Active Image Clustering: Seeking Constraints from Humans to Complement Algorithms

Arijit Biswas and David Jacobs
Computer Science Department
University of Maryland, College Park
arijitbiswas87@gmail.com, djacobs@umiacs.umd.edu

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 heuristics, and show that these do not significantly impact the effectiveness of our method. We have run experiments in two different domains, namely leaf images and face images, and show that clustering performance can be improved significantly.

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, producing many interesting and effective algorithms. However, all clustering depends 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 clustering produced by automatic means.

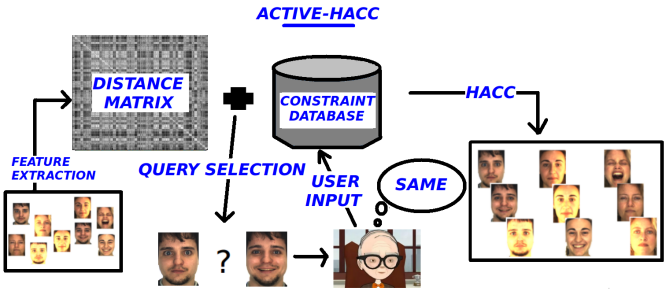


Figure 1: Pipeline of our system: Active-HACC is our proposed active algorithm for selecting data-pairs to get constraints and HACC is a constrained clustering algorithm.

To illustrate the value of this approach, we use the example 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 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 [6]. Classifiers that can identify species require large sets of leaf images, labeled by species. Accurately labeling such images requires experience and botanical knowledge. One approach that can re-

duce this effort is to cluster all the images into groups 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.

We also assume that a person always provides correct constraints and that we know the number of clusters beforehand. In practice, humans are highly accurate at image comparison. For example, [13] shows that humans achieve 99.2% accuracy in a face comparison task, even without the option of responding "don't know". 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 [19] 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 Figure 1 (face images from [16]).

We do experiments to evaluate our algorithm in two different domains: face images and leaf images. Since we assume that people are highly accurate, in experiments we can simulate their behavior using ground truth.

2. Related Work

Combining active learning [2, 18] with constrained clustering [4] has been a growing area of interest in machine learning as well as in computer vision. Some active constraint clustering approaches are also known for image clustering [7, 9]. In [7], the authors have proposed a heuristic, which works reasonably well but has several parameters that must be tuned properly for different datasets. In [9], the authors take a fuzzy clustering based approach to find images near cluster boundaries to form useful data pairs.

In [3], Basu *et al.* also proposed 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. Mallapragada *et al.* [15] have proposed an approach that uses a min-max criterion to find informative questions. They rely on the "Explore" stage in the beginning as [3] does. There are also a couple of active clustering algorithms [22, 23] based on spectral eigenvectors, but they are good for two-cluster problems only.

In [11], Huang *et al.* 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 *et al.* [10] has an active constraint clustering algorithm for documents with language modeling and it is not clear how we could adopt this algorithm for image clustering.

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 it; 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 our 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.

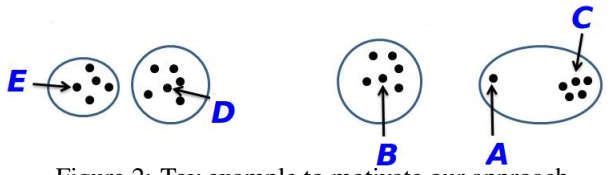


Figure 2: Toy example to motivate our approach.

To provide 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 points with the letters "A" to "E" for ease of reference. We now imagine 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. 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 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 [14] to group very close points together and 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 more formally. We define the best image pair that we will present to a person for labeling as:

$$(\hat{d_i}, \hat{d_j}) = \arg\max_{d_i, d_j} E_J(d_i, d_j) \quad (1)$$

$E_J(d_i, d_j)$ 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 we have any distance matrix for a dataset.

