and is described in detail in Section 3. We denote the random variable described by this pdf as $d^k(i,j)$, where the superscript k denotes that the distance is now "known". Note that it still may not be the *actual* deterministic distance $d(i,j)$, unless the crowd's responses are completely error free, which is often not the case in practice.

Of the $\binom{n}{2}$ distances, let D_k represent the set of known distances, i.e., the ones for which feedback has been obtained from the crowd. Let D_u represent the remaining set of "unknown" distances, i.e., distances between those pair of objects for which feedback has not been explicitly obtained from the crowd. Consider $d^u(i,j) \in D_u$. Even though no information about this distance has yet been solicited, some distributional information about this distance can be derived since it depends on other pairwise distances in a complex manner (due to the triangle inequality property). We discuss this issue next.

Joint Distribution of All Pairs Distances: Consider the set of all distances $D_k \cup D_u$. We may view this set as a *distance vector* $\mathbf{D}$ of length $\binom{n}{2}$, whose every entry is a random variable representing the distance between the respective two objects (i,j). The space of all instances of $\mathbf{D}$ is $[0,1]^{\binom{n}{2}}$, however since the $\binom{n}{2}$ distances are interdependent upon each other due to the metric properties, the *valid instances* are those that satisfy the triangle property, i.e., for the triangle $\triangle_{i,j,k}$ defined by any three objects (i,j,k), the three corresponding distances should satisfy the triangle inequality.

Let $Pr(\mathbf{D})$ represent the joint probability distribution of $\mathbf{D}$. Our task is to estimate $Pr(\mathbf{D})$ such that the marginal distribution for a known random variable $d^k(i,j)$ should correspond to the pdf learned from the crowd. We note that once we have an accurate estimation of $Pr(\mathbf{D})$, we can get estimates of the distributions of the unknown random variables $d^u(i,j)$ by computing their marginals. In the next subsection we formalize the problems considered in this paper.

Table 1 summarizes the notations used in the paper.

EXAMPLE 1. *Image indexing for K-nearest neighbor queries: Our proposed framework is apt to process K-nearest neighbor queries over an image database, where, given a query image, the objective is to obtain an ordered list of images from the database, ordered by how closely they match the query image. To handle such queries faster, one potential avenue is to pre-process the image database and create an index that will cluster the images according to their distance among themselves. Then, as an example, if we have found that a query image $\mathcal{I}$ is far from a database image i and and if the indexes inform us that another image j is close enough to i, then, we may never need to actually compute the distance between $\mathcal{I}$ and j.*

With such an application in mind, consider a toy image database in Figure 1(a) with $n = 4$ images (i,j,k,l), where our eventual goal is to find the distances between all pairs of images. Assume that out of six possible pairs of distances, three are known: (i,j), (j,k), and (i,k). I.e., for each of these pairs, we have solicited feedback from several workers in the crowd, and aggregated the feedbacks to obtain a single probability distribution to describe the distance. The distances of the remaining three pairs are unknown and need to be estimated, again as probability distributions. Furthermore, if we need to solicit further feedback on a question, i.e., get the crowd to provide distance for an unknown pair, we intend to find what is the best question (best pair) to ask.

2.2 Problem Formulations

Recall from Section 1 that the iterative distance estimation framework involves three probabilistic components, which gives rise to three problems that need to be solved: (a) how to aggregate feedbacks from multiple workers for a specific distance question, (b) given some of the learned distances, how to estimate the remaining unknown distances, and (c) which object pair to select next for soliciting feedback from the crowd. In the remainder of this section, we provide formal definitions of these problems, and offer some insights into their complexities.

2.2.1 Problem 1: Aggregation of Workers Feedback for a Specific Object Pair

The first problem may be specified as follows:

PROBLEM 1. *Given a set of m feedbacks for the distance question $Q(i,j)$, where each feedback could be a pdf, aggregate those feedback to create a single pdf for the random variable $d^k(i,j)$.*

Using Example 1, this is akin to aggregating three different feedbacks from three different workers to compute $d^k(i,j)$.

2.2.2 Problem 2: Estimation of Unknown Distances

In this problem we need to leverage the known aggregated distances in D_k to estimate the remaining unknown distances D_u. Obviously, if the distances are completely arbitrary, the unknown distances cannot be computed from the known distances. However, if the distances are *metrics*, in particular satisfying the triangle inequality property, then this property can be leveraged in making better estimates of the unknown distances. Many well known distances are metric, such as, $\ell_2, \ell_1, \ell_\infty$, while other popular distances such as, Jaccard distance and Cosine distance could be transformed to metrics. For us, the challenge is to investigate how this property can be used in the case when the distances are probability distributions rather than fixed deterministic values.

Recall that $\mathbf{D}$ is a random vector representing all the $\binom{n}{2}$ distances, and $Pr(\mathbf{D})$ represents the joint distribution of $\mathbf{D}$. We now describe some important properties that $Pr(\mathbf{D})$ should possess.

The space of all instances of $\mathbf{D}$, i.e., $[0,1]^{\binom{n}{2}}$, may be divided into two as follows: (a) *Valid instances*, i.e., any instance of $\mathbf{D}$ such that all triangles $\triangle_{i,j,k}$ satisfy the triangle inequality, and (b) *Invalid instances*, i.e., any instance of $\mathbf{D}$ such that there exists a triangle $\triangle_{i,j,k}$ that does not satisfy the triangle inequality. Thus $Pr(\mathbf{D})$ should be a function constrained such that the cumulative probability mass over all valid (respectively invalid) instances of $\mathbf{D}$ should be 1 (respectively 0).

Additionally, $Pr(\mathbf{D})$ should be constrained such that the marginal distributions corresponding to the individual random variables in D_k (i.e. the known distances) should agree with the corresponding distance pdfs learned from the crowd. However, this constraint may not be always possible to satisfy, as crowd feedback is inherently an error-prone human activity, which can result in inconsistent feedback that violates the triangle inequality. Thus our task will be to estimate $Pr(\mathbf{D})$ such that the marginal distributions corresponding to individual random variables in D_k are "as close as possible" to the pdfs learned from the crowd.

Once such a $Pr(\mathbf{D})$ has been constructed, the pdfs of the unknown distances can estimated by computing the marginal distributions of each variable in D_u.

In the rest of this subsection, we provide more details of the problem formulation.

Discretization of the pdfs using Histograms: For computational convenience, for the rest of the paper we assume that (single or multi-dimensional) probability distributions are represented as discrete histograms, as is common in databases [17]. In particular, we assume that the $[0,1]$ interval is discretized into equi-width intervals of width ρ (where ρ is a predefined parameter). A r-dimensional pdf is thus represented by a r-dimensional histogram with $(\frac{1}{\rho})^r$ buckets. Each bucket contains a probability mass representing the probability of occurrence of its center value, and the sum of the probabilities of all buckets equals 1.

For the running example in Figure 1(a), we use $\rho = 0.5$. Thus a one-dimensional pdf is represented by a 2-bucket histogram, where the first bucket is between $[0 - 0.5]$ with center at 0.25 and the second bucket is $[0.5 - 1.0]$ with center at 0.75. Figure 1(b) of the running example shows how each known distance (known edge) is represented as a one-dimensional histogram after discretizing and aggregating inputs from multiple users, where the feedback values are replaced by the corresponding bucket centers (we describe details of our techniques for input aggregation, i.e., Problem 1, in Section 3).

