The Crowd-Median Algorithm

Hannes Heikinheimo
Rovio Entertainment Ltd
hannes.heikinheimo@rovio.com

Antti Ukkonen
Helsinki Institute for Information Technology HIIT
Aalto University
antti.ukkonen@aalto.fi

Abstract

The power of human computation is founded on the capabilities of humans to process qualitative information in a manner that is hard to reproduce with a computer. However, all machine learning algorithms rely on mathematical operations, such as sums, averages, least squares etc. that are less suitable for human computation. This paper is an effort to combine these two aspects of data processing. We consider the problem of computing a *centroid* of a data set, a key component in many data-analysis applications such as clustering, using a very simple human intelligence task (HIT). In this task the workers must choose the outlier from a set of three items. After presenting a number of such triplets to the workers, the item chosen the least number of times as the outlier is selected as the centroid. We provide a proof that the centroid determined by this procedure is equal the mean of a univariate normal distribution. Furthermore, as a demonstration of the viability of our method, we implement a human computation based variant of the k-means clustering algorithm. We present experiments where the proposed method is used to find an "average" image in a collection, and cluster images to semantic categories.

Introduction

Finding a *representative item*, or a *centroid*, from a set $\mathcal{D}$ of items is an important operation in many computational learning and data-analysis tasks. In the simplest case the items are numbers, and we can use any standard measure for central tendency, such as the mean or the median. Moreover, depending on the method used, the chosen centroid may either be some particular $x \in \mathcal{D}$, or it can be an item from the same family as those in $\mathcal{D}$. In general, a common definition of a representative item can be given in terms of a *distance function* d between items in $\mathcal{D}$. The item in $\mathcal{D}$ that is "close" to every other item in $\mathcal{D}$ according to the function d, is in many cases a natural choice as the representative item.

In this paper we study the problem of *finding the centroid of a large set $\mathcal{D}$ of items in a distributed human computation system* using simple human intelligence tasks (HITs). The evident challenge is that all straightforward implementations of the centroid measures listed above require either sorting, nontrivial arithmetic (such as computing sums and divisions), or assessments of the absolute distance between two items. Furthermore, it can be assumed that no explicit numeric feature space exists for our items to reside in. Instead, an implicit distance is induced by the judgements of the human workers. As a solution to these challenges we present a fully parallelized centroid computation method that uses an intuitive and simple HIT as its basic building block. We overcome the challenges listed above by avoiding the usage of absolute measures of distance altogether, hence solving the problem in a robust manner that is well suited for human computation.

Figure 1: Example of a triplet task with images. The middle photograph is different in terms of the content of the images, as the scene it portrays (beach) is different from the two others (mountains).

The solution we propose is based on the following simple task, called a *triplet task*:

> "Out of three shown items pick one that appears to be different from the two others."

Figure 1 illustrates landscape photographs as example items for such a task. Two of the photos show mountains (albeit of different kinds), while the middle photo is a beach scene, clearly deviating by theme from the two mountain pictures.

The resulting algorithm, which we call CROWD-MEDIAN, computes a centroid using such triplet tasks as follows:

1. Pick a large enough number of triplets (subsets of size three) from $\mathcal{D}$.
2. Let human computers solve the corresponding triplet tasks.
3. For each item in $\mathcal{D}$, compute a *penalty score* defined as

the number of times the item was chosen to be "different".

4. Return the item having the lowest penalty score (breaking ties at random).

As we will show empirically in this paper, this algorithm will produce a centroid that is *either the same, or very close to the one that minimizes the sum of distances* to every other item in $\mathcal{D}$. We also provide a formal proof that when the method is applied to the univariate normal distribution with mean μ, the produced centroid is *equal to* μ when all possible triplets are considered.

As a demonstration of the viability of the CROWD-MEDIAN algorithm, we implement the k-means clustering algorithm using human computation. We use two types of HITs: a simple assignment HIT for the k-means' assignment step and the triplet task described above for the update step. To the best of our knowledge, this is the first attempt at implementing k-means in a full-fledged human computation setting.

Summary of contributions

- We define the notion of a *triplet centroid*, and show that it coincides with the mean of any univariate normal distribution.
- We propose the CROWD-MEDIAN algorithm that finds good approximations to the triplet centroid using a simple and easy to understand HIT.
- We show using simulations that the triplet centroid is a good approximation to the point that minimizes the sum of distances to other points.
- We show that the k-means algorithm can be implemented in a human computation setting by building upon CROWD-MEDIAN.

We continue with an overview of related work, and present some design guidelines that we followed when designing the CROWD-MEDIAN algorithm. Then we give a formal definition of the triplet centroid, analyse its properties, and describe the CROWD-MEDIAN algorithm in detail. Finally, we discuss experiments: first we present simulation results, followed by descriptions of two real human computation experiments.

Related Work

We discuss existing work divided to two broad categories, with the second being more related to this paper.

Human computation systems: Research on human computation systems aims at finding general high-level abstractions to provide more standardized interfaces for crowdsourcing. Some approaches, such as CrowdDB (Franklin et al. 2011), and Deco (Park et al. 2012; Parameswaran et al. 2012b), combine relational database technology with human computation. Others provide support for human computation at the programming language level (Minder and Bernstein 2011; Barowy et al. 2012). CrowdForge is a framework for executing complex work (rather than very simple HITs) in a crowdsourcing environment (Kittur et al. 2011). Our work is related to this area of research in the sense that we aim at solving a *simple* and *generic* problem that is not bound to a particular application. Our algorithm could be included as a built-in function to any human computation system.