3.2.1 Consensus Clustering based Pairwise Probability Distribution

We first need to estimate the probability that each data pair will be in same cluster. We have borrowed ideas from consensus clustering [17] (also referred to as aggregation of clustering) to find those probabilities. In consensus clustering we typically compute multiple clusterings of the same dataset and then produce a single clustering assignment that reflects a consensus. Consensus clustering in unsupervised learning is similar to ensemble learning in a supervised framework. However we avoid any optimization, and just use multiple clusterings to estimate the probability that the elements of a pair belong to the same cluster.

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 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 the i-th and j-th data points 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 $P(d_i, d_j) = \text{prob}(d_i = d_j|R) = R$. $d_i = d_j$ implies d_i and d_j are from same class. We experimentally verify that this method produces reasonable values. In Figure 3 (data from [1]), 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 grows more accurate for larger data sets.

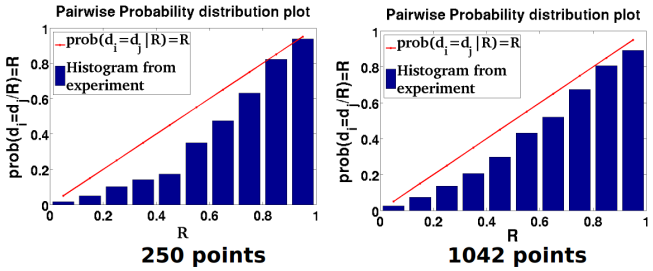


Figure 3: Pairwise probability distribution of real data.

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[12] 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 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 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 will discuss later why we build on the MSF rather than constrained K-means [21] or other constrained algorithms [24].

Since we are working with hierarchical clustering with constraints we have to discuss feasibility and dead-end issues [8]. 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 satisfy all constraints is NP-complete [8]. 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 find this is not a problem.

3.3. Our Algorithm (Active-HACC)

In 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 [20]. 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:

$$\text{JCC}_{C_1}(C_2) = \frac{SS}{SS + SD + DS} \tag{3}$$

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.

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 = 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.

$J_{q+1}^{d_i,d_j,\mathbf{y}}$ = JCC$_{H_q}$(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.

$J_{q+1}^{d_i,d_j,\mathbf{n}}$ = JCC$_{H_q}$(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 constraints is that d_i and d_j are not in same cluster.

Now, we ask the user about the pair:

$$(\hat{d_i}, \hat{d_j}) = \arg\max_{d_i,d_j} E_J(d_i, d_j) \tag{4}$$

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

$$E_J(d_i, d_j) = |\text{JCC}_{H_q}(H_q) - (P(d_i, d_j) J_{q+1}^{d_i,d_j,\mathbf{y}} + (1 - P(d_i, d_j)) J_{q+1}^{d_i,d_j,\mathbf{n}})| \tag{5}$$

Note that 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 HACC$(D, C_q, M_q \cup d_i = d_j) = H_q$ and if they are in different clusters then 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:

$$E_J(d_i, d_j) = |(1 - P(d_i, d_j))(1 - J_{q+1}^{d_i,d_j,\mathbf{n}})| \tag{6}$$

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

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

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.

3.3.1 Complexity of the algorithm

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

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)

Kruskal's algorithm [12] 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 $O(N^4 logN)$. 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 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 complex, and is described in the supplementary material (available on author's webpage). 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 which 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 will be made publicly available.

4. Experimental Results

We have experimented in two different domains, leaf images and face images. For leaves, we create three subsets from a huge corpora of leaf images used for Leafsnap [1]. 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 (1042 images from 62 species). Leaf-100 and Leaf-250 have same number of leaves from all the 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 [13]. 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 same number of images in each class.

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 [14]	Simple K-means without any human intervention
CAC1 [7]	A heuristic to find the best pair
Active-PCKMeans [3]	An active constrained clustering algorithm

Table 1: Summary of the the algorithms that we compare.