Estimating $Pr(\mathbf{D})$: Once we have the histograms for each individual known edge, the joint distribution $Pr(\mathbf{D})$ needs to be estimated as a multi-dimensional histogram with $(\frac{1}{\rho})^{\binom{n}{2}}$ buckets. Our task is to estimate the probability mass of each of these buckets. Using the running example, there are 2^6 buckets, whose centers range from [0.25, 0.25, 0.25, 0.25, 0.25, 0.25] to [0.75, 0.75, 0.75, 0.75, 0.75, 0.75]. Computing the probability mass of a specific bucket, e.g., $Pr(0.25, 0.27, 0.25, 0.25, 0.25, 0.75)$, is equivalent of computing the probability of the simultaneous events $d(i,j) = 0.25$ & $d(j,k) = 0.27$ & $d(i,k) = 0.25$ & $d(i,l) = 0.25$ & $d(k,l) = 0.25$ & $d(j,l) = 0.75$. The computation of $Pr(\mathbf{D})$ can be modeled as a linear system with $(\frac{1}{\rho})^{\binom{n}{2}}$ unknowns, where each unknown represents the probability mass of a bucket. These unknowns have to satisfy three types of linear constraints:

(1) *Constraints imposed by the known pdfs*: $Pr(\mathbf{D})$ should be such that its marginal for any known distance $d^k(i,j)$ should satisfy the corresponding one-dimensional pdf learned from the crowd. Thus, each bucket of each known marginal pdf will generate a linear constraint. In our running example, a one-dimensional bucket such as $Pr(d(i,k) = 0.25)$ will generate a linear equation of the form $\sum Pr(*,*,0.25,*,*,*) = Pr(d(i,k) = 0.25)$.

(2) *Constraints due to triangle inequality*: Some of the buckets in the joint distribution must have zero probability mass if they violate triangle inequality constraints. In our running example, consider any of the 8 bucket of the form $(0.75, 0.25, 0.25, *, *, *)$. The probability mass of each such bucket has to be set to 0, since $d(i,j) = 0.75$, $d(j,k) = 0.25$ and $d(i,k) = 0.25$ does not satisfy the triangle inequality (this happens irrespective of any combination of the values for the remaining three edges, hence they are represented as '*').

(3) *Probability axiom constraint*: A final constraint requires

that the sum of all the buckets of the joint distribution adds up to 1. In our running example, this implies that $Pr(0.25, 0.25, 0.25, 0.25, 0.25, 0.25)+Pr(0.25, 0.25, 0.25, 0.25, 0.25, 0.75)+ \ldots + Pr(0.75, 0.75, 0.75, 0.75, 0.75, 0.75) = 1$.

If $\mathbf{W}$ represents the vector of $(\frac{1}{\rho})^{\binom{n}{2}}$ unknowns, and M represents the set of constraints, then the linear system may be expressed as $\mathbf{AW} = \mathbf{b}$, where $\mathbf{A}$ is a Boolean matrix of size $|M| \times (\frac{1}{\rho})^{\binom{n}{2}}$, and $\mathbf{b}$ is a vector of length $|M|$. Interestingly, as the following discussion shows, solving this linear system is not a straightforward task.

Scenario 1: Over-Constrained Case: In general, an *over-constrained linear system* $\mathbf{AW} = \mathbf{b}$ is one which has no feasible solution [15]. In our case, it is indeed possible that the marginal distributions corresponding to the individual random variables in D_k (i.e. the known distances) that are learned from the crowd gives rise to an over-constrained scenario. This is because crowd feedback is inherently an error-prone human activity, which can result in inconsistent feedback that violates the triangle inequality. For example, $\triangle_{i,j,k}$ in Example 1 has only one deterministic instance with edge weights $d(i,j) = 0.75$, $d(j,k) = 0.25$ and $d(i,k) = 0.25$. Clearly, $\triangle_{i,j,k}$ does not satisfy the triangle inequality, since $d(i,j) > d(i,k) + d(j,k)$. Hence, there is no valid joint distribution $Pr(\mathbf{D})$ which can estimate the known pdfs.In such cases, we estimate $Pr(\mathbf{D})$ such that the marginal distributions corresponding to individual random variables in D_k are "as close as possible" (using *least squares* principle) to the pdfs learned from the crowd. More formally, given $\mathbf{A}$ and $\mathbf{b}$, we estimate $\mathbf{W}$ such that $||\mathbf{AW}-\mathbf{b}||^2$ is *minimized*.

Scenario 2: Under-Constrained Case: In general, an *under-constrained linear system* $\mathbf{AW}=\mathbf{b}$ is one which has multiple feasible solutions [15]. In our case, while estimating $\mathbf{W}$, we may also encounter under-constrained scenarios. Using Example 1 and considering triangle $\triangle_{i,j,l}$, we note that any of the following solutions are feasible: $d(i,l) = 0.75$, $d(l,j) = 0.75$, or $d(i,l) = 0.75$, $d(l,j) = 0.25$, or $d(i,l) = 0.25$, $d(l,j) = 0.75$. In such cases, *maximum entropy* principles [23] are used to choose a solution that is consistent with all the constraints but otherwise is as uniform as possible. More formally, the objective is to solve the linear system $\mathbf{AW}=\mathbf{b}$ that maximizes the entropy of the joint distribution $-\sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$.

Scenario 3: Combined Case: Since our problem instances may involve both over and under-constrained scenarios, we unify both into a single minimization problem using a weighted linear combination, where the weight λ can be used to tune the solution to ensure better least square or higher uniformity. Our final problem is described as follows:

PROBLEM 2. *Estimate the joint distribution vector* $\mathbf{W}$ *such that* $f(\mathbf{W}) = \lambda \times ||\mathbf{AW}-\mathbf{b}||^2 + (1-\lambda) \times \sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$ *is* minimized.

Before we move to our next problem definition, we point out an interesting issue. The exponential size of Problem 2 (the number of buckets in the multi-dimensional histogram is intractably large for most real-world instances) suggests that a complete solution of Problem 2 is prohibitive. Fortunately, we observe that computing the joint distribution is merely a intermediate (and not strictly necessary) objective - our eventual objective is to estimate the one-dimensional pdfs of the unknown distances $d^u(i,j)$. This issue is discussed in more detail in Section 4, and in particular we present heuristics to directly compute the unknown one-dimensional pdfs without having to compute the intermediate joint distribution.

2.2.3 Problem 3: Asking the Next Best Question

Recall that our overall approach is an iterative process. If we have the need to solicit further feedback from the crowd, we have to select an object pair from D_u, as human workers have not yet been involved in providing feedback about such pairs. Our objective is to select the most promising pair, i.e., that is most likely to reduce the "uncertainty" of the remaining unknown distances the most. We measure uncertainty by aggregating the *variances* of the remaining unknown distance pdfs (the variance of $d^u(i,j)$ with mean μ is measured as $\sigma^2_{d^u(i,j)} = \sum_{\forall q} p_q * (q-\mu)^2$).

PROBLEM 3. *From the set* D_u *of the candidate object pairs, choose the next best question* $Q(i,j)$ *to solicit feedback from the human workers, such that, upon receiving the feedback, the* aggregated variance *over the remaining unknown distances is minimized.*

Aggregated variance, `AggrVar` is formalized in one of the two natural ways, *average variance or largest variance*:
(1) Average variance over the remaining unknown distances:

$$\frac{\sum \sigma^2_{d^u(i',j')}}{|D_u| - 1}, d^u(i',j') \in \{D_u - d^u(i,j)\}. \quad (1)$$

(2) Largest variance over the remaining unknown distances:

$$\max_{d^u(i',j')} \sigma^2_{d^u(i',j')}, d^u(i',j') \in \{D_u - d^u(i,j)\}. \quad (2)$$

Considering Example 1, this problem will seek to choose the next best question (i.e., edge or object pair) from $D_u = \{(i,l),(j,l),(k,l)\}$.

