Int J Comput Vis (2014) 108:133–147
DOI 10.1007/s11263-013-0680-6

Active Image Clustering with Pairwise Constraints from Humans

Arijit Biswas · David Jacobs

Received: 25 February 2013 / Accepted: 23 November 2013 / Published online: 10 December 2013

Abstract We propose a method of clustering images that combines algorithmic and human input. An algorithm provides us with pairwise image similarities. We then actively obtain selected, more accurate pairwise similarities from humans. A novel method is developed to choose the most useful pairs to show a person, obtaining constraints that improve clustering. In a clustering assignment, elements in each data pair are either in the same cluster or in different clusters. We simulate inverting these pairwise relations and see how that affects the overall clustering. We choose a pair that maximizes the expected change in the clustering. The proposed algorithm has high time complexity, so we also propose a version of this algorithm that is much faster and exactly replicates our original algorithm. We further improve run-time by adding two heuristics, and show that these do not significantly impact the effectiveness of our method. We have run experiments in three different domains, namely leaf, face and scene images, and show that the proposed method improves clustering performance significantly.

Keywords Clustering · Active learning · Human in the loop · Pairwise constraints · Image labeling

1 Introduction

Clustering, or *unsupervised learning*, is a critical part of the analysis of data. There has been a huge volume of work on clustering (Jain 2010), producing many interesting and effective algorithms. However, all clustering algorithms depend on some method of computing a distance between items to be clustered that reflects their similarity. For most tasks, automatically computed distances provide useful information about similarity, but still produce significant errors. This leads even the best clustering algorithms to produce clusters that do not contain objects from the same class.

We therefore propose a new clustering method that brings a human into the loop. In many tasks, experts, or even naive humans, can provide very accurate answers to the question of whether two objects belong in the same cluster. In spite of this accuracy, it is not practical to expect people to cluster thousands of objects into meaningful groups. Our goal, therefore is to meld human and automatic resources by directing valuable human attention to those judgments that are most critical to improving clusterings produced by automatic means.

We illustrate the value of this approach with examples of clustering in surveillance videos and plant images.

– There are many applications for clustering faces or actions in surveillance videos. This could allow, for example, an analyst to determine whether the same person has visited a number of locations, or to find different people who have performed similar actions. Images from videos have variations in pose, illumination and resolution that make automatic analysis extremely challenging, so that automatic clustering will be quite error-prone. But a person can readily look at two face images or actions and tell if they are similar.
– There has been a great deal of interest recently in obtaining large, labeled image sets for plant species identification (Kumar et al. 2012). Classifiers that can identify species require large sets of leaf images, labeled with their species. Accurately labeling such images requires

A. Biswas (✉) · D. Jacobs
Computer Science Department, University of Maryland,
College Park, MD 20742, USA
e-mail: arijitbiswas87@gmail.com

D. Jacobs
e-mail: djacobs@umiacs.umd.edu

experience and botanical knowledge. One approach that can reduce this effort is to cluster all the images into groups such that each come from a single species, and then have botanists label each group. Initial clustering can be performed using generic algorithms that measure the similarity of two leaves, but this clustering will be quite noisy, because such algorithms are still imperfect. At the same time, we observe that even an untrained person can compare two leaf images and provide an accurate assessment of their similarity.

We may then summarize the clustering problem we have solved as having the following characteristics:

– We begin with a collection of objects that can be grouped into a set of disjoint clusters.
– We have an automatic algorithm that can give us some useful information about the similarity of two objects.
– A person can make these judgments with much greater accuracy than existing algorithms.

Our proposed approach can be applied to clustering problems, where it is possible to define the level of partitioning explicitly, e.g: faces, leaves, moths, birds, cars and any clustering problem for fine grained classification. In these examples, we can ask questions like "do these pair of images belong to the same person or not" for faces or "do these pair of leaf images belong to the same species or not" for leaves while getting human feedback.

We also assume that a person never provides incorrect constraints and that we know the number of clusters beforehand. In practice, humans can be highly accurate at image comparison for some tasks. For example, (Kumar et al. 2009) shows that humans achieve 99.2 % accuracy in a face comparison task, even without the option of responding "don't know". We expect human input to vary in reliability a lot depending on the task and our method is only designed to handle cases where human input is pretty reliable. We also note that most of the previous active clustering algorithms have assumed that humans are always correct in judging the similarity of two images (Basu et al. 2004; Mallapragada et al. 2008; Wang and Davidson 2010; Xu et al. 2005; Xiong et al. 2012; Huang et al. 2007; Huang and Lam 2009). Like much work in clustering and all prior work on active clustering, we focus on the problem of forming clusters. Many approaches have been developed for determining the number of clusters (Sugar and James 2003) but this is outside the scope of our current work.

Given this problem formulation, we have proposed an algorithm that does the following:

1. Cluster objects into groups, combining automatically computed distances with any constraints provided by people.
2. Choose a useful question to ask a person. The person will compare two objects and indicate whether they belong to the same group (or answer "don't know").
3. Repeat, using the information provided by the person as a new constraint.
4. Continue asking questions until a reasonable clustering is achieved or the human budget is exhausted.

We show the pipeline of our algorithm in Fig. 1 (face images from (Martinez and Benavente 1998)).

The main innovation of our work is a new algorithm for Step 2. We simulate the potential effects of all possible questions, and select the question that will cause the largest effective change in the current clustering. Computing this requires two ingredients. First, we develop a method for estimating the probability distribution of a human's answer to each question, and show empirically that this distribution is accurate. Second, we develop efficient methods for simulating the effects of a large number of questions. In part, we are able to do this by using a very simple clustering algorithm; although this leads to suboptimal initial clustering, we demonstrate that this choice quickly pays off by allowing us to select useful questions. And we develop efficient algorithms and useful heuristics that allow us to perform incremental clustering quickly.

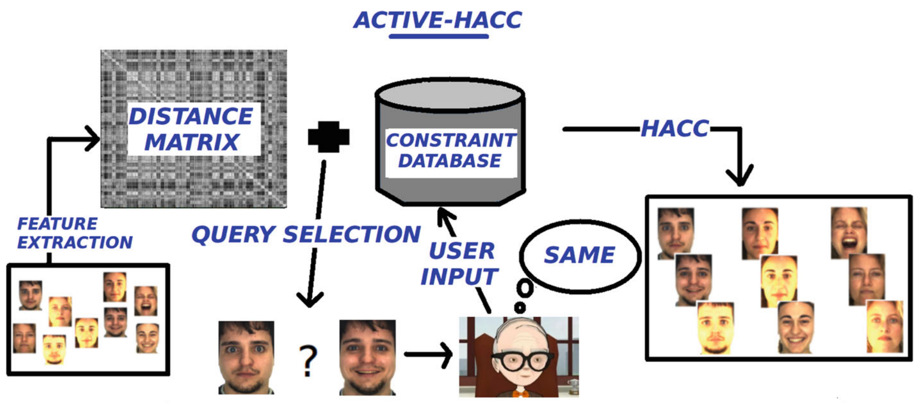


Fig. 1 Pipeline of our system: Active-HACC is our proposed active algorithm for selecting data-pairs to get constraints and HACC is a constrained clustering algorithm

We perform experiments to evaluate our algorithm in three different domains: face, leaf and scene images. Since we assume that people are highly accurate, in experiments we can simulate their behavior using ground truth. However we also demonstrate some experiments with real human input in Sect. 4.2. We note that a preliminary version of this work appeared in (Biswas and Jacobs 2012). We point out the major extensions to that work:

– We relax the assumption that humans are always correct and perform experiments with real humans in the loop using Mechanical Turk. We find that our algorithm still produces useful results and is better than the state-of-the-art approaches.
– We also compare our approach with a recent active learning algorithm for clustering (Xiong et al. 2012) and a semi-supervised clustering algorithm (Wagstaff et al. 2001) with pairwise constraints.
– We run experiments for scene image clustering and demonstrate that our algorithm is better than the state-of-the-art active clustering algorithms. Features extracted from scene images are usually less useful than features from leaves or faces, making automatic clustering of scene images extremely hard. We find that we need more constraints from humans to cluster scene images than a similar sized leaf or face image dataset.
– We expand the related work section to provide a more detailed literature review in this area.