Human computation algorithms: From an algorithmic point of view the main question in human computation concerns how a particular problem can be phrased in terms of HITs, and how the answers to these are combined to produce the desired solution. Human computation algorithms for elementary problems such as filtering a data set (Parameswaran et al. 2012a), finding the maximum item in a data set (Parameswaran et al. 2012a; Venetis et al. 2012), search in a graph (Parameswaran et al. 2011), enumerating all items that satisfy some criterion (Trushkowsky et al. 2013), and building taxonomies (Chilton et al. 2013) have been considered. This paper falls in this category as well, since our main aim is to design and analyze a novel human computation algorithm for finding the centroid of a data set.

A HIT very similar to our triplet task has also been used by others to find low-dimensional embeddings of a data set (Tamuz et al. 2011; van der Maaten and Weinberger 2012). However, the precise formulation of the HIT as given in those papers differs from ours in a subtle but important way. They fix one of the items as a reference point, and ask the worker to choose from the two remaining items the one that is more similar to the reference item. Our version of the HIT uses no reference item, and the worker is asked to identify one of the items in the triplet as an outlier. Both approaches produce the same kind of output, but from the point of view of the workers the tasks differ slightly.

Finally, a very related algorithmic problem is that of clustering a given set of items to homogenous groups, a fundamental operation in exploratory data-analysis. Algorithms for *crowdclustering* (Gomes et al. 2011; Yi et al. 2012a; 2012b) generally use human computation to assess the similarity between the items being clustered. While our main contribution is about centroid computation, we also apply these results to implement the k-means clustering algorithm using human computation.

Design Principles

A centroid is naturally defined in terms of a distance function, but assessing the absolute distance between two items is very difficult for humans in most cases. We also want to avoid phrasing the task in a way that the answer must be chosen from some arbitrary distance scale. Rather, we formulate the task so that finding the answer involves only *relative judgements* of distance.

In addition to this, our definition aims to satisfy the following technical requirements:

1. *Correct:* The centroid produced by the method should be as close as possible to a centroid given by a standard approach (such as the mean). In general our definition is not in a mathematical sense equal to any standard definition of a centroid. However, we show that it is in fact *equal to the mean* if the underlying distribution is the univariate normal distribution. We also show empirically that for high-dimensional distributions the centroid computed using CROWD-MEDIAN is almost always the same as the one

found by the distance minimization approach.

2. *Scalable:* The complexity of a single HIT should not depend on the size of $\mathcal{D}$. A very simple HIT for determining the centroid would be to show the worker all items in $\mathcal{D}$ and ask to select the one that the user thinks is the most representative. Clearly such a method is not scalable, as the task becomes very difficult when $\mathcal{D}$ contains, say, hundreds or thousands of items. In our approach, the complexity of each HIT (in terms of the number of items shown to the user) is *constant*, that is, completely independent of the size of $\mathcal{D}$.
3. *Parallelizable:* The HITs should be as independent of each other as possible, because this implies that all required HITs can be instantiated at the beginning, and be each handled by a different worker. If an arbitrary large number of HITs can be used, our approach is in this sense fully parallelizable.
4. *Economical:* The number of HITs required to obtain a useful solution should be linear in the size of $\mathcal{D}$, or at most $O(|\mathcal{D}| \log |\mathcal{D}|)$. We show empirically that our method can find good solutions using only a small number of HITs (in relation to the size of $\mathcal{D}$), but we must give up a little in the ability to parallelize the computation.

The above requirements are in no way particular to our problem, but are relevant to any human computation task, and may provide a useful approach to assess human computation algorithms in general.

The Crowd-Median Algorithm

We continue with a formal description of the CROWD-MEDIAN algorithm and its analysis. Let $\mathcal{D}$ denote a set of items. In this paper we mainly assume that $\mathcal{D}$ contains photographs, but our method is applicable to any items that can be easily perceived by humans, such as text, video or sound. Let d be a distance function between items in $\mathcal{D}$. Our objective is to find the $x \in \mathcal{D}$ that is a *centroid* of $\mathcal{D}$ in terms of d, commonly defined as the $x \in \mathcal{D}$ that minimizes the sum

$$D(x) = \sum_{u \in \mathcal{D}} d(x, u). \tag{1}$$

However, in a human computation system we cannot assume that d is defined explicitly. Rather, we say that the function d is *implicitly induced in the minds of the workers*, who can make decisions about the similarity of items in $\mathcal{D}$ on the basis of d, without having to give absolute values of d. Our main technical challenge is thus to approximate Equation 1 using only relative judgments of distance.

The Triplet Centroid

Next we define a *novel notion of a centroid* of a data set $\mathcal{D}$ that is suitable for human computation. Let $\mathcal{T}$ denote the set of all *triplets* of $\mathcal{D}$, that is, $\mathcal{T} = \{T \subset \mathcal{D} \mid |T| = 3\}$. For any $x \in \mathcal{D}$, let $\mathcal{T}_x \subset \mathcal{T}$ denote those triplets to which x belongs to: $\mathcal{T}_x = \{T \in \mathcal{T} \mid x \in T\}$. We say that x is an *outlier* in the triplet $T = (x, u, v) \in \mathcal{T}_x$, whenever the distance from x to both u and v is larger than the distance between u and v. More formally, we define the predicate

$$\Omega(x \mid T) = \mathrm{I}\{d(x, u) > d(u, v)\}\,\mathrm{I}\{d(x, v) > d(u, v)\},$$

where $\mathrm{I}\{\cdot\}$ is the indicator function. Clearly the predicate $\Omega(x \mid T)$ can be evaluated using only relative statements of distances, making it easy for humans to solve.