3. PROBLEM 1: AGGREGATION OF WORKERS FEEDBACK

In this section, we describe our proposed solution `Conv-Inp-Aggr` of aggregating multiple feedbacks on a single object pair (i.e., an edge).

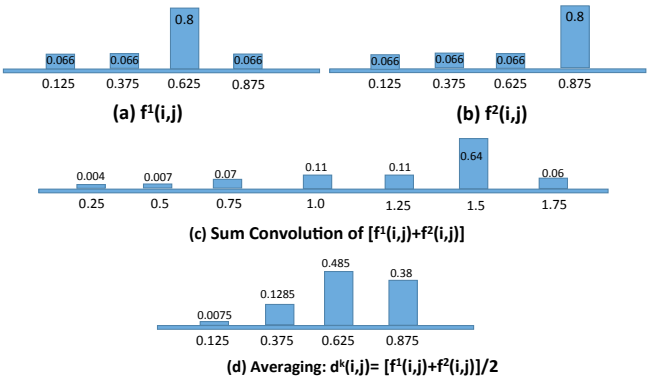


Figure 2: Worker Feedback Aggregation

In general, given a set of m different feedbacks $f^1(i,j), f^2(i,j), \ldots f^m(i,j)$, where each feedback is a random variable describing distance on an object pair (i,j), such that the set of random variables are *independently distributed*, our objective is to define a new random variable

whose distribution represents the *average* of the underlying input pdfs, i.e., pdf of $\frac{f^1(i,j)+f^2(i,j)+...+f^m(i,j)}{m}$. The independence assumption allows us to use the prior technique of *sum-convolution* [1] to obtain the sum of the input pdfs and then averaging that convolved pdf to obtain the average.

Algorithm 1 `Conv-Inp-Aggr`

1: **Input:** Set of m feedbacks for (i, j).
2: Perform a sequence of $m-1$ Sum-convolutions over the feedback pdfs.
3: Calculate $d^k(i, j)$ by re-calibrating the resultant pdf of previous step into pre-specified adjusted range. This step require averaging over the bucket values and reallocate the probability masses accordingly.
4: **return** $d^k(i, j)$

We illustrate this approach using the first two feedbacks for the pair (i, j) in our running example in Figure 1(a). The first worker's feedback (denoted as $f^1(i, j)$) of 0.55 is converted into a pdf. This is shown in Figure 2(a) as a 4-bucket histogram (i.e., with $\rho = 0.25$, buckets with boundaries $[0-0.25]$, $[0.25-0.5]$, $[0.5-0.75]$, $[0.75-1.0]$, and centers at 0.125, 0.375, 0.625, 0.875 respectively). As the feedback value 0.55 is in $[0.5-0.75]$, we can assign a probability mass of 1 to this bucket, and 0 to all other buckets. However, if we have prior information that the worker is only correct 80% of the time (correctness probability $p = 0.8$), we can assign a probability mass of 0.8 to the bucket $[0.5-0.75]$, and distribute the remaining probability mass uniformly among the remaining three buckets. This latter approach is used to generate the pdf illustrated in Figure 2(a). Similarly, Figure 2(b) shows the pdf for feedback 2 of (i, j).

The sum-convolution of these two pdfs is presented in Figure 2(c). Since the centers of the buckets of each of the individual pdf are between $[0.125, 0.875]$, their sum can be any value between $[0.25, 1.75]$. For each discrete value x between $[0.25, 1.75]$, the probability of $f^1(i, j)+f(^2(i, j)$ equal to x is calculated by computing the joint probability of $f^1(i, j) = x'$ and $f^2(i, j) = x$", such that, $x' + x" = x$.

With $m = 2$ feedbacks, the bucket values are then reassigned to the centers as follows: $0.25 \rightarrow 0.125$, $0.5 \rightarrow 0.25$, ..., $1.75 \rightarrow 0.875$. After this is done, if we have a transformed bucket center with non-zero probability that does not correspond to any of the input buckets, then the mass of that bucket is redistributed to its closest bucket. When two buckets are equally close, the mass is equally divided between the two buckets. As an example, since $1.0 \rightarrow 0.5$ after averaging, but 0.5 does not correspond to any bucket center , the probability mass of $Pr(f^1(i, j) + f^2(i, j) = 1.0)$ gets uniformly split between its two closest centers 0.375 and 0.625. The resultant distribution is given in Figure 2(d).

Figure 1(b) shows the aggregation results for (i, j) of Figure 1(a) with worker being completely accurate ($p = 1.0$) and with $\rho = 0.5$.

Running Time: If each pdf is approximated using an equi-width histogram of width ρ, the time to perform average convolution involving m different pdfs is $O(m \times 1/\rho^2)$ [1].

4. PROBLEM 2: ESTIMATION OF UNKNOWN DISTANCES

In this section, we present our proposed solutions of the problem 2- i.e., how to estimate the distance of the unknown object pairs from the given known distances. Using Example 1, this step is to estimate three unknown distances $D_u = \{(i, l), (j, l), (k, l)\}$, by leveraging the three known distances. We present two alternative solutions - an optimal solution by computing joint distribution that is exponential to the number of object pair $\binom{n}{2}$, and a much faster heuristic alternative.

4.1 Algorithms for Optimal Solution

Recall our proposed formulation in Section 2.2 and note that the optimal solution of computing the unknown distances is to first produce a joint distribution $Pr(\mathbf{D})$ on a high-dimensional space over all $\binom{n}{2}$ object pairs. This is due to our underlying abstraction that assumes that all objects are connected to each other which gives rise to a complete graph - hence the distribution of an unknown edge can not simply be learned in isolation. Once the joint distribution is obtained, the unknown pdfs are to be computed as marginals from the joint distribution. We investigate and design algorithms for the following two scenarios:

(1) As demonstrated in Example 1, our problem can unfortunately be both *over as well as under-constrained*. In fact, when the known pdfs are inconsistent (i.e., do not satisfy triangle inequality), there may not be any *feasible solution* to compute $Pr(\mathbf{D})$ that satisfies all the known pdfs. At the same time, a part of our solution space may still be under-constrained, especially the part that involves the unknown pdfs where multiple feasible solutions may exist.

(2) For the special case when the known pdfs are consistent, the scenario is merely under-constrained and may have multiple feasible solutions, as we describe in Section 4.1.2.

4.1.1 Combined Case

For this scenario, the problem of computing the joint distribution is formalized as an optimization problem (Problem 2) with the objective to minimize a weighted linear combination of least square and negative entropy (notice $-Pr(w) \log Pr(w)$ is the entropy), i.e., $f(\mathbf{W}) = \alpha \times ||\mathbf{AW} - \mathbf{b}||^2 + \beta \times \sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$ is to be *minimized*. The first part of the formulation is designed for the over-constrained settings, i.e., we satisfy the known pdfs as closely as possible if there is no feasible solution, whereas the second part of the formulation is to handle under-constrained nature of the problem through maximum entropy modeling that will choose the joint distribution model that is consistent with all the constraints but otherwise is as uniform as possible. From the joint distribution $Pr(\mathbf{D})$, we obtain the unknown distance pdfs by computing appropriate marginals.