The distance matrix for face images and leaf images are calculated based on algorithms in [5]. Once we get the distance matrix, we can run our proposed algorithm on all these datasets. 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 effected 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/Face-1042, we run one 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 that we compare are described in Table 1. Figures 4a-4d (where Jaccard's Coefficient corresponding to one misclassification per cluster is displayed using green squares) show performance evaluations of all the algorithms on Leaf-250, Face-250, Leaf-1042 and Face-500 (we include results for Leaf-100 and Face-100 in the supplementary material). We use S=100 to find the pairwise probability 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 (shown in supplementary material). 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 Figure 4b, we compare the results for different algorithms for the Leaf-1042 dataset. For this dataset only we use K-means initially as part of H_1, 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 3782 questions.
- We wanted to compare our algorithm with [9] and [11], 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 [9], on which they have reported results. This is a relatively easy dataset with 150 points from 3 clusters in 4 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. In leaf images the distance matrix is more accurate than in face images. So even if we have two similar datasets from two different domains, we need more questions for face images than leaf images. For smaller datasets in which the distance matrix is not very accurate, FAST-Active-HACC-H1_H2 becomes comparable to, though still better than [3].

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.

Acknowledgment: This work was supported by NSF grant #0968546 and Army Research Office, ARO #W911NF0810466.

References

[1] http://leafsnap.com/. 4, 6

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

[3] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Fourth SIAM International Conference on Data Mining*, 2004. 2, 6, 7

[4] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms*. Data Mining and Knowledge Discovery Series. IEEE Computer Society, 2008. 2

[5] P. Belhumeur. Personal communication with author. 2011. 6

[6] P. N. Belhumeur, D. Chen, S. Feiner, D. W. Jacobs, W. J. Kress, H. B. Ling, I. Lopez, R. Ramamoorthi, S. Sheorey, S. White, and L. Zhang. Searching the world's herbaria: A system for visual identification of plant species. In *ECCV*, pages IV: 116–129, 2008. 1

[7] A. Biswas and D. Jacobs. Large scale image clustering with active pairwise constraints. *International Conference in Machine Learning 2011 Workshop on Combining Learning Strategies to Reduce Label Cost*. 2, 6

[8] I. Davidson and S. S. Ravi. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data Min. Knowl. Discov*, 18(2):257–282, 2009. 4

[9] N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised fuzzy clustering for image database categorization. In *Multimedia Information Retrieval*, pages 9–16. ACM, 2005. 2, 7