We continue by showing how a centroid can be defined in terms of $\Omega(x \mid T)$. The *exact triplet score* $S(x)$ of an item $x \in \mathcal{D}$ is equal to the number of triplets in $\mathcal{T}_x$ where the item x is the outlier, defined as

$$S(x) = \sum_{T \in \mathcal{T}_x} \Omega(x \mid T). \tag{2}$$

Definition 1. *The exact* triplet centroid c^t *of a data set* $\mathcal{D}$ *is defined as*

$$c^t = \operatorname{argmin}_{x \in \mathcal{D}} S(x).$$

To find c^t we must compute the exact $S(x)$ for every $x \in \mathcal{D}$. This can be done by considering every $T \in \mathcal{T}$, and for each T finding the x that satisfies $\Omega(x \mid T)$. This is precisely the *triplet task* that we outlined in the introduction:

Definition 2. Triplet task: *Given the triplet* $T \in \mathcal{T}$*, find the* $x \in T$ *that satisfies* $\Omega(x \mid T)$.

Computing the exact triplet centroid requires to solve $\binom{|\mathcal{D}|}{3}$ triplet tasks. This is infeasible in a real human computation system unless $\mathcal{D}$ is very small. Later we show how to find an approximate triplet centroid using only a relatively small number of HITs.

Analysis of the Exact Triplet Centroid

The main result of this section is that for the *univariate normal distribution* the exact triplet centroid coincides with the mean. This makes it a well-founded choice of a centroid.

First, observe that the exact triplet score has a simple probabilistic interpretation. For every $x \in \mathcal{D}$ we have

$$\Pr_{T \sim \mathcal{T}_x}[\Omega(x \mid T) = 1] = \frac{S(x)}{\binom{|\mathcal{D}|-1}{2}},$$

which means that $S(x)$ is proportional to the probability of x being the outlier in a randomly chosen triplet T from $\mathcal{T}_x$. (In the following, we omit the subscript $T \sim \mathcal{T}_x$.) The above equality implies that the $x \in \mathcal{D}$ that minimizes the outlier probability also minimizes $S(x)$. While $S(x)$ is defined in terms of a finite data set $\mathcal{D}$, the probability $\Pr[\Omega(x \mid T) = 1]$ is meaningful for continuous distributions as well.

Next we derive an analytic expression for the probability $q(x)$ of a fixed x being an outlier when u and v are drawn from an arbitrary univariate distribution D over the set of reals $\mathbb{R}$ with density function p. First fix both x and u, and draw v from D. Observe that x is an outlier in the triplet (x, u, v) if v belongs either in the interval $\left[2u - x, \frac{x+u}{2}\right]$ for $x > u$, or $\left[\frac{x+u}{2}, 2u - x\right]$ for $x < u$. Otherwise either u or v will become the outlier. Because v is drawn from D, the probability of x becoming the outlier given a fixed u can be expressed as

$$\Pr_{v \sim D}[\Omega(x \mid (x, u, v)) = 1] = \begin{cases} \int_{i=2u-x}^{\frac{x+u}{2}} p(i)\, di, & x > u, \\ \int_{i=\frac{x+u}{2}}^{2u-x} p(i)\, di, & x < u. \end{cases}$$

By marginalization over u (also drawn from the distribution D) the probability $q(x)$ becomes

$$q(x) = \int_{u=-\infty}^{x} p(u) \int_{i=2u-x}^{\frac{x+u}{2}} p(i)\, di\, du + \int_{u=x}^{\infty} p(u) \int_{i=\frac{x+u}{2}}^{2u-x} p(i)\, di\, du. \quad (3)$$

The above equation holds for any univariate D.

We continue by showing that for distributions that are *symmetric around the median* c the function $q(x)$ has a stationary point at $x = c$. Recall that the median c satisfies $\int_{-\infty}^{c} p(x)\, dx = \frac{1}{2}$.

Lemma 1. *Let p denote a density function with median c so that $p(c-i) = p(c+i)$ for all i. Define $q(x)$ as in Equation 3. We have $q'(c) = 0$.*

Proof. The derivative of q is equal to

$$q'(x) = \int_{u=-\infty}^{x} f(u,x)\, du - \int_{u=x}^{\infty} f(u,x)\, du, \quad (4)$$

where

$$f(u,x) = p(u)\left(\frac{1}{2}p(\frac{x+u}{2}) + p(2u-x)\right). \quad (5)$$

To show that $q'(c) = 0$ it is enough to argue that the integrals in Eq. 4 cancel each other out if $f(u,c)$ is symmetric around c, that is, if $f(c-i,c) = f(c+i,c)$ holds for all i. This is easy to show since p is symmetric around c. □

The existence of a stationary point at c is only a necessary condition for $q(x)$ to have a minimum at $x = c$. Indeed, for certain distributions that are symmetric around c the stationary point at c is in fact a maximum. An example of this is a bimodal distribution with two peaks symmetrically on both sides of c. If the peaks are sufficiently far apart from each other, it can be shown that $q(x)$ has a maximum at $x = c$. A detailed analysis of sufficient conditions that the density function p must satisfy so that $q(x)$ has a minimum at $x = c$ is left as future work. However, next we show that for any univariate *normal distribution* the stationary point of $q(x)$ at $x = c$ is a minimum, and in this common and useful special case the exact triplet centroid is therefore equal to the median (and thereby the mean).