2 Related Work

Recently computer vision researchers have been interested in melding active learning (Angluin 1987; Settles 2010) with computer vision. We will discuss a few recent works in this area. In (Vijayanarasimhan et al. 2010), the authors propose a method for active selection of a batch of images for labeling using a SVM classifier where the total cost for labeling at each iteration is constrained by a predefined budget. The proposed method is shown to be useful for object recognition, activity recognition and image retrieval. The authors in (Vijayanarasimhan and Grauman 2011) propose a method of training a large scale object detector in an active manner. The proposed algorithm iteratively poses annotation requests to humans and gathers relevant images for labeling from Flickr, where the true labels of the data are not known beforehand. In (Vijayanarasimhan and Grauman 2012), the authors propose a method for active frame selection for labeling such that a video sequence is labeled (at the pixel level) with as little human input as possible. They propose two methods, one is a basic flow-based approach while the other is a more sophisticated approach that uses a flow model within a Markov Random Field (Chellappa and Jain 1993). (Branson et al. 2010) propose an interactive human-in-the-loop method for bird classification. Humans will answer simple questions on visual attributes of birds such that a bird can be classified with limited human interaction. In (Branson et al. 2011), the authors propose a method for large scale interactive learning of deformable part models (Felzenszwalb et al. 2008). The authors in (Siddiquie and Gupta 2010) propose an active technique for learning appearance and contextual models for scene understanding simultaneously. This is the first multi-class active learning technique that takes contextual interactions between different parts of a scene image into account.

The authors in (Guo and Greiner 2007) proposed an active learning method for labeling unlabeled data given a set of labeled instances. They use an optimistic information theoretic way to use the unlabeled data; that is to find the instance whose best possible label assignment could maximize the mutual information about the labels of the remaining unlabeled instances. Also in (Dagli et al. 2006), the authors propose an active learning method, which uses an information-theoretic diversity measure, to label samples from a large collection of unlabeled images. In (Kapoor et al. 2007) the authors propose a method for active learning that uses expected value of information to label images. However all of these approaches want to get labels of images and do not try to partition the dataset as we do. Note that we are only computing the expected change in clustering, which is different from pure information theoretic approaches, because it is not clear what is the best possible way to quantify information gain for active learning in clustering. However our approach is inspired by information theoretic approaches for active learning in general. In (Holub et al. 2008) the authors propose an active learning technique for object recognition, which aims to minimize the entropy of the current unlabeled set of images given a labeled set. The authors in (Joshi et al. 2010) proposed the first multi-class active learning technique that requires binary (yes or no) user inputs. This method points out the difficulty of providing category labels for large multi-class problems and suggests using pairwise similarity questions for labeling with access to an already available labeled set. In a clustering problem we usually do not have any labeled data available for selection of these pairwise questions. In (Jain and Kapoor 2009) the authors proposed an active learning scheme for large multi-class problems where the goal is to select the data point that strengthens the existing classification model the most in terms of its discriminative capabilities.

Clustering has been an interesting topic of research in machine learning, computer vision and natural language processing for more than 50 years. We refer to (Jain 2010) for a nice and comprehensive review. Combining active learning with constrained clustering (Basu et al. 2008) has long been an area of interest for machine learning researchers.

In (Basu et al. 2004), the authors propose an active clustering algorithm. They suggest two major phases in an active learning setting namely "Explore" (cluster center initialization) and "Consolidate" (data points are added to cluster centers). In problems with a large number of clusters (which is very common in the image clustering domain), the "Explore" stage itself takes a large number of questions to initialize the distinct cluster centers. The authors in (Mallapragada et al. 2008) have proposed an approach that uses a min-max criterion to find informative questions. They also rely on the "Explore" stage in the beginning as (Basu et al. 2004) does. There are also a couple of active clustering algorithms (Wang and Davidson 2010; Xu et al. 2005) based on spectral eigenvectors, but they are good for two-cluster problems only. Very recently the authors in (Xiong et al. 2012) proposed an active spectral clustering algorithm with k-nearest neighbor graphs that can be used for more than two-cluster problems. We compare with their approach and show that our proposed approach is much better. In (Huang et al. 2007), the authors have proposed an active framework for constrained document clustering. This paper is philosophically similar to our approach, i.e., they also try to ask questions to maximize the gain. They begin with a skeleton structure of neighborhoods covering all the clusters. They then search for an informative data pair to match an unlabeled data point to one of the centroids of the existing neighborhoods. Also, they use an "Explore" stage to build an initial skeleton structure, which we already know to be a potential problem when there is a large number of clusters. Another approach by Huang and Lam (2009) has an active constrained clustering algorithm for documents with language modeling; it is not clear how we could adopt this algorithm for image clustering. In (Basu et al. 2002) the authors proposed a semi-supervised method for clustering using seeding, where labeled data is used to generate better initial cluster centers and to constrain the clustering solution; this approach does not have any active learning component. Some methods (Gomes et al. 2011) were also developed to use multiple humans to cluster datasets, where each user is asked to locally cluster a part of the dataset and finally those results are aggregated to find a complete partitioning.

Some active constraint clustering approaches are also known for image clustering (Biswas and Jacobs 2011; Grira et al. 2005). In (Biswas and Jacobs 2011), the authors have proposed a heuristic, which works well but has several parameters that must be tuned properly for different datasets. In (Grira et al. 2005), the authors take a fuzzy clustering based approach to find images near cluster boundaries to form useful data pairs.

As described in this section, there are many approaches in active learning, which want to maximize the information obtained from humans. Some of these methods were proposed for tasks, which are different from clustering with pairwise constraints. However there are some approaches (Mallapragada et al. 2008; Huang et al. 2007), which were proposed specifically for clustering with constraints and were designed to maximize the information gain from humans. However these active approaches measured the information gain using a local approach, where they do not consider the affect of a constraint on the complete dataset. For example, some constraints might be useful locally, i.e., can change clustering of a few points, but may not have high impact if the overall clustering of the dataset is considered. The novelty of our approach is in judging the effectiveness of a constraint by measuring how much it changes the overall clustering (a global approach), i.e., we expect a useful constraint to change assignment of many points in the dataset. A local approach can be useful when we already have a reasonable clustering to start with and we want to correct a few existing mistakes in that. On the other hand, a global approach like ours is more suitable for a problem like image clustering, where the initial clustering itself is very poor and it needs large changes to achieve a reasonable clustering. Thus, our method can ask useful questions from the very beginning and can achieve a good clustering by asking fewer questions than other methods.

3 Our Approach

We now describe our approach to active clustering. We first motivate this approach with a simple example, and then describe technical details.

3.1 High Level Intuition

We have developed a novel algorithm that determines which questions to ask a person in order to improve clustering. This algorithm is based on the intuitive idea of asking questions that will have the largest expected effect on the clustering. This really has two components. First, if we ask a person to compare two objects, her answer will only affect the clustering immediately if it differs from current clustering; that is, if the person says either that two objects that are not currently in the same cluster should be, or that two objects that are in the same cluster should not be. Any answer that differs from the current clustering must result in moving at least one object to a different cluster, but some answers will affect many more objects. So the second component of our algorithm is to ask questions whose answers might have a big effect on the way objects are clustered.

To provide a better intuition about our approach, we consider the simple toy example of clustering a set of 2D points. Figure 2 shows a collection of such points as black disks. The circles indicate four clusters that might be formed by an automatic clustering algorithm. We have marked five of these

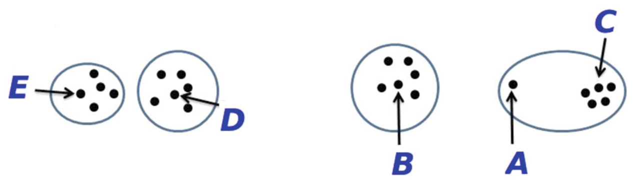


Fig. 2 Toy example to motivate our approach

points with the letters “A” to “E” for the ease of reference. Now suppose that an expert can compare two of these points and tell us whether they truly belong in the same cluster. Considering the following possibilities, we find:

- Comparing B and C is not that desirable, since it is likely that we have already correctly placed them in different clusters. A human opinion about these two points is unlikely to change anything.
- Comparing A and B (or A and C) will tell us in which cluster A truly belongs to. Since A is between two clusters, it is quite possible that this question will change the cluster to which we assign A. However, the answer to this question will only affect A.
- Comparing D and E will provide the most information. These two clusters are close, and it is somewhat ambiguous whether they should be separated into two clusters, or joined into one. So it is reasonably likely that a person might say D and E belong in the same cluster. If they do, this will lead us not only to treat D and E differently, but in fact to join the two clusters together, affecting the grouping of many points.