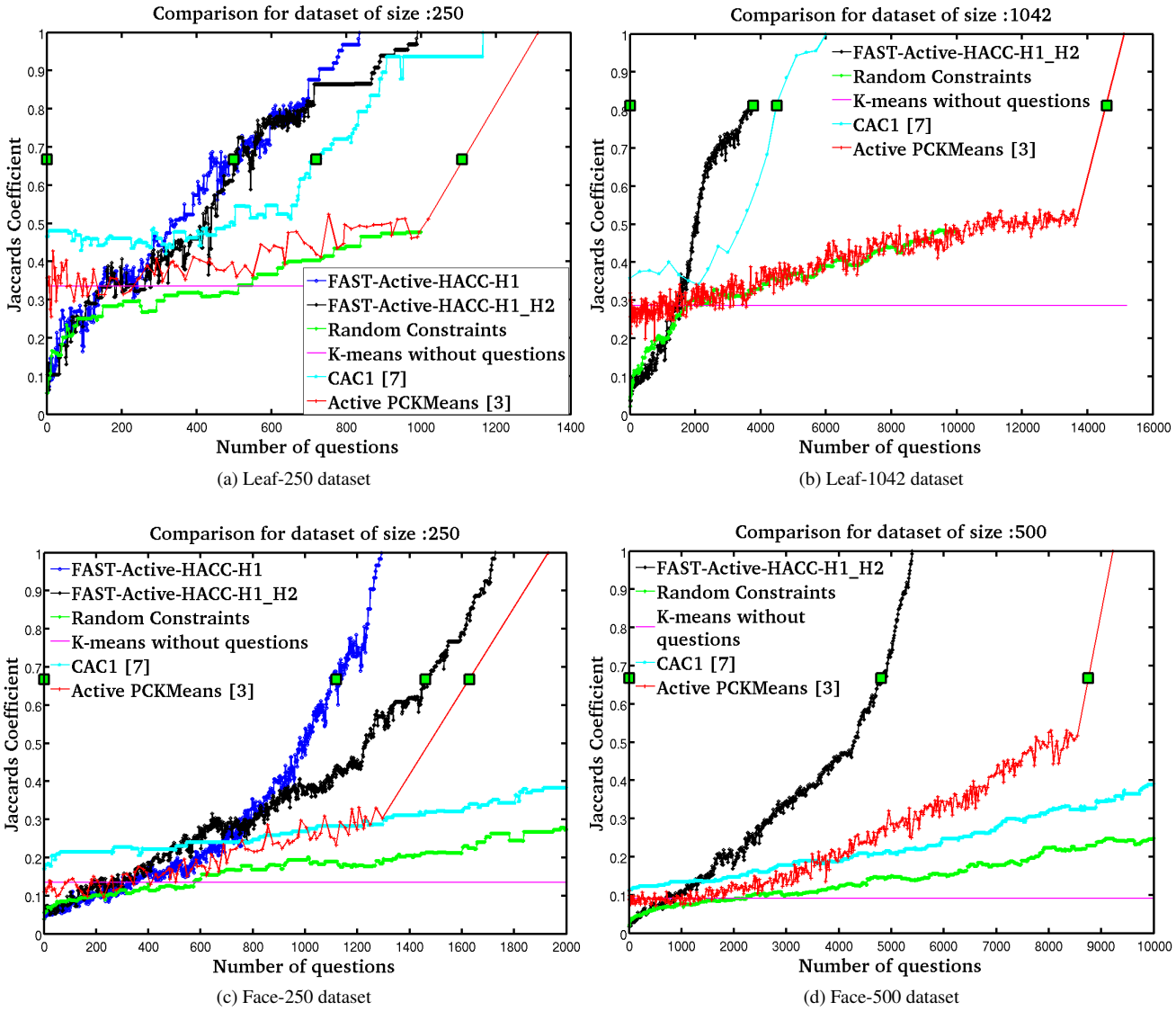
[10] R. Huang and W. Lam. An active learning framework for semi-supervised document clustering with language modeling. *Data Knowl. Eng*, 68(1):49–67, 2009. 2

[11] R. Huang, W. Lam, and Z. Zhang. Active learning of constraints for semi-supervised text clustering. In *SDM*. SIAM, 2007. 2, 7

[12] Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the AMS*, 2:48–50, 1956. 4, 5

[13] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, pages 365–372. IEEE, 2009. 2, 6

[14] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Mathematics Statistics and Probability*, 1967. 3, 6



(a) Leaf-250 dataset

(b) Leaf-1042 dataset

(c) Face-250 dataset

(d) Face-500 dataset

Figure 4: Performance plot showing how the Jaccard's Coefficient increase with the number of questions. Our proposed algorithm Active-HACC significantly outperforms all other algorithms we compare.

[15] P. K. Mallapragada, R. Jin, and A. K. Jain. Active query selection for semi-supervised clustering. In *ICPR*, pages 1–4, 2008. 2

[16] A. Martinez and R. Benavente. The AR Face Database. *CVC Technical Report #24*, 1998. 2

[17] K. Punera and J. Ghosh. Consensus-based ensembles of soft clusterings. *Applied Artificial Intelligence*, 22(7&8):780–810, 2008. 3

[18] B. Settles. Active learning literature survey. Technical report, 2010. 2

[19] C. A. Sugar and G. M. James. Finding the number of clusters in a data set: An information theoretic approach. pages 750–763, 2003. 2

[20] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. 4

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *ICML*, pages 577–584, 2001. 4

[22] X. Wang and I. Davidson. Active spectral clustering. In *ICDM*, pages 561–568. IEEE Computer Society, 2010. 2

[23] Q. Xu, M. desJardins, and K. Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science*, volume 3735. Springer, 2005. 2

[24] S. Yan, A. H. Wang, D. Lee, and C. L. Giles. Pairwise constrained clustering for sparse and high dimensional feature spaces. In *PAKDD*, volume 5476 of *Lecture Notes in Computer Science*, pages 620–627. Springer, 2009. 4