Theorem 1. *Let D be the univariate normal distribution with mean (and median) μ and variance σ^2. The function $q(x)$ as defined in Eq. 3 has a global minimum at $x = \mu$.*

Proof. We substitute

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \quad (6)$$

into Equation 5, and show that when $x < \mu$ we have $q'(x) < 0$, and when $x > \mu$ we have $q'(x) > 0$, where $q'(x)$ is given in Eq. 4. After substituting we obtain

$$q'(x) = -\frac{\sqrt{10}\,\mathrm{erf}(\frac{3\sqrt{10}(\mu-x)}{10\sigma})\exp(\frac{(\mu-x)^2}{10\sigma^2})}{5\sigma\sqrt{\pi}}, \quad (7)$$

where $\mathrm{erf}(\cdot)$ denotes the error function[1], and is defined as

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \exp(-t^2)\, dt. \quad (8)$$

First, observe that besides the leading minus, every term in Equation 7 except $\mathrm{erf}(\frac{3\sqrt{10}(\mu-x)}{10\sigma})$ is always strictly positive. (Clearly the standard deviation σ is always positive.) The sign of $q'(x)$ is thus fully determined by the sign of the error function. Second, observe that for positive arguments the error function is positive, and and for negative arguments the error function is negative. The sign of $\mathrm{erf}(\frac{3\sqrt{10}(\mu-x)}{10\sigma})$ is thus fully determined by the sign of $(\mu-x)$. Indeed, we have thus $q'(x) < 0$ when $x < \mu$, and $q'(x) > 0$ when $x > \mu$. This shows that the stationary point at $x = \mu = c$ must be a unique global minimum. □

Reducing the Number of HITs Needed

In practice computing the exact triplet score as defined in Equation 2 for every $x \in \mathcal{D}$ is going to be only rarely feasible. The number of HITs needed, $\binom{|\mathcal{D}|}{3}$, is simply too much even for a moderately sized $\mathcal{D}$. For example, a relatively small data set of 1000 items results in over 166 million HITs. In practice we can only use a fraction of these.

Recall that $S(x)$ is proportional to the probability of x being the outlier in a randomly chosen T form $\mathcal{T}_x$. Rather than computing this probability exactly, we can compute its sample mean estimator, by sampling triplets with replacement. While this approach is in theory feasible, the accuracy of the sample mean is in practice not enough, given that we can afford only a few triplets for each x.

We propose a heuristic that prunes data points until we are left with a reasonably small subset of $\mathcal{D}$ from which we simply select the solution at random. A more formal analysis of the heuristic is left as future work. In this paper we show experimentally that it can perform very well. The method works as follows:

- Given $\mathcal{D}$, pick a small (of order $O(|\mathcal{D}|)$) sample of triplets, and compute $S(x)$ for every $x \in \mathcal{D}$.
- Identify the median score S_m, and remove all such points x from $\mathcal{D}$ for which $S(x) > S_m$.
- Repeat this recursively on the remaining points in $\mathcal{D}$ for L *pruning cycles*.

Sampling of the triplets in each cycle is done in a special way to guarantee that we have at least some evidence for every $x \in \mathcal{D}$. Specifically, we iterate over every $x \in \mathcal{D}$, and select for each x two other points from $\mathcal{D}$ uniformly at random. After one pass over $\mathcal{D}$ we have thus $|\mathcal{D}|$ triplets. By making H passes, we obtain a sample of $H|\mathcal{D}|$ triplets, with every $x \in \mathcal{D}$ appearing in at least H triplets, and $3H$ triplets in expectation. Now we have all necessary ingredients to present the CROWD-MEDIAN algorithm in Algorithm 1. It takes as input the data set $\mathcal{D}$, as well as two parameters, L and H, that determine how the sampling is to be carried out. Here L is the number of pruning cycles, i.e., the number of times $\mathcal{D}$

[1]The error function is encountered in certain problems in e.g. statistics.

Algorithm 1 CROWD-MEDIAN$(\mathcal{D}, H, L)$

for $l \in 1$ to L **do**
  $\mathcal{T} \leftarrow \emptyset$, $S(x) \leftarrow 0$ for every $x \in \mathcal{D}$
  for $h \in 1$ to H **do**
    for $x \in \mathcal{D}$ **do**
      sample y, z from $\mathcal{D}$ unif. at random
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x, y, z)\}$
  for $(x, y, z) \in \mathcal{T}$ **do**
    $w \leftarrow$ OUTLIER(x, y, z)
    $S(w) \leftarrow S(w) + 1$
  $S_m \leftarrow \text{median}\{S(x) \mid x \in \mathcal{D}\}$
  $\mathcal{D} \leftarrow \{x \in \mathcal{D} \mid S(x) \leq S_m\}$
$S_{\min} \leftarrow \min\{S(x) \mid x \in \mathcal{D}\}$
return a random point from $\{x \in \mathcal{D} \mid S(x) = S_{\min}\}$

is pruned. At its core, however, the algorithm is the same as the one outlined in the Introduction.

Clustering Images

As an example of an application of the CROWD-MEDIAN algorithm we describe a simple system for clustering images. Unlike existing approaches to crowdclustering, we propose to implement the k-means algorithm on a human computation system.

Implementing k-means

The standard k-means algorithm (Lloyd 1982) requires the following two operations:

1. ASSIGN: Given a set of items $\mathcal{C} \subset \mathcal{D}$ and an additional item $x \in \mathcal{D}$, find the item $c \in \mathcal{C}$ that is closest to x according to the distance function d.
2. UPDATE: Given the set $C \subseteq \mathcal{D}$, find the "center" of C, that is, the item $x \in C$ that minimizes $\sum_{c \in C} d(x, c)$.

The algorithm starts from an initial (randomly chosen) set of centers, and alternates between the two operations until convergence. Observe that we do not need a direct mechanism to evaluate the function d provided we can carry out the above operations through some other means. The ASSIGN operation is straightforward to frame as a HIT: we simply show the worker all images in the set C, as well as the image $x \in \mathcal{D}$, and ask her to pick the image in C that resembles x the most. The UPDATE step can be implemented in an approximate manner with the CROWD-MEDIAN algorithm.