LEMMA 1. *$f(\mathbf{W})$ is convex.*

PROOF. (Sketch) It can be shown that the linear aggregation of two convex functions is always convex [3], which proves the above lemma. □

Algorithm `LS-MaxEnt-CG`: Based on Lemma 1 $f(\mathbf{W})$ is convex. We propose Algorithm `LS-MaxEnt-CG`, by appropriately adapting nonlinear conjugate gradient algorithms [27, 10] that are popular iterative algorithms to solve such nonlinear convex optimization problems. The overall pseudocode is presented below in Algorithm 2.

Using Example 1 with $\rho = 0.5$, the joint distribution produces the probability for each of the 2^6 buckets that sum up to 1. From this joint distribution, the marginal distributions can be computed for the three unknown edges. (i, l) :

Algorithm 2 `LS-MaxEnt-CG`

1: **Input:** matrix A, constraint vector b, vector $\mathbf{W}$ with $\frac{1}{\rho}^{\binom{n}{2}}$ unknown variables, tolerance error η.
2: Initialize $\mathbf{W}$ with the steepest direction in the first iteration $\Delta\mathbf{W}_0 = -\nabla_{\mathbf{W}} f(\mathbf{W}_0)$
3: In the i-th iteration, compute β_i' using Fetcher-Reeves method [11].
4: Update the conjugate direction: $s_i = \Delta\mathbf{W}_i + \beta_i' s_{i-1}$.
5: Perform a line search to obtain α_i', $\alpha_i' = \arg\min_{\alpha'} f(\mathbf{W}_i + \alpha' s_i)$.
6: Update the position: $\mathbf{W}_{i+1} = \mathbf{W}_i + \alpha_i' s_i$
7: Repeat Steps $3-7$ to until the *error* $\leq \eta$.
8: **return** $f(\mathbf{W})$

$[0.25 : 0.366, 0.75 : 0.634]$,$(j,l)$: $[0.25 : 0.366, 0.75 : 0.634]$, (k,l) : $[0.25 : 0.366, 0.75 : 0.634]$.

Running Time: It has been shown in [10] that conjugate gradient has a running time complexity of $O(m'\sqrt{\kappa})$, where m' is the number of non-zero entries in the matrix $\mathbf{A}$ and κ is the number of iteration before convergence. However, in our case, as described in Section 2.2, the size of the input matrix $\mathbf{A}$ itself is exponential to the number of object pairs.

4.1.2 Under-Constrained Case

For the under-constrained settings, the optimization function becomes simpler, with the objective to maximize entropy $f(\mathbf{W}) = -\sum_{w\in\mathbf{W}} Pr(w)\log Pr(w)$, while satisfying the known constraints. Each constraint C_i is a restriction on some subset of these possible $(\frac{1}{\rho})^{\binom{n}{2}}$ cells to sum up to some observed value $p(C_i)$. More specifically, each $C_i = \sum(w_i \times I_{i,j})$, where $I_{i,j} = 1$ if j-th cell is included in the constraint C_i, and 0 otherwise.

Algorithm `MaxEnt-IPS`: It has been shown that the objective function always has a unique solution as long as the constraints are consistent [21]. Of course, this problem can be solved using a general purpose optimization algorithm. However, we propose `MaxEnt-IPS`, an *iterative proportional scaling (or IPS)* algorithm [23, 21] that exploits the structural property of the objective function and uses the observation that the optimal w_i values can be expressed in the following product form.

$$w_j^{\mu} = \mu_0 \Pi_{C_i} {\mu_i}^{I_{i,j}}$$

For each constraint C_i, there is a constraint μ_i that gets updated inside the IPS algorithm and μ_0 is a normalization constant to ensure that all cells add up to 1. This algorithm iteratively updates the μ_i's and the cell values w_i's. It is guaranteed to converge to the optimal solution as long as all constraints are consistent. Once the histogram buckets W and hence the joint distribution $Pr(\mathbf{D})$ is computed, the unknown marginals are obtained similarly as before. We omit further details and the pseudo-code for brevity but refer to [23, 21] for for more information on the IPS method.

`MaxEnt-IPS` does not converge for the input presented in Example 1 (b), as it is over-constrained. However, if we modify the example such that the aggregated feedback for (j,k) is 0.75 instead of 0.25, then the following outputs are obtained for the three edges: (i,l) : $[0.25 : 0.333, 0.75 : 0.667]$,$(j,l)$: $[0.25 : 0.333, 0.75 : 0.667]$, (k,l) : $[0.25 : 0.333, 0.75 : 0.667]$.

Running Time: The maximum entropy modeling is known to be NP-hard [18]. The `MaxEnt-IPS` algorithm terminates based on the convergence of all the μ's. In each iteration it makes updates to all the buckets in the joint distribution, which is exponential in size $(O(\frac{1}{\rho})^{\binom{n}{2}})$. If `MaxEnt-IPS` requires κ iterations to converse, the asymptotic complexity of this algorithm is exponential, i.e., $O(\kappa \times (\frac{1}{\rho})^{\binom{n}{2}})$.

4.2 Efficient Heuristic Algorithm

Both the problem variants and their respective solutions studied in Sections 4.1.1 and 4.1.2 first compute the joint distribution over an $\binom{n}{2}$-dimensional space as optimization problems. After that, the unknown distributions are computed from the joint distribution. Even with $n = 5$ objects and $\rho = 0.5$, the joint distribution is to be computed on an $2^{\binom{5}{2}} = 2^{10}$ dimensional space. Due to its exponential nature, computing the joint distribution is practically impossible as n increases. As a realistic alternative, we next present `Tri-Exp`, an *efficient heuristic algorithm that avoids computing the entire joint distribution, but explores the individual triangles in a greedy manner to estimate the pdfs of the unknown edges.* The pseudo-code is presented in Algorithm 3.

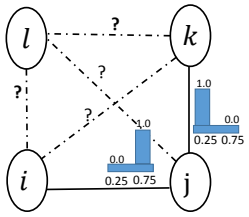


Figure 3: Example to Illustrate `Tri-Exp`

While Algorithm `Tri-Exp` avoids computing the joint distribution and instead performs a greedy exploration over the individual triangles one-by-one, there are still considerable challenges - each unknown object pair (edge) is involved in $n-2$ different triangles (with different triangle inequality constraints) and the algorithm must be adapted to estimate the pdf of the unknown edge such that it satisfies all the triangles. In particular, it encounters two scenarios.

Scenario 1: During execution, the algorithm may encounter some triangles which have two edges already known and only the third edge is to be estimated. For such cases, the algorithm *will greedily select that unknown edge that completes the highest number of triangles*, once estimated. When an unknown edge is involved in multiple triangles with two edges known for each triangle, then the final estimated pdf must satisfy the triangle inequality property of all the triangles. We first estimate the pdf of the unknown edge considering each triangle, following which the final pdf is computed by performing the sum-convolution and averaging the convolved pdf (recall Section 3), such that the triangle inequality property is satisfied for all the triangles.

Scenario 2: Another scenario that is likely to occur is when there only exists triangles with two unknown edges. In such cases, both of the unknown edges are jointly estimated, by relying on the known edge.

Solution Considering Scenario 1: As an example, consider Figure 3 and note that based on this greedy selection, at the very first iteration, it will select (i,k) for estimation, as that will complete at least one triangle $\triangle_{i,k,l}$ (because,

the two edges of this triangle are already known and the third edge is to be estimated), whereas none of the other unknown edges will complete any triangle. Considering triangle $\triangle_{i,j,k}$, the algorithm will have to apply the triangle inequality property to select the possible ranges of values that (i, j) is allowed to take.