Consequently, we select questions for human attention that maximize the product of the probability that the answer will cause a change in our current clustering and the size of this change, should it occur. Finding the best such question can potentially require a large amount of computation. If we are clustering N objects, then there will be $O(N^2)$ same-or-different questions to consider, and for each we must determine its possible effects. For this reason, we adopt a simple, greedy clustering algorithm. Without human assistance, this algorithm does not perform well, but by using a simple clustering algorithm, we can more easily and quickly select good questions to ask a human, and rapidly improve our clustering performance. In Sect. 3.3 we explain the brute force version of our proposed algorithm and in Sect. 3.4 we describe a faster version of our algorithm. This faster version is possible due to the use of a simple and greedy clustering algorithm. In order to further speed up our algorithm, we have also experimented with two additional heuristics:

First, when our estimate of the probable response of a human indicates that it is very likely that the human response will agree with the current clustering, we do not bother to simulate the results of a different response. For all datasets we exclude simulation of pairs which are very unlikely to be in the same cluster. For larger datasets (of size more than 1000), we initially use K-means (MacQueen 1967) to group very close points together. These points are always clustered together. We represent them using their centroids and then run our algorithm. We refer to this heuristic as H1.

Second, we observe that when we simulate an additional constraint between a data pair, change in clustering assignments is often limited to clusters that contain the points in that pair. Determining those changes is much faster than checking for all possible changes. We perform experiments with this approximation and we find that it makes our algorithm’s performance a little worse but much faster. We refer to this heuristic as H2.

3.2 Components of Our Algorithm

We now explain our algorithm and its components formally. We define the best image pair that we will present to a person for labeling as:

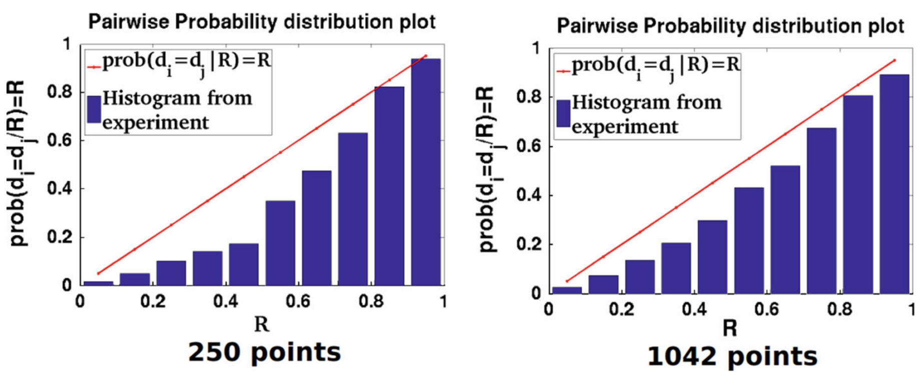
$$(\hat{d}_i, \hat{d}_j) = \underset{d_i, d_j}{\arg\max}\ E_J(d_i, d_j) \tag{1}$$

$E_J(d_i, d_j)$ is the expected change in the clustering if a question is asked about an image pair d_i and d_j. Since we do not know the ground truth clustering, we cannot choose image pairs that are guaranteed to increase the quality of the clustering. One of the main insights of our work is that by finding pairs that most rapidly *change* the clustering we quickly converge to a good clustering. Now, we formally describe how we compute the components needed to determine $E_J(d_i, d_j)$ given that we have a matrix containing all pairwise distances for a dataset.

3.2.1 Consensus Clustering Based Pairwise Probability Distribution

We first need to estimate the probability that each data pair will belong to the same cluster. We have borrowed ideas from consensus clustering (Punera and Ghosh 2008) (also referred to as aggregation of clustering) to find those probabilities. Consensus clustering typically computes multiple clusterings of the same dataset and then produces a single clustering assignment that reflects a consensus. This can be developed as an optimization problem that is known to be NP-complete. Consensus clustering in unsupervised learning is similar to ensemble learning in a supervised framework. However, we avoid optimization and just use multiple clusterings to estimate the probability that a pair of points belong to the same cluster.

Fig. 3 Pairwise probability distribution of real data



Specifically, if we have N data points and S clusterings, let $\mathbf{A_s}$ be the $N \times N$ matrix where element (i, j) is 1 if the i-th and j-th data points (referred as d_i and d_j respectively from now on) are in the same cluster in the s-th clustering, and zero otherwise. We use the K-means algorithm to produce clusters, each time beginning with different random initial cluster centroids. Now if $\mathbf{P}$ is the probability matrix for N data points where again element (i, j) is the pairwise probability of d_i and d_j being in the same cluster,

$$\mathbf{P} = \frac{1}{S}\sum_{s=1}^{S}\mathbf{A_s} \tag{2}$$

If (d_i, d_j) is any pair of points and R is a random variable corresponding to pair (d_i, d_j) resulting from the above random process then we estimate:

$$P(d_i, d_j) = \text{prob}(d_i = d_j|R) = R \tag{3}$$

$d_i = d_j$ implies d_i and d_j are from same class. We experimentally verify that this method produces reasonable values. In Fig. 3 (using images and pairwise distances from (Kumar et al. 2012)), we plot $\text{prob}(d_i = d_j|R) = R$ and a histogram generated from the results of those experiments showing the fraction of times that a pair with a given R is from the same cluster. Our heuristic provides a good estimate of $\text{prob}(d_i = d_j|R) = R$, which becomes more accurate with larger data sets.

3.2.2 Minimum Spanning Forests (MSF)

We perform clustering by creating a *Minimum Spanning Forest* (MSF). We define an MSF as a collection of trees which we get if we run Kruskal's algorithm (Kruskal 1956) and stop when we have K spanning trees. We can think of clustering as finding a forest with K spanning trees from a set of disconnected nodes. We use a constrained clustering algorithm that is very similar to Kruskal's algorithm, but also respects constraints. In the clustering community a similar approach is well-known as bottom up or hierarchical agglomerative clustering (HAC). We assume we are provided with the distance matrix for a given dataset. We can consider each image of the dataset as an individual node in a complete graph $G = (V, E)$ and let their mutual distances represent edge weights. We can also have information about a pair of images being in the same cluster ("must-link" constraints) or different clusters ("can't-link" constraints). Let us assume we create a copy of the graph G without any edges and call it $F = (V, \emptyset)$. First, we add edges corresponding to must-link constraints to the forest F. Next we sort the edges in E in an ascending order of their weights and store them in a queue E_p. We start popping edges from E_p and add them to F. While doing this, we always maintain the tree structure, i.e. do not add an edge between two nodes if the nodes are already connected. We will also not add an edge between two trees if there is any can't-link constraint between any of the pairs of nodes in those two trees. We continue doing this until we have K trees in the forest (referred as MSF in future). We refer to this algorithm as HAC with constraints (HACC).

We selected MSF as the base clustering algorithm for several reasons. First, MSF does not have any randomness involved. Since we want to measure the change in clustering for all possible pairs, we would not like the change in clustering to be affected by anything but the added constraint. Second, our high level idea is to simulate the results of all questions so that we can choose the question that has the greatest expected affect on the clustering. Doing this in a brute force way, even with the simplest clustering algorithm is not practical. So it is critical that we use a semi-supervised clustering algorithm in which it is possible to perform $O(N^2)$ incremental clustering operations efficiently. We have been able to develop fast ways of simulating the incremental effects of single constraints while using MSF as the semi-supervised clustering algorithm. Third, MSF's sensitivity to constraints, i.e., addition of some constraints can potentially change the MSF clustering result by a large amount, which helps us to measure the effectiveness of a constraint in a better way. However we emphasize that our use of this constrained version of minimum spanning forests is not meant to suggest that this semi-supervised clustering algorithm is competitive in performance with other methods. We expect MSF to work well along with the pairwise constraints which we obtain using our proposed active algorithm and that is demonstrated more in the experimental results (Sect. 4).

Since we are working with hierarchical clustering with constraints, we have to discuss feasibility and dead-end issues (Davidson and Ravi 2009). The feasibility problem is defined as given a set of data and set of constraints, does there exist a partitioning of the data into K clusters? In our problem the answer is obviously yes because all of the constraints are true. However determining whether there is a feasible solution which satisfies all constraints is NP-complete (Davidson and Ravi 2009). In HACC, dead-end situations (reaching an iteration with more than K clusters, where no further pair of clusters can be joined without violating any of the constraints) can occur in principle, but in practice we do not encounter this problem.

