Large Scale Image Clustering with Active Pairwise Constraints

Arijit Biswas
David Jacobs
University of Maryland, College Park, USA

Abstract

In this paper, we propose a method of clustering large image sets using human input. We assume an algorithm provides us with pairwise similarities. We then actively ask for more accurate pairwise similarities between images from humans. Using all similarities, we cluster the images and show that the improvement gain is significant even when the available human resources are very limited.

1. Introduction

The high level motivation behind our work is to bring humans in the loop to improve the performance of computer vision algorithms. Algorithms, once developed, are cheap but human attention is costly. Since algorithms alone do not always perform with sufficient accuracy, so humans can play a significant role in improving standard machine learning or computer vision algorithms.

There has been a great deal of interest recently in obtaining large, labeled image sets. We experiment using images from the domain of automatic plant species identification (Belhumeur et al., 2008). To create classifiers that can identify species, large sets of leaf images must be labeled by species. Accurately labeling such images requires experience and botanical knowledge.

One approach that can reduce this effort is to cluster all the images into groups that each come from a single species, and then 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. Therefore, we propose to approach the problem of image labeling by first clustering leaves, using input from algorithms and naive users, so that professional botanists can later label them efficiently.

To do this, we draw on ideas from two well established machine learning fields, active learning (Settles, 2010) and constrained clustering (Basu et al., 2008). In active learning algorithms are allowed to query users interactively. Complementary to this, constrained clustering uses partial, pairwise knowledge about data. In this paper, we present method of actively asking for pairwise similarity between images (constraints) and we cluster images using those constraints. We choose the set of pairs serially and only after a pair is marked as similar or not, do we choose the next one, using a greedy approach.

We use a large leaf image dataset (Belhumeur et al., 2008), which was collected in a real-world project to develop interactive species identification tools, for our experiments. In this work, we consider the idealization that human inputs come without noise (they are always correct); in future work, we will incorporate uncertainty in human inputs. We compare our algorithm with (Basu et al., 2004), which is one of the best and closest to ours in assumptions.

2. Related Work

Active learning has been studied for many years (e.g., (Angluin, 1987)). The three main settings which have been considered in the field are membership query synthesis, stream based selective sampling and pool based sampling (Settles, 2010). Similarly constrained clustering is also a well explored area. The book by Basu et al (Basu et al., 2008) gives a good overview of this area.

There are also some work in using both active learning and constrained clustering framework. (Basu et al., 2004) proposed a method that, given a limited budget, determines what inputs we should get from an

Presented at the ICML 2011 Workshop on Combining Learning Strategies to Reduce Label Cost, Bellevue, Washington, USA.
oracle. But the algorithm does not consider maximizing gain from the available human/oracle resource. They propose two major phases in an active learning setting namely “Explore” and “Consolidate”. In the “Explore” phase, cluster centers are initialized and queries are used to make sure such that each cluster center is distinct. In the “Consolidate” phase, data points are added to the created centers and queries are used to make cluster assignments perfect. It seems many queries are used to find the distinct cluster centers only. This work is probably most suitable for a setting in which the number of clusters is not too large, and available human resources are sufficient to ensure good performance.

Xu et al in (Xu et al., 2005) proposed a constrained clustering technique based on spectral eigenvectors. This algorithm identifies crucial or near cluster boundary points. This is indeed useful, but their algorithm works for two cluster problems only. In (Huang & Lam, 2007), (Mallapragada et al., 2008), (Grira et al., 2005) and (Zhang & Wong, 2009), they have addressed a similar kind of problem, but none of them is explicit about how we can get the best use of human effort.

3. Algorithmic Details

3.1. Motivation

Although we want to seek pairwise constraints, as we discussed earlier, we can not ask for all possible pairs. If there are N images in a database there will be \(O(N^2)\) possible pairs and so asking humans to look at all possible pairs will be very tedious, expensive and time consuming. So a better option, is to use a clustering algorithm that will make use of some input from humans. That is why we aim to develop an algorithm where clustering is aided by pairwise constraints but the number of pairwise constraints required are minimized.

In this subsection, we would discuss two important questions, which are very relevant to this research work.

1. Which pairwise constraints are most important?

To illustrate this we showcase an example in figure 1. These are two real world cases (each row is a pair) where the algorithm thinks two leaves are different (after clustering without any user intervention) but if we look at them carefully we could say that they are similar. In both the cases our algorithm was not robust to the intrachuster variation. Since we build the features based on shapes only, the algorithm does not take into account the color or internal venation structure of leaves. But humans can use those as extra cues (however we should remember color of leaves change over the year). Also, the segmentation algorithm is good enough but still it is not 100% accurate, so sometimes the shape of segmented leaves is not as good as the original leaves. But humans are far better in distinguishing foreground from the background. So all of these human capabilities, which we are yet to replicate in computers, can be used along with algorithms to get much better results.