A downside of the k-means algorithm is that the number of clusters must be specified in advance, and usually the cluster centers are initialized at random. We implemented a simple initialization method for k-means that uses the triplet task. This initialization phase finds initial cluster centers, and is completely *parameter free*, i.e., k is not an input to the algorithm. The method is as follows: we scan over the data $\mathcal{D}$, maintain a set of current cluster centers, and create new triplet tasks on the fly where every $x \in \mathcal{D}$ is compared with the current set of cluster centers. If x is an outlier with respect to every cluster center found so far, then x is added to the set of cluster centers.

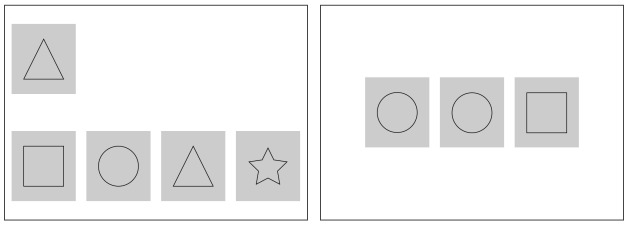

Figure 2: Schematic depictions of the user interfaces for the assignment (left, $k = 4$) and triplet tasks (right) in the image clustering system used in our experiments. The worker must in both cases simply click on one of the photographs, upon which the system immediately retrieves a new task.

We implemented the k-means and the initialization method in a system for clustering images. This system consists of a server that provides a simple HTTP interface for requesting HITs, submitting results, and performing a number of maintenance operations. The client is a lightweight HTML5 application that runs in all modern web browsers. The server maintains a pool of tasks and sends the next available task to the client upon request. After a task is completed, the client submits the result back to the server and requests a new task, which is shown immediately shown to the worker. The complete system is available as open-source[2].

The system operates in three different phases: the *initialization phase*, the *assignment phase*, and the *update phase*. The assignment and update phases correspond to the two steps of the k-means algorithm. The update phase is implemented with the CROWD-MEDIAN algorithm, while the assignment phase is based on a HIT where the worker is shown a image in the top left corner, with a number of other images laid out below it. The task is to click on the image in the bottom row that is most similar to the image above. The assignment phase consists of $|\mathcal{D}|$ such HITs, one for every $x \in \mathcal{D}$. Schematic depictions of the user-interface for the assignment and triplet tasks are shown in Figure 2.

Experiments

Simulating Crowd-Median on Artificial Data

Accuracy of the Exact Method We first provide some empirical evidence that the exact triplet centroid c^t indeed is a very good approximation of the distance sum minimizing centroid c^d also for high dimensional data. To this end we ran a simulation where 100 samples of 100 points were drawn from a multivariate Gaussian of m dimensions, for different values of m. For each point x in each sample we computed both the distance sum $D(x)$ (Eq. 1), as well as the exact triplet score $S(x)$ (Eq. 2).

Table 1 shows the fraction of the 100 samples in which the exact triplet centroid c^t was equal to the distance centroid c^d. We observe that this fraction seems to be increasing in m, suggesting that the approach is also applicable when the data space is high-dimensional.

[2] http://anttiukkonen.com/hcomp

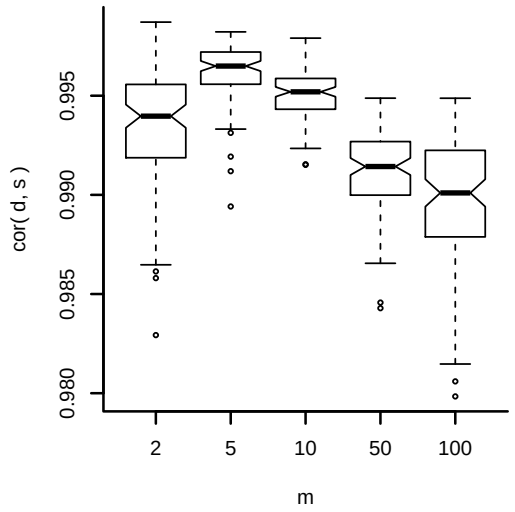

Figure 3: Correlation between distance sum D and the exact triplet score S for different dimensionalities m of the data.

Table 1: Fraction of cases where the triplet centroid c^t coincides with the distance centroid c^d

data dimensionality	2	5	10	50	100
fraction correct	0.60	0.81	0.89	0.86	0.93
rand. success prob.	0.03	0.05	0.19	0.93	1.00

Table 1 also shows the probability that a random point drawn from the distribution is equally close to the distance centroid c^d as is the triplet centroid c^t. (The reported numbers are medians from the 100 samples.) More formally, this probability is defined as $\Pr[X \leq \|c^t - c^d\|^2]$, where X follows the χ^2 distribution with m degrees of freedom. The random success probability tells us that in low dimensions c^t is very close to c^d. As m increases the pairwise distances become more concentrated, and almost all points are roughly at the same distance from c^d as the exact triplet centroid c^t.

It is therefore also of interest to understand how well the exact triplet score S agrees with the distance sum D for all data points $x \in \mathcal{D}$, in particular as m increases. Figure 3 shows the correlation between $D(x)$ (Eq. 1) and $S(x)$ (Eq. 2) in random samples of 100 points from a standard m-dimensional Gaussian distribution. We observe that the correlation decreases as m increases. However, the variation is in practice negligible, and in absolute terms the correlations are very high, over 0.98 in almost every case. This means that the triplet score is also useful for finding points that are close to c^d, as well as locating outliers.