3.3 Our Algorithm (Active-HACC)

In Sect. 3.2.1, we estimate a distribution that allows us to determine the probability that a person will provide a constraint that differs from the current clustering. In this section, we determine the magnitude of the change this will have on the current clustering. To do this, we define a measure of similarity between two clustering assignments, which we call Relative Jaccard's Coefficient, by analogy to the Jaccard's Coefficient (Tan et al. 2005). If C_1 and C_2 are two clustering assignments of the same dataset, the Relative Jaccard's Coefficient of clustering C_2 with respect to C_1 is defined as:

$$\mathrm{JCC}_{C_1}(C_2) = \frac{SS}{SS + SD + DS} \tag{4}$$

where SS is the number of pairs that are in the same cluster in both C_1 and C_2, SD is the number of pairs that are in same cluster in C_1 but in different clusters in C_2, and DS is the number of pairs that are in different clusters in C_1 but are in same cluster in C_2. This becomes the traditional Jaccard's coefficient if C_1 is the ground truth clustering.

Now, we describe our algorithm, assuming that we have asked q questions and need to determine the $(q+1)$-th question to ask. We define:

D: The dataset, that should be clustered.
N: Number of images in the dataset.
K: Number of clusters.
d_i, d_j: Any pair of images from D.
C_q: The set of can't-link constraints obtained after we have asked q questions.
M_q: The set of must-link constraints obtained after we have asked q questions. Note $|C_q| + |M_q| = q$.
$H_q = \mathrm{HACC}(D, C_q, M_q)$: HACC is the clustering function on a given dataset D, using the must-link constraints (M_q) and can't-link constraints (C_q). H_q is the clustering that is produced.
$J_{q+1}^{d_i,d_j,\mathbf{y}} = \mathrm{JCC}_{H_q}(\mathrm{HACC}(D, C_q, M_q \cup d_i = d_j))$: Relative Jaccard's Coefficient of a clustering after $q+1$ questions with respect to H_q, if the $(q+1)$-th constraint is that d_i and d_j are in same cluster.
$J_{q+1}^{d_i,d_j,\mathbf{n}} = \mathrm{JCC}_{H_q}(\mathrm{HACC}(D, C_q \cup d_i \neq d_j, M_q))$: Relative Jaccard's Coefficient of a clustering after $q+1$ questions with respect to H_q, if the $(q+1)$-th constraint is that d_i and d_j are not in same cluster.

Now, we ask the user about the pair:

$$(\hat{d_i}, \hat{d_j}) = \underset{d_i, d_j}{\arg\max}\, E_J(d_i, d_j) \tag{5}$$

where $E_J(d_i, d_j)$ is the expected change in the Relative Jaccard's Coefficient and is defined as follows:

$$E_J(d_i, d_j) = \Big|\mathrm{JCC}_{H_q}(H_q) - (P(d_i, d_j) J_{q+1}^{d_i,d_j,\mathbf{y}} + (1 - P(d_i, d_j)) J_{q+1}^{d_i,d_j,\mathbf{n}})\Big| \tag{6}$$

Note that $\mathrm{JCC}_{H_q}(H_q) = 1$ and $P(d_i, d_j)$ is the probability of d_i and d_j being in the same cluster.

Now we can see that if points d_i and d_j are in the same cluster after q questions, then $\mathrm{HACC}(D, C_q, M_q \cup d_i = d_j) = H_q$ and if they are in different clusters then $\mathrm{HACC}(D, C_q \cup d_i \neq d_j, M_q) = H_q$. So we have:

– d_i and d_j are in the same cluster after q questions:

$$E_J(d_i, d_j) = \Big|(1 - P(d_i, d_j))(1 - J_{q+1}^{d_i,d_j,\mathbf{n}})\Big| \tag{7}$$

– d_i and d_j are in different clusters after q questions:

$$E_J(d_i, d_j) = \Big|P(d_i, d_j)(1 - J_{q+1}^{d_i,d_j,\mathbf{y}})\Big| \tag{8}$$

Using this approach, we find the data pair that will produce the greatest expected change in the clustering. After receiving an answer from the user, we update our constraints and continue.

When we plot the clustering performance, we show the Jaccard's Coefficient relative to ground truth, as the number of questions increases. This curve does not always increase monotonically, but it generally increases. Pseudo code of our proposed active algorithm is shown in Algorithm 1.

Algorithm 1 Active-HACC

```
Given: $D$, Max_questions, $M_q = \emptyset$, $C_q = \emptyset$, num_questions=0
while num_questions $\leq$ Max_questions do
  HACC($D$,$M_q$,$C_q$)
  For all pairs $(d_i, d_j)$, evaluate $E_J(d_i, d_j)$
  Find the pair $(d_i, d_j)$ with maximum $E_J(d_i, d_j)$
  Ask and Update $M_q$ and $C_q$
  num_questions=num_questions+1
end while
Output: HACC($D$,$M_q$,$C_q$)
```

3.3.1 Complexity of the Algorithm

We now discuss the rather high complexity of the brute-force version of Active-HACC. We will then consider several optimizations and heuristics to improve this run time. For each question we have $O(N^2)$ (N is the number of images) possible pairs. To simulate what will happen for each pair in a brute force manner, we will have to run the constrained clustering algorithm for each pair. We now describe the complexity of Active-HACC.

Kruskal's algorithm (Kruskal 1956) for minimum spanning tree runs in $O(N^2 \log N)$ time (if $|E| = O(N^2)$). HACC also has $O(N^2 \log N)$ complexity from a very similar analysis to Kruskal's, except for the issue of keeping track of the can't-links. To do this efficiently, we maintain an $l \times l$ lower triangular matrix $\mathbf{A}$ in which l is the current number of clusters. $A(m, n) = 1$ if $m > n$ and there is a can't-link constraint between clusters m and n, and $A(m, n) = 0$ otherwise. Before merging two clusters m and n, we check that $A(m, n) = 0$ $(m > n)$. In this case, we assign cluster n to have identity m. We update A by setting row m equal to the OR of rows m and n, and removing row and column n. This update takes $O(N)$ time, and can occur in $O(N)$ iterations. So enforcing the can't-link constraints adds $O(N^2)$ time to Kruskal's algorithm, which still runs in $O(N^2 \log N)$ time.

If we run this variation on Kruskal's algorithm for $O(N^2)$ pairs, the complexity of choosing each question will be $O(N^4 \log N)$. Even with moderate N (say N = 100) we cannot ask $O(N)$ questions with this much cost for each question. So we will propose a much less computationally complex version of Active-HACC in Sect. 3.4.

In part, this complexity helps motivate our use of a very simple clustering algorithm such as HACC. Since we must simulate $O(N^2)$ possibilities for each question, it is important that the clustering algorithm be relatively cheap. Moreover, as we will see, HACC lends itself to incremental clustering, in which simulating the effects of one constraint can be done efficiently. At the same time, HACC is quite interesting because the addition of one constraint can in some cases significantly alter the clustering.

3.4 FAST-Active-HACC

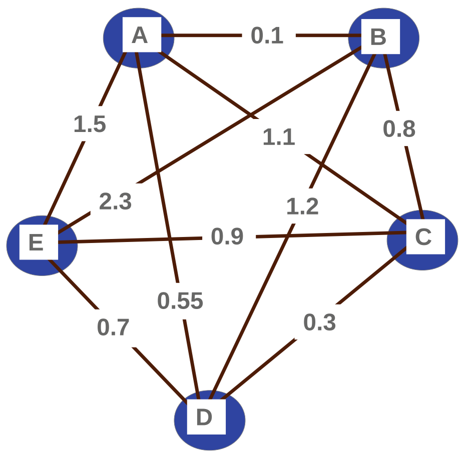
Our previous analysis assumes that we rerun HACC $O(N^2)$ times to simulate the effects of every question we consider asking. This is wasteful, since most new constraints only have a small effect on the clustering. We save a good deal of effort by incrementally computing these changes starting from the existing clustering. However, these changes can be somewhat complex. When a can't-link constraint is added to points in the same cluster, we must remove an edge from the current MSF, to disconnect these points. Removing one edge can have a domino effect, since there may be other edges that would have been added if that edge had not been present. Similarly, adding a must-link constraint might require us to merge two clusters that have can't-link constraints between them, requiring us to remove edges to separate any points connected by a can't-link constraint. We must simulate any effects of these changes.

Our complete algorithm is quite complex, and more detail is described throughout this section. Here we provide a couple of key intuitions. First, we save a good deal of time by creating data structures once that can be used in $O(N^2)$ simulations. Whenever we add a can't-link constraint between two points in a cluster, we must remove the last edge added to the MSF that is on the path between these points. To facilitate this, we keep track of the path on the MSF between all pairs of points in the same cluster. Also, as we perform the initial clustering, whenever an edge is not added to the MSF because of a can't-link constraint or because it would destroy the tree structure, we keep track of any edge that blocks it. That is, edge B blocks edge A if edge A would be added if not for the presence of edge B. Then, whenever an edge is removed, we can reconsider adding any edge that was blocked by this edge. Of course, as we "go back in time" and make changes to the MSF, we must carefully account for any later decisions that might also change.