Our method will estimate the pdf of (i, k) as $Pr((i, k) = 0.25) = 0.0$, $Pr((i, k) = 0.75) = 1.0$ considering $\triangle_{i,j,k}$. After that, (i, k) should also be estimated considering another triangle $\triangle_{i,l,k}$. The final pdf of (i, k) must satisfy the triangle inequality property of both of these triangles.

Algorithm 3 `Tri-Exp`: heuristic distance estimation algorithm

1: **Input:** known and unknown distance edges.
2: **if** There exists triangles with one unknown and two known edges **then**
3: Greedily select that unknown edge and estimate it such that it results in the maximum number of triangles with all known edges
4: **else**
5: When no such triangle is found, consider a triangle and estimate two unknown edges jointly
6: **end if**
7: Perform sum convolution and averaging for all associated triangles such that triangle inequality is satisfied
8: Repeat steps $2 - 7$ until all edges are estimated
9: **return** distance edges

Solution Considering Scenario 2: Consider Figure 3 again and assume that (i, k) is estimated in iteration one. Even after that, both $\triangle_{i,j,l}$ and $\triangle_{j,k,l}$ have two unknown edges.

In $\triangle_{j,k,l}$, where both (k, l) and (j, l) are unknowns and are to be estimated using the pdf of the known edge (j, k). Without further knowledge, we calculate the joint distribution for (j, l) and (k, l) by assigning uniform probability to each of these possible values. Once, we get the joint distribution, we calculate the pdfs for both (j, k) and (j, l) which will be exactly equal to each other, which is $\{0.25 : 0.5, 0.75 : 0.5\}$. As before, when multiple triangles are involved with an unknown edge, the pdf of that edge needs to be estimated considering triangle inequality property of all the involved triangles.

`Tri-Exp` outputs the following pdfs for the example in Figure 3 $(i, k) : [0.25 : 0.5, 0.75 : 0.5], (k, l) : [0.25 : 0.61, 0.75 : 0.39], (j, l) : [0.25 : 0.43, 0.75 : 0.57], (i, l) = [0.25 : 0.4, 0.75 : 0.6]$

Running Time: Time complexity of `Tri-Exp` is $O(|D_u|(n \times \frac{1}{\rho}^2 + log(|D_u|))$, where $|D_u|$ is the number of unknown pairs, ρ is the histogram-width, and n is the number of objects. At worst case, $|D_u| = O(n^2)$; in such cases, the algorithm takes cubic time to run. Nevertheless, this analysis shows that the running time of `Tri-Exp` is substantially superior than its exponential counterparts, `LS-MaxEnt-CG` or `MaxEnt-IPS`.

5. PROBLEM 3: ASKING THE NEXT BEST QUESTION

If there is still considerable "uncertainty" in the learned / estimated distances and we have an opportunity to solicit additional feedback, we investigate (in this third problem) which object pair should we choose to solicit the next feedback on. There are several variants of this problem. In the *online* variant, we have the liberty of asking one question at a time and continue the process until all initially unknown pdfs converges "satisfactorily", or a budget B expires. The budget could be used to specify a limit on the number of questions to be asked, or the maximum number of workers to be involved. In the *offline* variant, we need to decide all questions ahead of time so that the fixed budget expired. In the *hybrid* variant, we could solicit workers feedbacks for several batches of say k questions per iteration. In this paper we mainly focus on the online variant, but also present a simple extension to solve the offline problem.

Modeling Possible Worker feedback: Recall the definition of Problem 3 and note that from a given candidate set of questions D_u (where each question is on an object pair), the problem is to select that question which minimizes the aggregated variance `AggrVar` most. The challenge, however, is to be able to *anticipate* possible workers responses that is currently unknown, to be able to guide the optimization problem. A question $Q(i, j) \in D_u$ is essentially a random variable whose distribution has been estimated already by solving Problem 2. Without any further information, the framework has the following limited options to make guesses about future responses of the workers:

(1) The response pdf from the m workers, when aggregated, will be the same as the current estimated pdf of $d^u(i, j)$. Under this scenario, the framework does not learn anything new about $d(i, j)$ and hence `AggrVar` remains unchanged. We therefore do not use this option in our algorithm.

(2) The aggregated response of the worker will be identical to some measures of the current pdf that dictates its central tendency; for example the *mean* μ of the current pdf can used as the anticipated value of the future aggregated feedback.

In this latter case, the pdf of $d^u(i, j)$ changes (its variance becomes 0), and it is also likely to affect the pdfs of other edges (i.e., the joint distribution changes). More intuitively, when a pdf is represented by its mean, the other pdfs (edges) involved with it are likely to demonstrate lower divergence, hence tighter distribution. As described later, this option is used in our algorithm for selecting the next best question.

Consider a very simple example with 3 objects (i, j, k) that satisfy triangle inequality such that $(i, j) : Pr(d(i, j) = 0.125) = 1$; $(i, k) : Pr(d(i, k) = 0.125) = 0.9, Pr(d(i, k) = 0.375) = 0.1$. To satisfy triangle inequality, the pdf of the third edge (j, k) must be between $[0.0, 0.5]$. However, if we substitute (i, k) with its mean 0.15 (considering it as a candidate question), the pdf of (j, k) becomes tighter and only between $[0, 0.275]$. It is easy to notice that the latter pdf of (j, k) will result in a smaller variance in comparison with the former one.

Algorithm `Next-Best-Tri-Exp`**:** The algorithm for computing the next best question runs in iteration and considers each candidate question $Q(i, j)$ in turn. Then, it considers the impact of changing the current pdf of the object pair to its mean (to emulate workers' feedback). This is done by re-estimating the pdfs in $D_u - d^u(i, j)$. For that, it uses a sub-routine to solve Problem 2, described in Section 4 using any of `LS-MaxEnt-CG`, `MaxEnt-IPS`, or `Tri-Exp` algorithms. Once the unknown pdfs in $\{D_u - d^u(i, j)\}$ are re-estimated, it computes `AggrVar` using either Equation 1 or 2 and maintains the so-far best question by choosing the minimum.

Once all the candidates are evaluated, the best candidate is the one that results in the smallest `AggrVar`. The pseudo-code is presented in Algorithm 4. Using Example 1, this

Algorithm 4 `Next-Best-Tri-Exp`: Selecting the next best question

1: **Input:** known and estimated distance edges.
2: **for** $d^u(i,j) \in D_u$ **do**
3: Replace the distribution of $d^u(i,j)$ by its mean
4: Select $d^u(i,j) = \text{argmax}_{\forall d^u(i,j) \in D_u}$ `AggrVar`$(d^u(i,j))$ as the candidate question
5: **end for**
6: **return** $d^u(i,j)$

returns (i,l) as the next best question, as that minimizes the `AggrVar` based on both formulation of aggregated variance. **Running time:** To choose the next best question, this algorithms has to evaluate each candidate question in D_u. The primary computation time in each candidate question is taken to invoke an algorithm to solve Problem 2 as a subroutine. Therefore, the running time of this algorithm is asymptotically $O(|D_u| \times$ running time of the sub-routine).

Extension to the Offline Problem: If we need to decide how to spend all the budget B ahead of time, we need to decide all the questions offline, we note that the problem becomes computationally more challenging, as there will be an exponential number of possible choices ($\binom{|D_u|}{B}$, assuming the budget allows for B questions) and the ordering of the questions also matters in reducing aggregate variance. However, a simple extension to our current algorithm can effectively solve this offline problem, where we run our online solution B times to select the best B questions greedily. We present experiments on this regard and show that our proposed solution can be effective in solving the offline problem.