Accuracy vs. Cost Trade-off Next we study how well the heuristic we introduced for reducing the number of triplets works in a simulated setting. The purpose of this experiment is to find out what parameters of CROWD-MEDIAN lead to a good trade-off between the number of HITs used and the quality of the centroid found as compared to the true distance centroid. As data we use samples of 500 points drawn from a standard bivariate normal distribution. Note that computing the exact $S(x)$ for every $x \in \mathcal{D}$ would require over 20 million HITs in this case, which is clearly too much for practical purposes. Like above, we measure error as the probability of a point drawn randomly from the underlying distribution being at least as close to the true centroid c^d as the chosen point c^t. For simplicity we do not consider replicating the same HIT across a number of workers.

From the leftmost plot in Figure 4 we can see that the error is decreasing both in the number of passes H as well as the number of pruning cycles L. However, of these the number of cycles has a stronger effect. Apart from some noise, the curves corresponding to using l pruning cycles is below the curve for $l-1$ cycles. The middle plot in Figure 4 shows the number of HITs as a function of H. We can observe that when using more than 3 cycles the cost of adding one more pass exceed the costs of adding a new pruning cycle. Finally, in the rightmost plot of Figure 4 we show the error as a function of the number of HITs used. For each pruning cycle, the leftmost symbol corresponds to using a single pass, while the rightmost symbol indicates the performance with 10 passes. We have highlighted the symbols for $L = 6$ in bold. The optimum performance has an error somewhat below 0.1, and this is reached already when using 6 cycles with 3 passes on each, resulting in about 4000 HITs for a dataset of 500 points. This is four orders of magnitude less than what computing the exact triplet scores requires. On the other hand, as a comparison with Table 1 (for $m = 2$) reveals, the error we obtain using the heuristic is only roughly twice that of the exact method. We consider such errors to be perfectly acceptable for practical purposes.

Crowd-Median with Image Data

We ran simple human computation experiments using our image clustering system. In this paper we describe two of those. The objective of the experiments is twofold. On the one hand we want to demonstrate that the CROWD-MEDIAN algorithm produces intuitive centroids, as well as study human performance on the triplet task. On the other hand we want to show that our implementation of the k-means algorithm finds thematically coherent clusters in a real human computation setting.

Datasets Our first data set (NATURE) is a collection of 120 images with four thematic clusters (*forest*, *open country*, *coast*, *mountain*) that consists of randomly chosen photographs from the corresponding categories of the Scene data set[3] (Oliva and Torralba 2001). These categories were chosen because the photographs they contain are all of natural subjects, mainly landscape shots of different types, and are all rather easy for humans to distinguish. (We also experimented with the "urban" categories *highway*, *street*, *city center*, and *tall building* of the same image collection, and obtained very similar results.)

The second dataset (VOGUE) is a set of 60 cover images of the UK edition of the Vogue magazine, chosen at random from the time interval 1970 to 2012. URLs to the images

[3] http://cvcl.mit.edu/database.htm

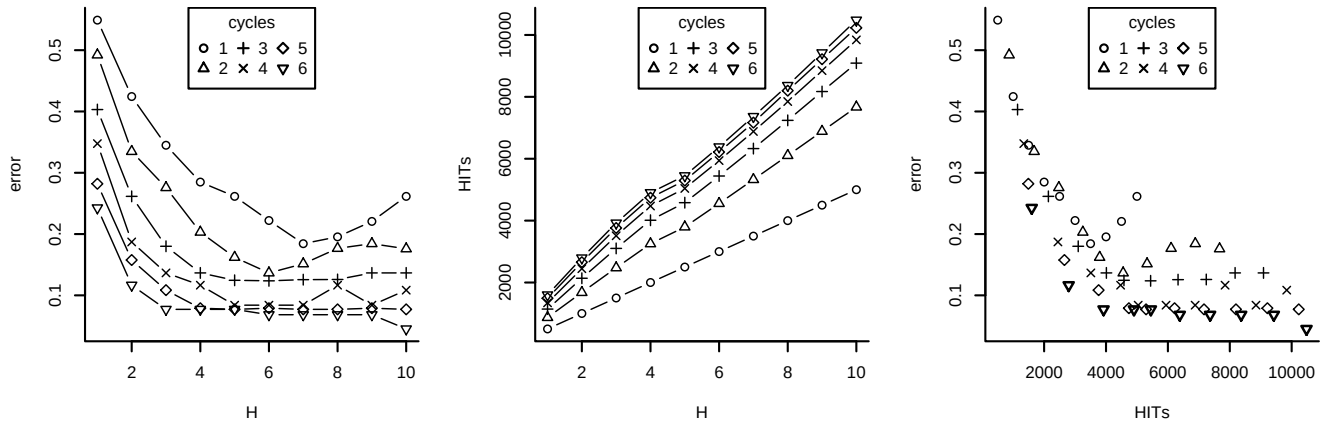

Figure 4: Simulation study of the triplet centroid: Error and number of HITs as a function of h, as well as number of HITs vs. error.