In a second optimization, we notice that we do not need to simulate the effects of all can't-link constraints. If a cluster has N_c elements, there are $\binom{N_c}{2}$ possible can't-link constraints, but only N_c possible edges that can be removed. This means that many can't-link constraints will cause us to remove the same edge from the MSF, and have the same effect on the clustering. These can be simulated together.

Since this overall procedure is complex, code has been made publicly available (http://www.umiacs.umd.edu/~arijit/code_1.html).

As described earlier, we are given a set of points and pairwise distances. We consider a complete graph $G = (V, E)$ where each pair of distinct nodes are connected by a unique edge and each edge has a weight given by the distance matrix. We are computing a minimum spanning forest (MSF) for G. To do this, we sort the edges, E by distance (in ascending order), place them in a priority queue, and pop them



Time Stamps

1	2	3	4	5	6	7	8	9	10
AB	CD	AD	DE	BC	CE	AC	BD	AE	BE

Sorted List: E

Fig. 4 Complete graph G = (V, E) with edge weights on the edges and a sorted edge list E

off the queue in turn. We will also refer to the sorted edge list as E. As we progress, we can divide E into four disjoint groups: Active edges (AE) are the set of edges that have been added to the MSF. The number of active edges may vary from $(N-K-M)$ to $(N-K)$, where M is the number of must-link constraints. Can't-link edges (CLE) are the set of edges that have been popped from the queue but have not been added to the MSF due to can't-link restrictions. Redundant edges (RE) are included in set RE if it was not added to the MSF because the pair of points connected by this edge were already in the same cluster. Since we want to maintain a forest, we avoid all cycles. Other edges (OE) are the set of edges in E which are still on the queue. We also put a time stamp, which is the index of an element in the sorted list, on each edge. An edge with lower weight will have a lower time stamp than an edge with higher weight. In Fig. 4, it is explained how we build the sorted list and apply time stamps. We use the abbreviations AE, CLE, RE and OE to denote both sets and an element of a set.

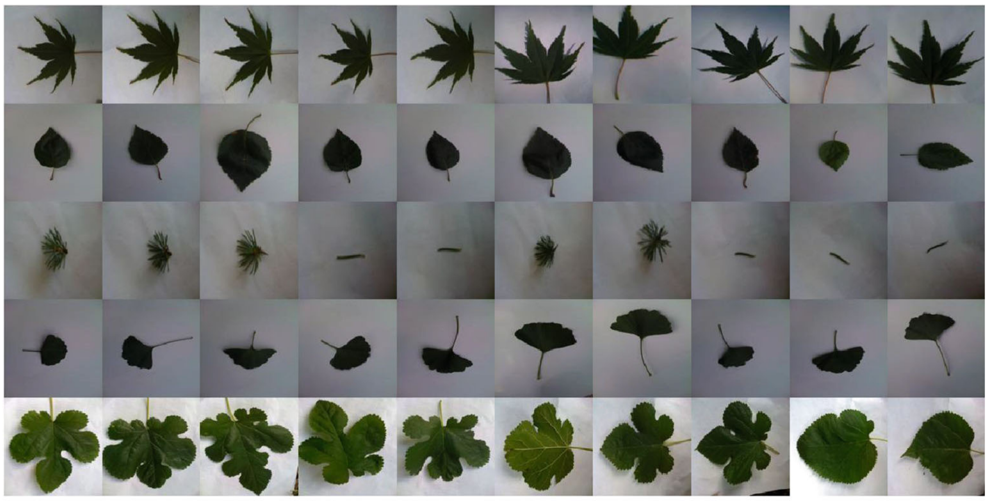
When we simulate adding a can't-link constraint to the same tree in an MSF, we must remove some edge, to break the tree into two components that each contain one of the nodes corresponding to the simulated can't-link constraint. We must then simulate HACC's behavior from the point at which the removed edge had been initially added. Similarly, if we simulate adding a must-link constraint that connects two trees that have a can't-link constraint between them, we must join these two trees, and then remove edges to break this new tree apart, so that again no tree violates can't-link constraints. In either event, we must undo some edge or edges that were added to the MSF, and determine which subsequent decisions might be altered by this change. To make this efficient, we maintain data structures that keep track of the dependencies between various decisions we have made. It is important to note that these data structures need to be constructed only once, for all $O(N^2)$ questions whose answers we simulate.

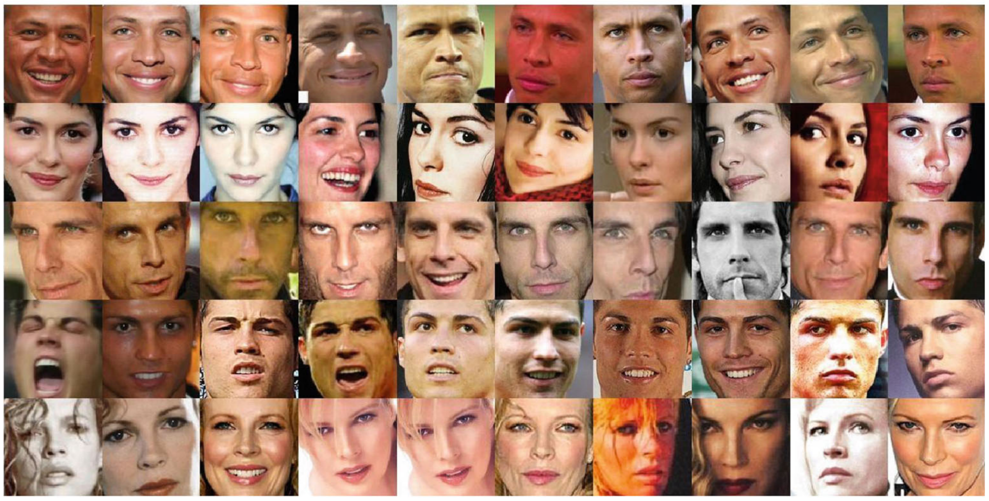
One of the interesting data structures, which we use when we simulate an additional constraint is a Priority Queue (PQ). In PQ we store possible edges to be added in the MSF but sorted based on their time stamps. This structure is reset after the simulation of each constraint. But there are some additional properties of the elements in this queue and we describe them below:

– Edges in the PQ must not be present in the MSF.
– The MSF should always be consistent while we add an edge from PQ to MSF or remove an edge from MSF. That means it can have more or less than K clusters but all the must-link and can't-link constraints must be satisfied.
– Once we start popping elements from the priority queue, if we have already considered adding an edge at time t, we never look at an edge with time stamp $T \leq t$. That implies any other edge with $T \leq t$ can not be added to the priority queue from now on.

In our framework we keep a priority queue with a set of edges, that can possibly be added to the MSF. We want to minimize the total number of elements added to PQ, which we denote as $|PQ|$. The minimum set is the set of edges, which we can definitely add to MSF. Although it is easier to find which of the CLE/REs are affected when we remove an active edge, tracking which of active edges could be affected while we add a CLE/RE is complicated. It is hard to find a polynomial time precomputation process for this part, which can be computed once and used for all possible question simulations. The precomputation in that case involves taking care of all possible cycles of a complete graph, which could take exponential time. So we have a simple work around, that works well in practice. We try to minimize $|PQ|$ as much as possible using our algorithm but we may not get the exact minimum of $|PQ|$. Although we may end up considering some extra edges for addition to the MSF, we make sure our faster algorithm is exactly same as the brute-force.

Using the priority queue and some additional data structures to store dependencies between different parts of the graph, we can do fast simulation of all possible constraints. We present the pair with maximum expected change in the Relative Jaccard's Coefficient to some user, get feedback and continue doing this until our human budget is exhausted or we get reasonable clustering.