6. EXPERIMENTAL EVALUATION

Our development and test environment uses python 2.7 on a Linux Ubuntu 14.04 machine, with Intel core i5 2.3 GHz processor and a 6-GB RAM. All values are calculated as the average of three runs.

6.1 Datasets Description

We use three real world datasets and one synthetic dataset for our experiments. (1) **Image**: The real world dataset is obtained from the PASCAL database[2]. A total of 24 images of 3 different categories are extracted. We generate 3 subsets of size 10, 5, 5 for which we have solicited all pair distance information. Each pair is set up as a HIT (human intelligence task) in Amazon Mechanical Turk (AMT) and we solicit 10 different workers' feedback on the similarity of the images. A total of 50 different workers are involved in this study. (3) **SanFrancisco**: We choose 72 locations from the city of San Francisco and crawl traveling distances (bothways) among all pair of locations (2556 pairs) using google api[3]. The purpose this dataset is to validate the scalability of our algorithms. Here, we use the traveling distances as worker feedback instead of explicitly soliciting the workers' feedback. (2) **Cora**: This is a real world publication dataset of 1838 records, 190 real world entities. We use this dataset to compare our algorithms with Entity Resolution algorithms in [24]. We choose 3 random instances of this dataset with 20 records, which constitutes of 190 edges. We apply our algorithms in these instances and present our results. (4) **Synthetic**: We generate a large scale synthetic dataset for performing efficiency experiments. Here, we vary from 100 to 400 objects which gives rise from 4950 to 79800 object pairs.Additionally, another small synthetic dataset of 5 objects with 10 edges is generated.

[2]http://host.robots.ox.ac.uk/pascal/VOC/databases.html
[3]https://developers.google.com/maps/

6.2 Implemented Algorithms

(1) Worker Feedback Aggregation: We consider the following algorithms:

(i) `Conv-Inp-Aggr`: This is our proposed convolution based solution to aggregate workers feedback that is described in Section 3.

(ii) `BL-Inp-Aggr`: We implement a baseline algorithm that creates aggregated pdf by calculating the average probability over each discrete bucket center of the input pdfs. Here we ignore the ordinal nature of the feedback scale and treat each bucket as a categorical value.

(2) Estimation of Unknown Edges: We are unaware of any related works that study distance estimation in probabilistic settings.

(i) `Tri-Exp`: This algorithm is described in Section 4.2.

(ii) `LS-MaxEnt-CG`: This algorithm is designed to estimate the unknown edges considering both over and under constrained settings, described in section 4.1.1.

(iii) `MaxEnt-IPS`: This algorithm, described in section 4.1.2, refers to the optimal estimation of unknown edges considering only under-constrained settings.

(iv) `BL-Random`: We design a baseline algorithm that is similar to `Tri-Exp`. It estimates the unknown edges considering triangles; however, unlike `Tri-Exp` (which first attempts to consider the edges that complete the highest number of triangles), `BL-Random` arbitrarily chooses unknown edges and estimates them.

(3) Asking the Next Best Question: These algorithms are designed to demonstrate the effectiveness of the next best question in reducing `AggrVar`, as described in Section 5. As `LS-MaxEnt-CG` and `Maxent-IPS` are computationally prohibitive, we implement `Tri-Exp` and `BL-Random` as subroutines to decide the next best questions. We divide these algorithms into two parts - *Online* and *Offline*.

Online Algorithms: Here we solicit one question at a time to the crowd (i) `Next-Best-Tri-Exp`: This is our proposed solution in Section 5 that uses `Tri-Exp` at each iteration as the subroutine to re-estimate the unknown edges. (ii) `Next-Best-BL-Random`: This is again our proposed solution in Section 5 that uses `BL-Random` at each iteration as the subroutine.

Offline Algorithms: Here we solicit a set of questions ahead of time. (i) `Offline-Tri-Exp`: This is the offline variant of `Next-Best-Tri-Exp` described in Section 5.

(4) Entity Resolution(ER): As discussed in Section 7 on related works, under certain circumstances the problem of *entity resolution*, in particular the techniques proposed in [24], may be considered a special case of the distance estimation problem considered in this paper. Consequently, we experiment with the following algorithms:

(i) `Next-Best-Tri-Exp-ER`: This is a modified vesion of `Next-Best-Tri-Exp` algorithm where we find the number of questions that need to be asked so that `Aggr-Var` is zero.

(ii) `Rand-ER` : We implement the *Random* algorithm from [24]. We call this algorithm `Rand-ER`. This algorithm has a proven complexity of $O(nk)$, where n denotes the number of objects and k denotes the number of clusters/similar entities.

6.3 Experimental Set up

Parameter Settings: Unless otherwise mentioned, we assume $\rho = 0.25$. In other words, there are 4 equi-width buckets with bucket range $[0.0 - 0.25)$, $[0.25 - 0.5)$, $[0.5 - 0.75)$, $[0.75 - 1.0)$ with centers at 0.125, 0.375, 0.625 and 0.875. Depending on the value of p (worker correctness), the distribution of the known edges are created. For example, if a worker provides a feedback of 0.8, with $p = 60\%$, that edge is created by assigning probability of 60% on distance 0.875, and the remaining 40% probability is uniformly assigned to the other buckets. In practice, correctness probability can be obtained by asking a set of screening questions and then by averaging their accuracy. The weight of λ is set to 0.5 (unless otherwise stated) for Problem 2.

Quality Experiments:(i) *Worker Feedback Aggregation*: We use real data for this experiment as this dataset contains multiple workers feedback. We consider each triangle in isolation where all the edge distances are known. Hence, for each edge with 10 different feebacks, we know the ground truth distribution. We use `Conv-Inp-Aggr` and `BL-Inp-Aggr` for aggregating two out of the three edges. Based on our respective algorithm, we estimate the third edge. We then compute the ℓ_2 error of our estimated edge from the ground truth distribution for the third edge. (ii) *Unknown Edge Estimation*: Since `LS-MaxEnt-CG`, `MaxEnt-IPS` are exponential in the number of object pairs (i.e., $S^{^nC_2}$), we have to limit our settings to a very small dataset with $n = 5$ nodes and 10 edges. We use the small *Synthetic* dataset, as well as a subset of real world dataset for this experiment. For the *Synthetic* dataset, we consider `MaxEnt-IPS` as the optimal solution, and compare the effectiveness of the other three algorithms by calculating the average ℓ_2 error over the unknown edges, compared to the optimal. Out of the 10 edges, we randomly mark 4 edges as known (and create their distribution as described before), and estimate the remaining 6 unknown edges. For the *Image* dataset, all ground truth distributions are known for the selected 5 objects. Like above, we mark 4 randomly chosen edges to be known and estimate the remaining 6 edges by considering the 4 different algorithms. As before, we present the average ℓ_2 error - but this time in comparison with the ground truth. (iii)*Asking the Next Best Question*: We use the *SanFrancisco* dataset for which we have all pair of ground truth information. At each step, we replace the step of asking a question to the crowd by the ground truth information. The default value of p is 1.0 and the default budget $B = 20$ questions. Number of known edges is is set to 90% of the total edges.