2. Which are the questions where human input is worth taking?

Another thing we have taken into account in our algorithm is to minimize human inputs. Suppose two points are very close to each other in the feature space (set A in figure 2), we know that the probability of them to be in the same cluster is high. Again if two points are very far apart (set C in figure 2), the probability that they will be from different clusters will be higher. So we realize asking humans about those pairs is not the most effective use of available resources. But we can ask about a pair which are not so close and also not so far, i.e., we are not sure if they are
from the same class or not (set B in figure 2). So we should ask about that pair.

We should also try to ask about pairs in which each element of the pair in feature space have dense neighborhoods. If they have, the information we get can also be useful regarding those neighbors. Suppose we get the information from a user that \(a\) and \(b\) are not similar. If \(a\) has data points \(a_1, a_2, a_3\) and \(b\) has \(b_1, b_2\) in their respective \(\epsilon\)-neighborhoods, we would know the chance of \(a_1\) and \(b_1\) or \(a_3\) and \(b_2\) being similar would be much less. But if \(a\) and \(b\) did not have any points in their \(\epsilon\)-neighborhoods, we can not infer anything about other pairs of points.

### 3.2. Algorithm

First we explain the high level idea of the algorithm and then provide pseudo code of the detailed algorithm.

Our algorithm is based on an idea that each original class consists of some smaller size clusters in feature space (which is often true in large real world databases). These small size clusters are easier to identify but the harder part is to find out which of these small clusters are part of the same class. In our approach, we rely on algorithms to find these smaller clusters and humans to merge them properly to build the correct ones. We start with the given data and over cluster them. Then each cluster is represented by its medoid and we try to merge them until we reach the actual number of clusters. This merging process is based on the similarity of two small clusters, more specifically the similarity of medoids of two smaller clusters, which is provided by humans, to the extent possible. If human input is not sufficient to produce the desired number of clusters, we revert to automatic clustering to complete the process.

We use Jaccard’s coefficient (Tan et al., 2005), along with some other metrics to measure the accuracy of our clustering algorithm.

\[
\text{Jaccard’s Coefficient (JCC)} = \frac{SS_n}{SS_n + SD_n + DS_n}
\]

Here \(SS\) is the number of pairs which truly belong in the same cluster and that are assigned to the same cluster. \(SD\) is which are originally in the same cluster but now they are in different cluster. \(DS\) is the number of pairs which belong in different clusters but are placed in same cluster.

Jaccard’s coefficient is calculated based on pairs of data. We also want to get pairwise human inputs so we are motivated to build an algorithm in which we greedily try to maximize the Jaccard’s coefficient at each step by asking a question and getting its correct answer. Let us explain this issue further.

Suppose, we have clusters \(C_j, j = 1, ..., K\) after we ask \(n\) questions. Let \(JCC_n\) and \(JCC_{n+1}\) be the Jaccard’s coefficient after we ask \(n\) and \(n+1\) questions respectively. Let \(N_1\) and \(N_2\) be the size of the clusters, which we merge after asking the \((n+1)\)-th question and getting a positive answer.

Now, whichever question we will ask after the \(n\)-th question we want to maximize \(|JCC_{n+1} - JCC_n|\) (if we assume that all of the clusters after the \(n\)-th question are homogeneous, \(JCC_{n+1} \geq JCC_n\)). So we could try to merge all possible pairs and see which of the mergings is giving us the best improvement over JCC. However, if we do that, computational burden is increased a lot (\(O(K^2)\) computation in each step). For example, if we start with say 248 number of clusters, we have to pair around 61000 possible pairs and find Jaccard’s coefficient in each case before we can ask the first question. Doing that much computation is not desirable. Moreover, we would also like to take account of the likelihood that a question will lead to an answer that could change the way we will cluster the data in the future. That is, an answer that can already be anticipated by the distance between images is less useful than one that is uncertain. We incorporate these goals in the following algorithm.

\[
JCC_n = \frac{SS_n}{SS_n + SD_n + DS_n}
\]
Large Scale Image Clustering with Active Pairwise Constraints

\[ JCC_{n+1} = \frac{SS_n + N_1 N_2}{(SS_n + N_1 N_2) + (SD_n - N_1 N_2) + DS_n} = \]

\[ SS_n + N_1 N_2 \]

\[ SS_n + SD_n + DS_n \]

So we can easily see that the maximum scope for improving the Jaccard’s coefficient is by asking a question that causes us to merge two clusters of larger size, where they are truly come from the same cluster.