(a) Leaf images from the leaf dataset we used


(b) Face images from the face dataset we used

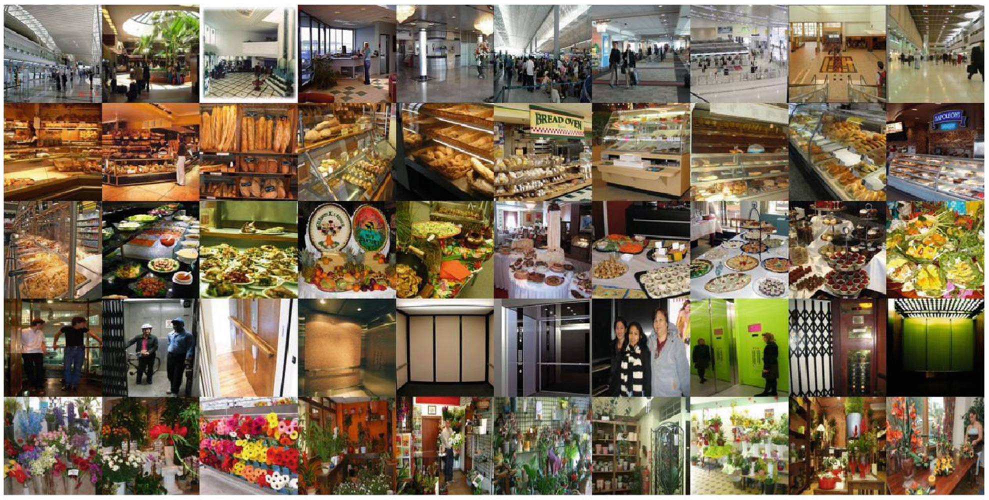

(c) Scene images from the scene dataset we used

Fig. 5 Some example images from the databases we used (each row in the montage shows images from the same class). **(a)** Leaf images from the leaf dataset we used. **(b)** Face images from the face dataset we used. **(c)** Scene images from the scene dataset we used

4 Experimental Results

We have experimented in three different domains, leaf, face and scene images. In Fig. 5 we show some images from the datasets that we used. For leaves, we create three subsets from a huge corpora of leaf images used for Leafsnap (Kumar et al. 2012). All leaf images are iPhone images of leaves on a white background. The leaf subsets are called Leaf-100 (containing 100 images from 10 different species), Leaf-250 (250 images from 25 different species) and Leaf-1042 (1,042 images from 62 species). Leaf-100 and Leaf-250 have the same number of leaves from all species. But in Leaf-1042, the number of leaves in each species vary from 4 to 31. Similarly for faces, we have extracted three subsets of images from a face dataset called PubFig (Kumar et al. 2009). The PubFig database is a large, real-world face dataset consisting of 58,797 images of 200 people collected from the Internet. Unlike most other existing face datasets, these images are taken in completely uncontrolled situations with non-cooperative subjects. Thus there is large variation in pose, lighting, expression, scene, camera, and imaging conditions, etc. The subsets of images are called Face-100 (100 images from 10 different people), Face-250 (250 images from 25 different people) and Face-500 (500 images from 50 different people). All of these face datasets have the same number of images in each class. For scenes, we have created a dataset called Scene-300 which has 300 images from 30 classes with each class having 10 images. Scene-300 is a subset of the Indoor Scene Recognition dataset (Quattoni and Torralba 2009). This dataset has images from different kinds of places such as airports, hospitals, libraries, kitchens etc. We first run experiments with perfect human responses, and then run a second experiment to evaluate the affect of real human responses.

The distance matrix for face images and leaf images are calculated based on algorithms described in P. Belhumeur (Personal communication with author, 2013) and Kumar et al. (2012) respectively. For scenes, 512 dimensional gist features are extracted from each image and Euclidean distances are calculated between all pairs of images to create the distance matrix. Once we get the distance matrix, we can run our proposed algorithm on all these datasets.

As described in Sect. 3.1 we have used a couple of heuristics named H1 and H2 to further speed up our algorithm. In H1, if the pairwise probability measure indicates that human response is going to be very similar to the current clustering, we do not bother to simulate a different response. In our experiments, if the probability that an image pair are from different classes is more than 0.9 and they are actually in different clusters, we do not simulate a must-link constraint for that pair. For larger datasets of size more than 1,000 we initially use K-means (MacQueen 1967) to group very close points together. For example for Leaf-1042, we cluster 1,042 images to 700 clusters and then run our algorithm with centroids of those 700 clusters. In H2, instead of computing the change in overall clustering, we only compute local changes involving the image pair under consideration. Often when a new constraint is added, the effects are very local and involves clusters which contain the image pair. We can determine these changes much faster than the overall change in clustering.

The main objective of running experiments on smaller datasets of size 100 and 250 is to show that even if we use heuristics the performance of the algorithm is not affected that much. The algorithm without heuristics is too slow to run on larger datasets. For example we run our algorithm on Face/Leaf-100 without any heuristic, with only H1 and then with H1 and H2 both. For Face/Leaf-250 we run our experiment with only H1 and then H1 and H2 both. For Face-500/Leaf-1042/Scene-300, we run a set of experiments using both heuristics H1 and H2.

4.1 Empirical Observations

We have run our algorithm on all of the datasets described above. All the algorithms to which we compare are described in Table 1. Figures 6 and 7 (where Jaccard's Coefficient corresponding to one misclassification per cluster is displayed using green squares) show performance evaluations of all the algorithms on Leaf-100, Face-100, Leaf-250, Face-250, Leaf-1042, Face-500 and Scene-300. We run K-means clustering 100 ($S = 100$) times on each dataset with different random initial set of centroids to find the pairwise probabil-

Table 1 Summary of the algorithms that we compare

Active-HACC	Proposed active learning version for Image Clustering
FAST-Active-HACC	Faster version of our proposed algorithm without any heuristic
FAST-Active-HACC-H1	Faster version of our proposed algorithm with H1 only
FAST-Active-HACC-H1_H2	Faster version of our proposed algorithm with H1 and H2 both
Random Constraints	Seek pairwise constraints randomly and use HACC
K-means without questions (MacQueen 1967)	Simple K-means without any human intervention
CAC1 (Biswas and Jacobs 2011)	A heuristic to find the best pair
Active-PCKMeans (Basu et al. 2004)	An active constrained clustering algorithm
COP-Kmeans (Wagstaff et al. 2001)	A semi-supervised clustering algorithm (with pairwise constraints)
(Xiong et al. 2012)	A spectral active clustering approach based on k nearest neighbor graph purification