Application to ER: We use *Cora* dataset to perform comparison with ER methods. We assume that each edge is described by a pdf with two ordinal buckets 0 (duplicate) and 1 (not duplicate). We use *number of questions* as our metric which is widely used in ER literature. This value describes the number of questions to be asked before all the entities are resolved. We use 3 random smaller instances of size 20 *Cora* dataset to evaluate our algorithm.

Scalability Experiments: We use the large scale synthetic dataset for the scalability experiments. We vary the following 4 parameters: (i) number of objects n. (ii) number of buckets b' to approximate the pdfs. (iii) number of unknown edges $|D_u|$. (iv) worker correctness p.When one of these aforementioned parameters is varied, the other three are kept constant. The default values for these 4 parameters are, $n = 100$, $|D_u| = 40\%$ of all edges, $b' = 4$, $p = 0.8$. Please note that we primarily present the scalability results for `Tri-Exp` and `BL-Random`, as `LS-MaxEnt-CG` and `MaxEnt-IPS` takes 1.5 days to converge even when $n = 6$.

6.4 Results

6.4.1 Summary of Results

Quality Experiments: Our first experiment on aggregating feedback suggests the superiority of `Conv-Inp-Aggr` over `BL-Inp-Aggr`. For unknown edge estimation, the results indicate that both `Tri-Exp` and `LS-MaxEnt-CG` perform better than the baseline `BL-Random`. For both of them, we observe that with higher worker accuracy (correctness) p, the error increases for all these competing algorithms. While this may appear counter-intuitive, our post-analysis indicates that this is due to the probabilistic nature of our proposed framework and the algorithms, which are most effective, when the workers responses are truly probabilistic. For the third problem, with more questions asked, the `AggrVar` reduces. In both of these aforementioned scenarios, `Next-Best-Tri-Exp` convincingly outperforms `Next-Best-BL-Random`.

Application to ER: Our result demonstrates that `Rand-ER` outperforms `Next-Best-Tri-Exp-ER`. This is expected since our method is designed to solve a more general problem than ER methods - the ER method assumes no worker uncertainty (i.e., workers are always 100% accurate), and it is dependent on the notion of transitive closure, which is a very special case of triangle inequality.

Scalability Experiments: We show that `Tri-Exp` performs reasonably well with the increasing number of objects, buckets, known edges, or worker correctness. The computation time of `BL-Random` is similar to that of `Tri-Exp`, while `Tri-Exp` is qualitatively superior. Therefore, we only present the results of `Tri-Exp` in these experiments. The algorithms that rely on computing joint distribution `LS-MaxEnt-CG`, `MaxEnt-IPS` do not converge beyond a very small number of objects ($n = 5$) even in days.

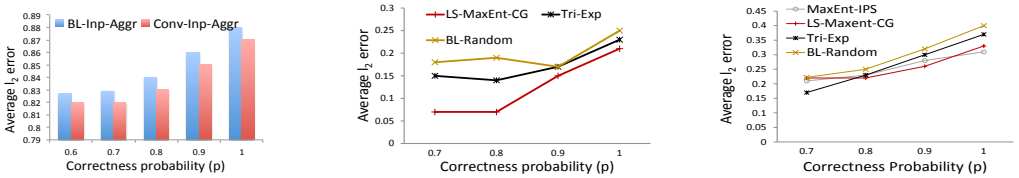
6.4.2 Quality Experiments

(i) *Worker feedback aggregation*: Figure 4(a) shows that `Conv-Inp-Aggr` consistently outperforms the baseline.

(ii) *Estimating Unknown Edges:* We present the results for estimating unknown edges in Figure 4(b) and 4(c). For the synthetic data, `LS-MaxEnt-CG` is superior to the other two methods, while `Tri-Exp` outperforms `BL-Random`. The pattern remains the same for the real data as both `LS-MaxEnt-CG` and `MaxEnt-IPS` exhibit superiority over `BL-Random`. `Tri-Exp` peforms reasonably well for real data. The fact that `LS-MaxEnt-CG` is the best performing algorithm for the real data demonstrates that, in reality, workers may indeed provide inconsistent feedback that do not obey triangle inequality, hence our proposed optimization model is appropriate to capture that settings.

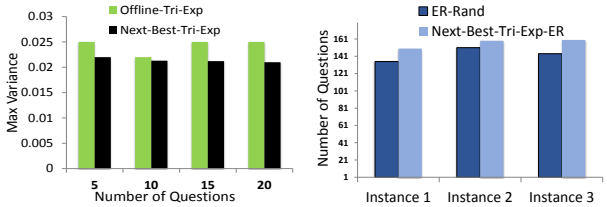
(iii)*Asking the Next Best Question:* We first compare our online algorithms `Next-Best-Tri-Exp` and `Next-Best-BL-Random`.

(a) Varying p: We vary p and present `AggrVar` considering maximum variance. Figure 6(a) presents the results for this



(a) Worker feedback aggregation real data (b) Error vs p synthetic Data (c) Error vs p real Data

Figure 4: **Quality Experiments:** i)Worker Feedback Aggregation ii) Unknown Edge Estimation



(a) `Online vs Offline` (b) `Comparison with ER`

Figure 5: **Experiments for validating Offline algorithms and Entity Resolution**

experiment. While the maximum variance for `Next-Best-BL-Random` and `Next-Best-Tri-Exp` decreases with increasing worker accuracy, latter performs better than the former. For average variance, we encounter the same pattern. Hence, we omit the results for brevity.

(b) Varying B: Our goal here is to test how `AggrVar` reduces with the increasing number of questions (budget B). Figure 6(b) and Figure 6(c) present the outcome of these experiments. It is interesting to observe that with a fairly small number of questions, the `AggrVar` reduces drastically and the system reaches a stable state.

(c) Online vs Offline Experiment: Figure 5(a) presents the result. As expected, `Next-Best-Tri-Exp` performs better than the `Offline-Tri-Exp`, but with very small margin. This result proves that `Offline-Tri-Exp` is very suitable for traditional crowdsourcing framework as online algorithms have high latency.

iv) *Entity Resolution:* Figure 5(b) shows the results for Entity Resolution. Although `Next-Best-Tri-Exp-ER` performs a little worse than `Rand-ER`, we argue that our method is not optimized for finding duplicate entities. Please notice that our method can be applied to find duplicate entities while it is not possible vice versa.

6.4.3 Scalability Experiments

(i) *Worker feedback aggregation:* We observe that the time to aggregate workers feedback is akin to the triangle computation time of `Tri-Exp`. For brevity, we omit the details.
(ii) *Unknown Edge Estimation:* We observe that both heuristic algorithms are equally efficient. Hence, we just present the results of `Tri-Exp`. (a) Varying n: Figure 7(a) presents these results and indicates that `Tri-Exp` converges in a reasonable time, even for higher values of n.
(b) Varying b': Figure 7(b) presents these results and indicates that `Tri-Exp` scales well with increasing b'.
(c) Varying $|D_k|$: Figure 7(c) presents these results and shows that `Tri-Exp` is scalable with increasing number of unknown edges and takes lesser time, as $|D_k|$ increases.
(d) Varying p: Figure 7(d) indicates that the running time of `Tri-Exp` is not affected by p.

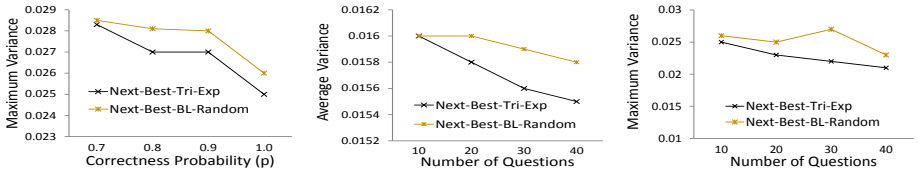
(iii)*Asking the Next Best Question:* The running time of `Next-Best-Tri-Exp` and `Next-Best-BL-Random` are similar and dominated by the size of $|D_u|$. These results are similar to that of Figure 7(c) and omitted for brevity.