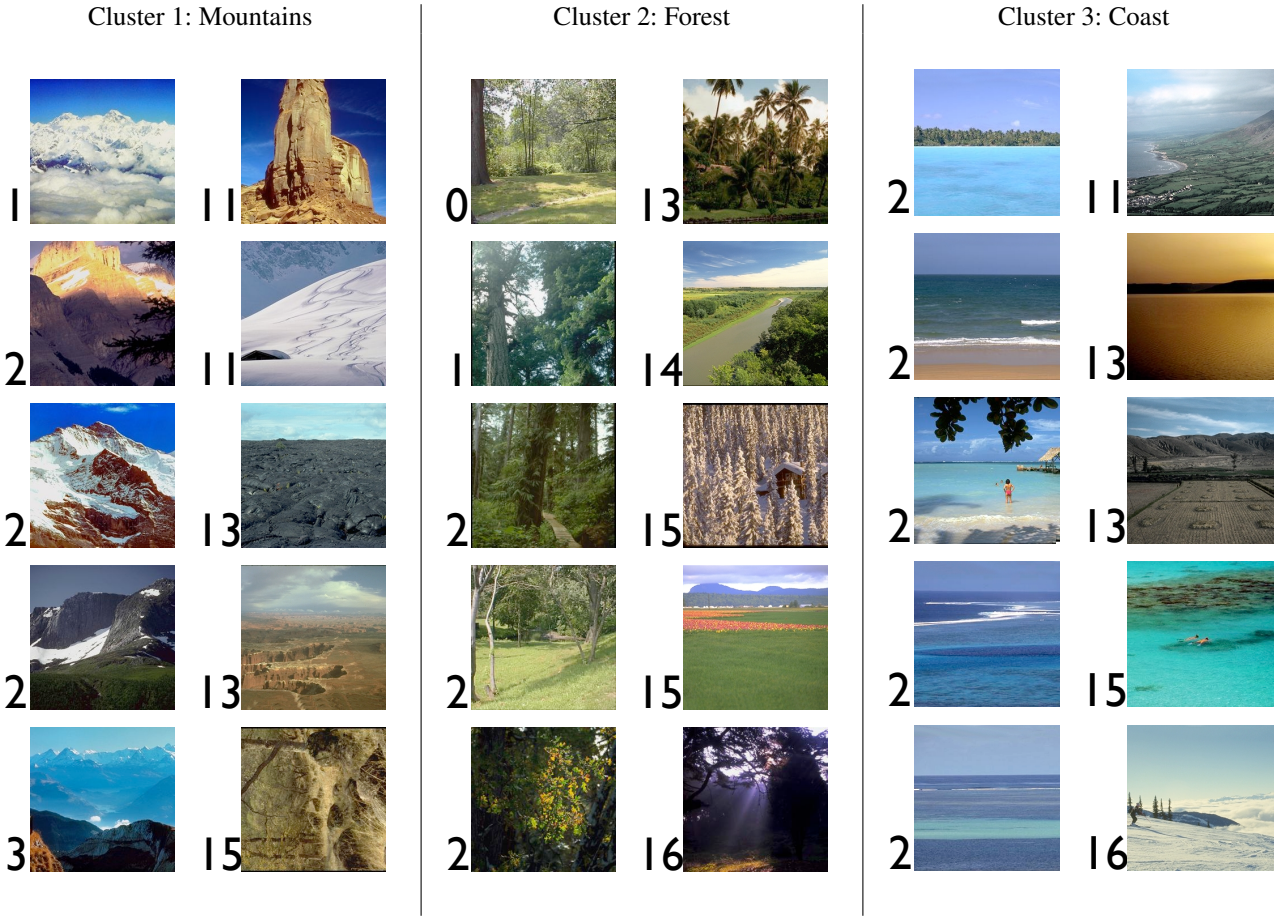

Figure 5: Results of a human computation experiment: Images having the lowest and highest triplet scores in three clusters we found in the NATURE data set (score shown next to the photographs). In all cases the images having a low score are thematically homogenous, while the high scoring photos are diverse, and contain thematical outliers.

were crawled from the Vogue website[4].

Experiment setup We recruited six workers (3 male, 3 female) who volunteered to participate in the experiment. Both experiments reported here were carried out in a single one hour session with all workers simultaneously present in the same space, working together on the same job. The workers were given a brief introduction to human computation, as well as written instructions of the two HITs. The test subjects were not rewarded in any way. In both experiments we employed a simple version of the CROWD-MEDIAN algorithm that only uses one pruning cycle with five passes, i.e., we set $L = 1$ and $H = 5$ (see Alg. 1). Every HIT was replicated three times, and the outcome was determined by majority vote, breaking ties at random.

Experiment 1: Clustering landscape images We ran the initialization phase, as well as one assignment and one update phase of our k-means algorithm on the NATURE data. The workers were not instructed to carry out the triplet task according to any specific criteria, but to simply go by their intuition.

Completing all 3812 HITs took 29 minutes and 35 seconds. The fastest worker submitted 871 HITs, while the slowest completed 381. The number of HITs, as well as the median completion time (in parenthesis) of the HITs in the initialization, assignment, and update phases were, 1220 (1.8 sec), 370 (1.4 sec), and 2222 (2.2 sec), respectively. Clearly the assignment task was in general easier than the triplet task. According to the Wilcoxon rank sum test the completion times of the triplet tasks in the initialization and update phases differ significantly (p-value $< 2.2 \times 10^{-16}$). This suggests that the repetitive nature of the triplet task in the initialization phase leads to slightly faster processing, albeit the test subjects reported this part as being rather boring.

The initialization phase discovered three centroids that correspond to the *forest*, *mountain*, and *coast* categories. This makes sense, as the *open country* images are more difficult to distinguish. They are easily confused with one of the other main categories of the NATURE data set. We consider this an excellent result, given that the workers were not given any instructions specific to the images being clustered, and that initialization was completely automatic. Figure 5 shows the clusters as well as 10 images having the lowest and highest triplet score after the update phase. In every case the images having a low score are thematically homogenous, suggesting that the CROWD-MEDIAN algorithm indeed identified images that are representative of the clusters. Moreover, images with a high score are diverse, and contain some examples of the *open country* category.