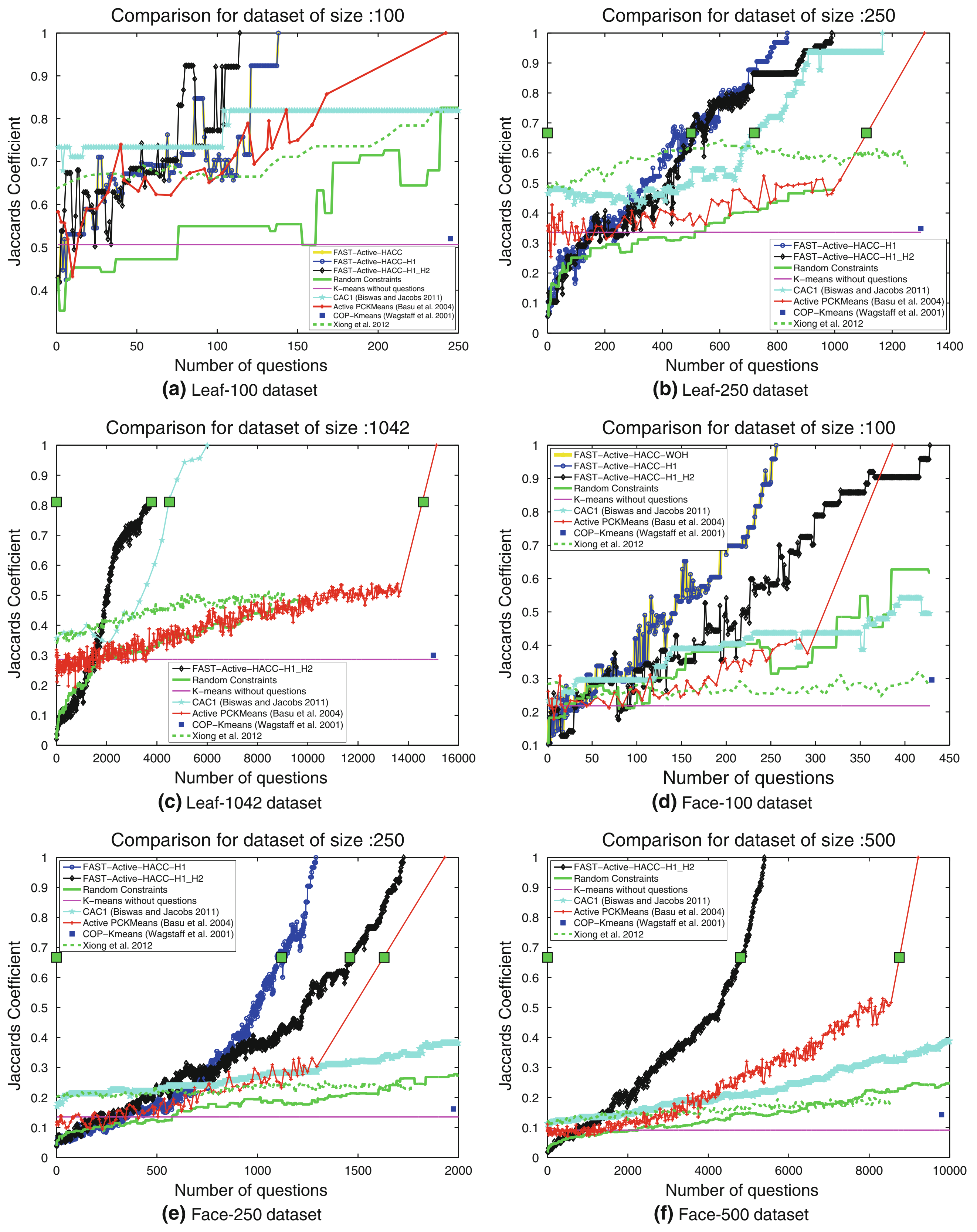


Fig. 6 Performance plot showing how the Jaccard's Coefficient increases with the number of questions. Our proposed algorithm Active-HACC significantly outperforms all other algorithms to which we compare. **(a)** Leaf-100 dataset, **(b)** Leaf-250 dataset, **(c)** Leaf-1042 dataset, **(d)** Face-100 dataset, **(e)** Face-250 dataset, **(f)** Face-500 dataset (Color figure online)

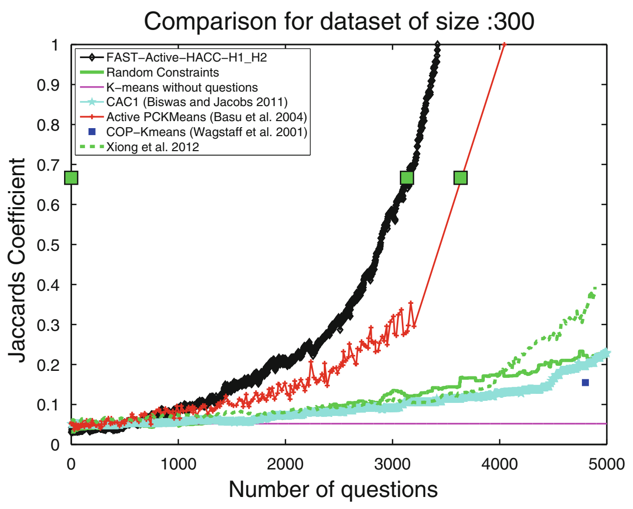


Fig. 7 Scene-300 dataset (Color figure online)

ity distributions. We see how Jaccard's Coefficient changes with the number of questions. Since we have the ground truth for these datasets we were able to calculate the Jaccard's Coefficient with respect to the ground truth. In real world situations we will not have the ground truth and will have to decide when we have reached a good clustering. One possible way to stop asking questions is when clustering assignments do not change even with additional constraints. Also, one of the advantages of FAST-Active-HACC is that we do not need to set any parameters. One of our main interests is in problems that grow big because the number of clusters becomes large. We make the following observations from the experiments:

– In all of these experiments our algorithm significantly outperforms all other algorithms. We were able to greatly reduce the number of constraints required for a reasonable clustering.
– We run both Active-HACC and FAST-Active-HACC for the Leaf-100 and Face-100 datasets. We see that FAST-Active-HACC is 25–30 times faster than Active-HACC for a dataset of size 100. Overall we expect FAST-Active-HACC to be $O(N)$ faster than Active-HACC. Since Active-HACC is slow, we could not experiment with it in larger datasets. We also observe that FAST-Active-HACC-H1 produces the exact same results as FAST-Active-HACC for Leaf-100/Face-100. For a dataset of size 1042, FAST-Active-HACC-H1_H2 takes around a minute per question. We believe this could be further sped up with more optimized code (our current implementation is in MATLAB) or parallel processing as our algorithm is highly parallelizable.
– In Fig. 6, we compare the results for different algorithms for the Leaf-1042 dataset. For this dataset only we use K-means initially as part of H1, to reduce the number of points to 700. Even with that, we get Jaccard's Coefficient of 0.8152 (one misclassification per cluster on average) by asking just 3,782 questions.
– We wanted to compare our algorithm with (Grira et al. 2005) and (Huang et al. 2007), but a direct comparison on our image datasets is not possible due to the complexity of their algorithm and the lack of publicly available code. However we also run experiments using the iris dataset to compare with (Grira et al. 2005), on which they have reported results. This is a relatively easy dataset with 150 points from three clusters in four dimensions. They have used the ratio of well categorized points to the total number of points as an evaluation metric (let us call it RWCP). Our algorithm reaches RWCP of 0.97 within three questions, where they take ten questions to reach the same RWCP.
– One of the major differences in these domains is the distance matrix. The leaf image distances are more accurate than the face images. Also the face image distances are more accurate than the scene images. So even if we have three similar sized datasets from three different domains, we need different amount of human interaction in each of these domains.

4.2 Experiments with Real Humans

In previously described experiments we assumed that humans are always correct when they provide pairwise constraints. We note that all previous active clustering algorithms (Basu et al. 2004; Xu et al. 2005; Huang et al. 2007; Mallapragada et al. 2008; Huang and Lam 2009) have explicitly made this assumption. Authors in (Kumar et al. 2009) demonstrate that this assumption is reasonable for face images, as humans are 99.2 % accurate in judging similarity of two faces. However we wanted to verify how robust our algorithm can be when we relax this assumption and allow real humans to judge similarity of images using Amazon Mechanical Turk. To allow for extensive quantitative evaluations while still using feedback from real users, we collected exhaustive human feedback for all possible pairs of images in our datasets Leaf-250 and Face-250 using Mechanical Turk. We ask about each image pair to a mechanical turk user to decide if a pair of images belong to the same category or not. In general Mturk workers found this task to be fun and interesting; their comments were "nice","good", "again provide these types of work" etc. However some workers provided random/noisy responses throughout a HIT, which we had to filter out. Overall we find that humans are wrong around 1.2 % of the time in judging the similarity of face images and around 1.9 % of the time in judging the similarity of leaf images. We plot the experimental results with real human input in Fig. 8. We find that with real human feedback we never achieve perfect clustering

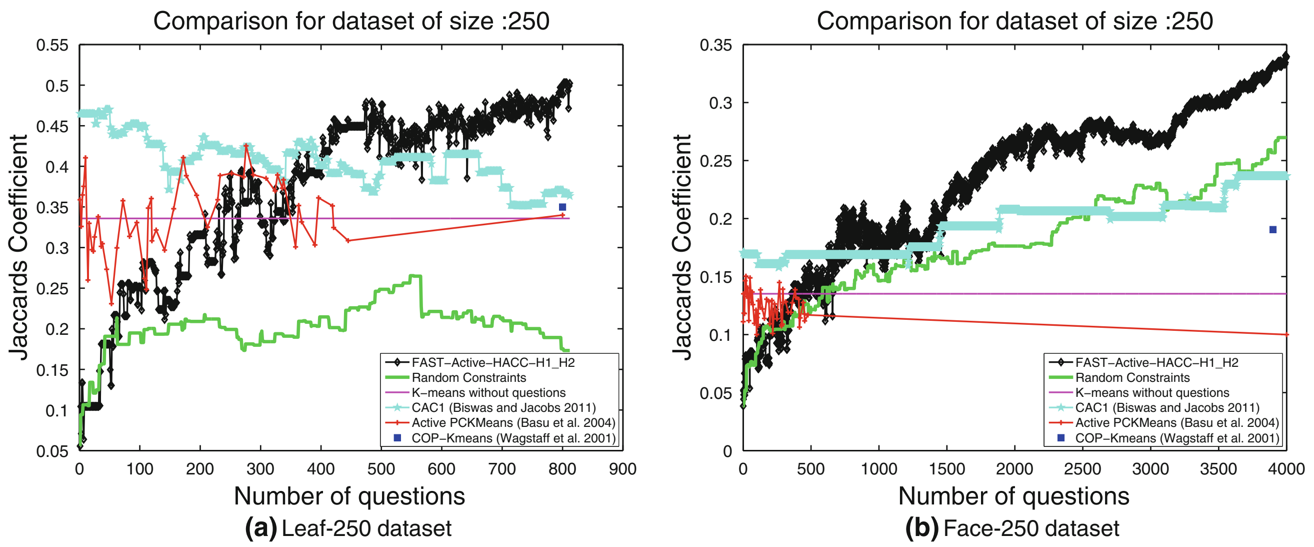


Fig. 8 Performance plot showing how the Jaccard's Coefficient increases with the number of questions when we get pairwise constraints from real humans in Mechanical Turk. **(a)** Leaf-250 dataset, **(b)** Face-250 dataset

(JCC = 1) because of the errors introduced by humans. Also note that after asking some number of questions all algorithms' performance start to become worse. Our experiments were run until when our proposed algorithm stopped improving. Determining a good stopping point automatically remains an interesting research problem. We find that our algorithm is still better than other active and passive algorithms. Although the heuristic proposed by (Biswas and Jacobs 2011) has a good starting point for Leaf-250, it's performance quickly becomes worse as real human input is introduced. However for Face-250, (Biswas and Jacobs 2011) does not perform well even in the beginning. Our proposed algorithm achieves maximum Jaccard's Coefficient (JCC) of 0.5 for Leaf-250 and 0.35 for Face-250. Note that JCC $= 0.5$ implies 20 clusters with 2 mistakes and 5 clusters with only one mistake on average (cluster size is 10). Similarly JCC $= 0.35$ implies 20 clusters with 3 mistakes and 5 clusters with 2 mistakes on average (cluster size is 10).