7. RELATED WORK

User Input Aggregation: Aggregation of opinions is studied in several prior works in AI [12, 8, 4]. An opinion is described as a pdf over a set of categorical values. Since, their methods do not consider the notion of distance, they do not offer an easy extension to our problem. Aggregation of binary feedback(Yes/No) in crowdsourcing is studied in [7, 14]. Their proposed models estimate both worker accuracy and the true answer considering a bipartite graph of workers and tasks. They do not extend beyond binary feedback while we assume a numeric feedback model. [20] study how to find the ranking of a tuple, where tuple scores are given by probability distributions. While this problem is fundamentally different from our first problem, their proposed approach nevertheless justifies our proposed way of convolving multiple pdfs for aggregation.

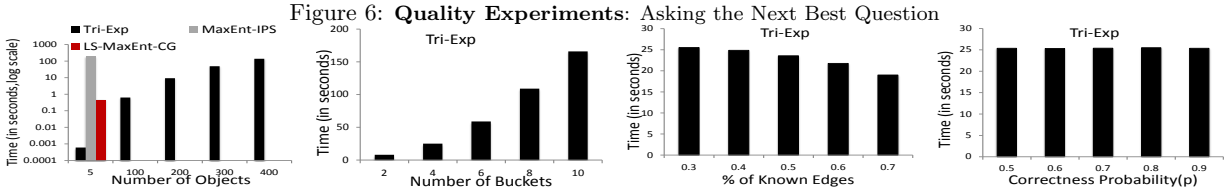
Distance Estimation: *Distance estimation using crowdsourcing* has gained a significant interest recently for solving a variety of computational problems that require distance estimation, such as top-k, clustering, entity resolution (ER), etc [28, 26, 22]. In most of these works, the dependency on distances is only indirect, as these works are based on asking users to resolve Boolean similarity or ranking questions, e.g., whether two objects are similar or not, or whether one object should be ranked higher than the other. In contrast, our work is the first to directly solicit, from the crowd, the broader notion of numeric distances between objects. In [28], the authors propose a crowdsourced clustering method by leveraging matrix completion techniques, where human workers are involved to annotate objects in a deterministic settings. Entity resolution using crowdsourcing have been studied in [25, 26, 24]. The closest related work is that of [24]. The main differences between this work and ours are: (a) the are only concerned with the Boolean notion of objects equivalency, whereas we try to learn numeric distances between objects, (b) they assume that the crowd can make no mistake, which is unrealistic for distance computations, and (c) they leverage the notion of transitive closure, which is a much simpler notion compared to that of triangle inequality. Therefore their main focus has been on determining the optimal set of questions to ask the crowd, whereas our focus has been on even more basic issues such as how to aggregate uncertain user feedbacks and update the probabilistic distribution models of the distances.

Asking Next Best Question: Our third problem formulation borrows motivation from [16, 26, 6]. [16] describes the problem of finding the maximum item from paiwise comparisons, [26] tackles entity resolution, and [6] studies top-k queries in uncertain database. They all designed algorithms



(a) Maximum variance with correctness (b) Average variance with number of questions (c) Maximum variance with number of questions

Figure 6: **Quality Experiments**: Asking the Next Best Question



(a) Time vs varying # of objects (b) Time vs varying # of buckets (c) Time vs varying # of unknown edges (d) Time vs varying worker correctness

Figure 7: **Scalability Experiments**: 4 different parameters are varied. Our default settings is $n = 100$, $p = 0.8$, $|D_u| = 50\%$, $b' = 4$.

for finding the next best question which maximize the expected accuracy for their respective problems. Both [16] and [26] prove that finding next best question is NP-Complete. In [6], authors construct a Tree of Possible Ordering(TPO) in order to find the next best question. Although we employ the similar settings, our unique problem formulation requires us to design novel solutions.

8. CONCLUSION

We present a probabilistic distance estimation framework in crowdsourcing platforms that has wide applicability in different domains. One of the novel contributions of the work is to consider worker feedback with probabilistic interpretation and describe the overall framework with three key components.The effectiveness of our proposed solutions are validated empirically using both real and synthetic data.

Acknowledgment

The work of Habibur Rahman and Gautam Das was supported in part by the Army Research Office under grant W911NF-15-1-0020, and a grant from Microsoft Research.

References

[1] B. Arai et al. Anytime measures for top-k algorithms. In *PVLDB*, 2007.

[2] Å. Björck. Numerical methods for least squares problems. *Pressure Rate Deconvolution Methods for Well Test Analysis*, 1996.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] A. Carvalho and K. Larson. A consensual linear opinion pool. *arXiv preprint arXiv:1204.5399*, 2012.

[5] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 969–984, New York, NY, USA, 2016. ACM.

[6] E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Crowdsourcing for top-k query processing over uncertain data. *Knowledge and Data Engineering, IEEE Transactions on*, 28(1):41–53, 2016.

[7] N. Dalvi et al. Aggregating crowdsourced binary ratings. In *WWW*, pages 285–294, 2013.

[8] F. Dietrich and C. List. Probabilistic opinion pooling. 2014.

[9] R. Fagin and L. Stockmeyer. Relaxing the triangle inequality in pattern matching. *International Journal of Computer Vision*, 30:219–231, 1998.

[10] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[11] R. Fletcher et al. Function minimization by conjugate gradients. *The computer journal*, 1964.

[12] A. Garg, T. Jayram, S. Vaithyanathan, and H. Zhu. Generalized opinion pooling. In *AMAI*, 2004.

[13] D. Gerardi et al. Aggregation of expert opinions. *Games and Economic Behavior*, 2009.

[14] A. Ghosh et al. Who moderates the moderators?: crowdsourcing abuse detection in user-generated content. In *EC*, 2011.

[15] G. H. Golub et al. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 1980.

[16] S. Guo et al. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, 2012.

[17] Y. Ioannidis. The history of histograms (abridged). In *VLDB*, 2003.

[18] C.-W. Ko et al. An exact algorithm for maximum entropy sampling. *Operations Research*, 1995.

[19] F. W. Lawvere. Metric spaces, generalized logic, and closed categories. *Rendiconti del seminario matématico e fisico di Milano*, 1973.

[20] J. Li and A. Deshpande. Ranking continuous probabilistic datasets. *Proceedings of the VLDB Endowment*, 3(1-2):638–649, 2010.

[21] H. Mannila et al. Prediction with local patterns using cross-entropy. In *KDD*, 1999.

[22] V. Polychronopoulos et al. Human-powered top-k lists. In *WebDB*, 2013.

[23] S. Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, 2000.

[24] N. Vesdapunt et al. Crowdsourcing algorithms for entity resolution. *PVLDB*, 2014.

[25] J. Wang et al. Crowder: Crowdsourcing entity resolution. *PVLDB*, 2012.

[26] S. E. Whang et al. Question selection for crowd entity resolution. *PVLDB*, 2013.

[27] C. Xie et al. A maximum entropy-least squares estimator for elastic origin–destination trip matrix estimation. *Transportation Research Part B: Methodological*, 2011.

[28] J. Yi et al. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *NIPS*, pages 1772–1780, 2012.