Experiment 2: Finding an "average" magazine cover The purpose of this experiment was to study if we can use human computation to identify a centroid in terms of "style" or "fashion". For modern computer vision algorithms, distinguishing an '80s look from a contemporary style is extremely difficult, if not impossible. This task, while by no means trivial, is still perfectly doable for humans. The workers were instructed to take aspects of "style" into account when solving the triplet task.

[4] http://www.vogue.co.uk/magazine/archive/

Running CROWD-MEDIAN with the VOGUE data took 12 minutes and 11 seconds. The total number of completed triplet tasks was 1131, with a median completion time of 2.6 seconds per HIT, which is slightly slower than with the NATURE data set. The fastest and slowest worker completed 328 and 105 tasks, with median completion times of 1.7 and 4.8 seconds per HIT, respectively.

We observe that the CROWD-MEDIAN algorithm gave a low triplet score to pictures that are examples of stereotypical women's magazine covers: a simple headshot of a female model. However, other stylistic attributes, such as hairstyle or clothing of the models did not affect the result. The lowest triplet score was awarded to the April 2011 cover[5] portraying Kate Winslet. In comparison, covers with a high triplet score feature either more than one model, or are otherwise different in their composition. Not surprisingly, the December 1999 cover[6] that has no image at all had the highest triplet score.

Conclusion and Future Work

We have proposed the notion of a *triplet centroid*, as well as the CROWD-MEDIAN algorithm for finding this type of centroid from a given data set $\mathcal{D}$ using human computation. We consider the CROWD-MEDIAN algorithm to be an interesting novel contribution to human computation. Unlike many existing human computation algorithms that process data sets, CROWD-MEDIAN computes a nontrivial aggregate of $\mathcal{D}$, rather than applying some function independently to every $x \in \mathcal{D}$. We provided both theoretical and empirical evidence that the triplet centroid coincides with existing centroid definitions. We also showed how the k-means clustering algorithm can be implemented using the CROWD-MEDIAN algorithm in the update step.

As future work, it is of interest to use CROWD-MEDIAN to implement other algorithms that require centroid computations, such as time series segmentation, or naive Bayes models. Also, instead of selecting a single centroid, finding a collection of representative items is also easily accomplished using the triplet score. This might have some advantages in practical applications when identifying a unique centroid is difficult. Finally, a more thorough analysis of the exact triplet score (Eq. 3) is of interest for a better theoretical understanding of the proposed algorithm.

Acknowledgements The authors wish to thank all volunteers of the human computation experiments, the anonymous reviewers for their excellent suggestions, as well as Esko Ukkonen for his insightful comments about Theorem 1.

References

Barowy, D. W.; Curtsinger, C.; Berger, E. D.; and McGregor, A. 2012. Automan: a platform for integrating human-based and digital computation. In *OOPSLA*, 639–654.

[5] http://www.vogue.co.uk/magazine/archive/issue/2011/April

[6] http://www.vogue.co.uk/magazine/archive/issue/1999/December

Chilton, L. B.; Little, G.; Edge, D.; Weld, D. S.; and Landay, J. A. 2013. Cascade: crowdsourcing taxonomy creation. In *CHI 2013*, 1999–2008.

Franklin, M. J.; Kossmann, D.; Kraska, T.; Ramesh, S.; and Xin, R. 2011. Crowddb: answering queries with crowdsourcing. In *SIGMOD 2011*, 61–72.

Gomes, R.; Welinder, P.; Krause, A.; and Perona, P. 2011. Crowdclustering. In *NIPS 2011*, 558–566.

Kittur, A.; Smus, B.; Khamkar, S.; and Kraut, R. E. 2011. Crowdforge: Crowdsourcing complex work. In *UIST 2011*, 43–52.

Lloyd, S. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory* 28(2):129–137.

Minder, P., and Bernstein, A. 2011. Crowdlang-first steps towards programmable human computers for general computation. In *HCOMP 2011*.

Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision* 42(3):145–175.

Parameswaran, A. G.; Sarma, A. D.; Garcia-Molina, H.; Polyzotis, N.; and Widom, J. 2011. Human-assisted graph search: it's okay to ask questions. *PVLDB* 4(5):267–278.

Parameswaran, A. G.; Garcia-Molina, H.; Park, H.; Polyzotis, N.; Ramesh, A.; and Widom, J. 2012a. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD 2012*, 361–372.

Parameswaran, A. G.; Park, H.; Garcia-Molina, H.; Polyzotis, N.; and Widom, J. 2012b. Deco: declarative crowdsourcing. In *CIKM 2012*, 1203–1212.

Park, H.; Pang, R.; Parameswaran, A. G.; Garcia-Molina, H.; Polyzotis, N.; and Widom, J. 2012. Deco: A system for declarative crowdsourcing. *PVLDB* 5(12):1990–1993.

Tamuz, O.; Liu, C.; Belongie, S.; Shamir, O.; and Kalai, A. 2011. Adaptively learning the crowd kernel. In *ICML 2011*, 673–680.

Trushkowsky, B.; Kraska, T.; Franklin, M. J.; and Sarkar, P. 2013. Crowdsourced enumeration queries. In *ICDE 2013*, 673–684.

van der Maaten, L., and Weinberger, K. 2012. Stochastic triplet embedding. In *MLSP 2012*, 1–6.

Venetis, P.; Garcia-Molina, H.; Huang, K.; and Polyzotis, N. 2012. Max algorithms in crowdsourcing environments. In *WWW 2012*, 989–998.

Yi, J.; Jin, R.; Jain, A.; Jain, S.; and Yang, T. 2012a. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *NIPS 2012*, 1781–1789.

Yi, J.; Jin, R.; Jain, A. K.; and Jain, S. 2012b. Crowdclustering with sparse pairwise labels: A matrix completion approach. In *HCOMP 2012*.