5 Conclusions

We have presented an approach for image clustering that incorporates pairwise constraints from humans. An algorithm is developed for choosing data pairs to use when querying a person for additional constraints. Since a brute force version of our algorithm is time consuming we also formulate a more complex but faster version in this paper. Our algorithm outperforms all state-of-the-art results in image clustering. Although this paper was focused on solving image clustering, this idea could be extended to any clustering domain in general.

References

Angluin, D. (1987). Queries and concept learning. *Machine Learning*, *2*(4), 319–342.

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. In C. Sammut & A. G. Hoffmann (eds.), *ICML* (pp. 27–34). San Francisco, CA: Morgan Kaufmann. ISBN 1-55860-873-7.

Basu, S., Banerjee, A., & Mooney, R. J. (2004). Active semi-supervision for pairwise constrained clustering. In *Fourth SIAM International Conference on Data Mining*. ISBN 0-89871-568-7.

Basu, S., Davidson, I., & Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms. Data mining and knowledge discovery series*. Washington, DC: IEEE Computer Society

Biswas, A. & Jacobs, D. (2011). Large scale image clustering with active pairwise constraints. In *International Conference in Machine Learning 2011 Workshop on Combining Learning Strategies to Reduce Label Cost*.

Biswas, A. & Jacobs, D. W. (2012). Active image clustering: Seeking constraints from humans to complement algorithms. In *IEEE Conference on CVPR* (pp. 2152–2159).

Branson, S., Perona, P., & Belongie, S. (2011). Strong supervision from weak annotation: Interactive training of deformable part models. In D. N. Metaxas, L. Quan, A. Sanfeliu, & L. J. Van Gool (eds.), *IEEE Internaional Conference on Computer Vision* (pp. 1832–1839). http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6118259.

Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., & Belongie, S. (2010). Visual recognition with humans in the loop. In K. Daniilidis, P. Maragos, & Paragios, N. (eds.), *ECCV Lecture Notes in Computer Science* (Vol. 6314, pp. 438–451). Berlin: Springer. ISBN 978-3-642-15560-4. doi:10.1007/978-3-642-15561-1.

Chellappa, R., & Jain, A. K. (1993). *Markov random fields: Theory and applications*. Boston: Academic Press.

Dagli, C. K., Rajaram, S., & Huang, T. S. (2006). Utilizing information theoretic diversity for SVM active learn. In *ICPR* (pp. 506–511). doi:10.1109/ICPR.2006.1161.

Davidson, I., & Ravi, S. S. (2009). Using instance-level constraints in agglomerative hierarchical clustering: Theoretical and empirical results. *Data Mining and Knowledge Discovery*, *18*(2), 257–282.

Felzenszwalb, P. F., McAllester, D., Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *CVPR* (pp. 1–8). doi:10.1109/CVPR.2008.4587597.

Gomes, R., Welinder, P., Krause, A.,& Perona, P. (2011). Crowdclustering. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, & K. Q. Weinberger (eds.), *NIPS* (pp. 558–566). http://books.nips.cc/nips24.html

Grira, N., Crucianu, M., & Boujemaa, N. (2005). Active semi-supervised fuzzy clustering for image database categorization. In *ACM SIGMM International Workshop on Multimedia Information Retrieval* (pp. 9–16).

Guo, Y. & Greiner, R. (2007). Optimistic active-learning using mutual information. In M. M. Veloso (ed.), *IJCAI* (pp. 823–829). http://dli.iiit.ac.in/ijcai/IJCAI-2007/PDF/IJCAI07-132.pdf

Holub, A. D., Perona, P., & Burl, M. C. (2008). Entropy-based active learning for object recognition. In *Online Learning for Classification Workshop* (pp. 1–8). doi:10.1109/CVPRW.2008.4563068.

Huang, R., & Lam, W. (2009). An active learning framework for semi-supervised document clustering with language modeling. *Data & Knowledge Engineering*, *68*(1), 49–67.

Huang, R., Wai, L. & Zhigang, Z. (2007). Active learning of constraints for semi-supervised text clustering. In *SDM, SIAM*.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666. doi:10.1016/j.patrec.2009.09.011.

Jain, P. & A. Kapoor. (2009). Active learning for large multi-class problems. In *CVPR* (pp. 762–769). doi:10.1109/CVPRW.2009.5206651.

Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2010). Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback. In *IEEE Conference on CVPR* (pp. 2995–3002). doi:10.1109/CVPR.2010.5540047.

Kapoor, A., Horvitz, E., Basu, S. (2007). Selective supervision: Guiding supervised learning with decision-theoretic active learning. In M. M. Veloso (ed.), *IJCAI* (pp. 877–882). http://dli.iiit.ac.in/ijcai/IJCAI-2007/PDF/IJCAI07-141.pdf.

Kruskal, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the AMS*, *2*, 48–50.

Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., & Soares, J. V. B. (2012) Leafsnap: A computer vision system for automatic plant species identification. In *ECCV 2, Lecture Notes in Computer Science* (Vol. 7573, pp. 502–516).

Kumar, N., Berg, A. C., Belhumeur, P. N., & Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *IEEE Conference on ICCV* (pp. 365–372).

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematics Statistics and Probability*.

Mallapragada, P. K., Jin, R., & Jain, A. K. (2008). Active query selection for semi-supervised clustering. In *ICPR* (pp. 1–4).

Martinez, A. M., & Benavente, R. (1998). *The AR face database*. CVC Technical, Report #24.

Punera, K., & Ghosh, J. (2008). Consensus-based ensembles of soft clusterings. *Applied Artificial Intelligence*, *22*(7&8), 780–810.

Quattoni, A. & Torralba, A. (2009). Recognizing indoor scenes. In *CVPR* (pp. 413–420). doi:10.1109/CVPRW.2009.5206537.

Settles, B. (2010). *Active learning literature survey*. Technical report.

Siddiquie, B., & Gupta, A. (2010). Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *IEEE Conference on CVPR* (pp. 2979–2986). doi:10.1109/CVPR.2010.5540044.

Sugar, C. A., & James, G. M. (2003). Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association*, *98*, 750–763.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). Introduction to Data Mining. Upper Saddle River, NJ: Addison-Wesley. ISBN 0-321-32136-7.

Vijayanarasimhan, S., & Grauman, K. (2011). Large-scale live active learning: Training object detectors with crawled data and crowds. In *IEEE conference on CVPR* (pp. 1449–1456). doi:10.1109/CVPR.2011.5995430.

Vijayanarasimhan, S. & Grauman, K. (2012). Active frame selection for label propagation in videos. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, & C. Schmid (eds.), *ECCV 5, Lecture Notes in Computer Science* (Vol. 7576, pp. 496–509). Berlin: Springer. ISBN 978-3-642-33714-7.

Vijayanarasimhan, S., Jain, P., & Grauman, K. (2010). Far-sighted active learning on a budget for image and video recognition. In *IEEE Conference on CVPR* (pp. 3035–3042). doi:10.1109/CVPR.2010.5540055.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-means clustering with background knowledge. In *Proceedings of 18th International Conference on Machine Learning* (pp. 577–584). San Francisco, CA: Morgan Kaufmann.

Wang, X., & Davidson, I. (2010). Active spectral clustering. In *ICDM* (pp. 561–568). IEEE Computer Society.

Xiong, C., Johnson, D. & Corso, J. J. (2012). Spectral active clustering via purification of the *k*-nearest neighbor graph. In *ECDM*.

Xu, Q., desJardins, M., & Wagstaff, K. (2005). Active constrained clustering by examining spectral eigenvectors. In *Discovery, science* (Vol. 3735). Berlin: Springer.