Figure 3. \( P(icd) \) distribution(for \( icd \leq 2 \))(sky), how the positive answered(blue) and negative answered(red) inter-cluster distances are on number line and relative position of \( D_{ud} \)(black)

But we should remember that we can not try to merge large size clusters only without considering the distance between them. So we add another component in our algorithm, which is finding the uncertain pairwise distance \( (D_{ud}) \), i.e., the intercluster distance at which we are most uncertain if a pair will be similar or not. This component is to make sure we use human resources as well as possible. We adopt a staircase method (Rose et al.), which is a common approach in psychology for determining thresholds, to determine the initial \( D_{ud} \). One of our observations is that the chance of getting a “yes” answer is much less than getting a “no” answer, as there are plenty of possible pairs and only some of them are from same cluster.

The pseudo code of our algorithm (which we call CAC1, i.e., Constrained Active Clustering 1) is given later but we define some terms below which we have to use in the following discussion and also in the pseudo code:

We over cluster all the given data points to \( K = k * n \) number of clusters, where \( k \) is the original number of clusters and \( K \) is over clustering.

\( P(icd) \) : Probability distribution of all possible initial intercluster distances (which we know).
\( P_{yes}(icd) \) : Probability distribution of intercluster distances for which we get a yes answer (we gradually build this distribution)
\( P_{no}(icd) \) : Probability distribution of intercluster distances for which we get a no answer (we gradually build this distribution)
\( D_{ud} \) : \( \text{mode}(P(icd)) \) initially
\( NO_{yes}, NO_{no} \) : Number of yes and no answers respectively
\( Max\text{no}_{questions} \) : Maximum number of questions allowed

We build the distribution of all possible pairwise distances \( P(icd) \) for a given dataset after overclustering the data and don’t change that afterwards. We start from a \( D_{ud} \) based on the mode of that distribution. When we get a “no” answer we decrease the \( D_{ud} \) value by some small constant(\( \delta_n \)) and if we get “yes”, we increase the \( D_{ud} \) by even smaller constant(\( \delta_y \)). We continue this as long as we get a reasonable number of “yes” answers. This approach is very similar to a staircase approach. We wait till we get 10 “yes” answers for our dataset. After that we find the intercluster distances for which we found “no” answers and also for which we find “yes” answers. We find the means of those two sets (yes and no) of values and from those two means we update the \( D_{ud} \) value. Now, we start sampling pairs which have intercluster distance close to \( D_{ud} \) and also consider the size of the clusters as we discussed earlier. Based on a heuristic function (which we will describe) we ask questions and update \( D_{ud} \)’s value based on means of \( P_{yes}(icd) \) and \( P_{no}(icd) \).

In figure 3, we show the positively and negatively answered samples and the relative position of \( D_{ud} \) on the number line as we increase the number of asked questions.

As we have discussed before, we build a heuristic function, which for all possible pairs takes into account how close any pairwise distance \( (D_d) \) from \( D_{ud} \) is and also size of the two clusters. There might be many ways to combine these two components, however we weight two components and add them to build a function. The weights are tuned to improve results but not totally optimized.

\[ HF = w_1 * N_1 * N_2 + w_2 \frac{1}{1 + |D_d - D_{ud}|} \]
Large Scale Image Clustering with Active Pairwise Constraints

Algorithm 1 CAC 1

while NO\textsubscript{yes} ≤ 10 and NO\textsubscript{questions} ≤ Maxno\textsubscript{questions} do
  Find the pair with maximum HF (heuristic function) value and ask for it
  NO\textsubscript{questions} = NO\textsubscript{questions} + 1
  if yes then
    NO\textsubscript{yes} = NO\textsubscript{yes} + 1
    D\textsubscript{ud} = D\textsubscript{ud} + δ\textsubscript{y}
    Merge those pairs and update HF values
  else
    D\textsubscript{ud} = D\textsubscript{ud} − δ\textsubscript{n}
  end if
end while

{Now we have reasonable guesses of $P_{yes}(icd)$ and $P_{no}(icd)$}
Find $\mu_{yes}$ and $\mu_{no}$ from $P_{yes}(icd)$ and $P_{no}(icd)$
$D_{ud} = w \cdot \mu_{yes} + (1 - w) \cdot \mu_{no}$

while NO\textsubscript{questions} ≤ Maxno\textsubscript{questions} do
  Find the pair with maximum HF value and ask for it
  NO\textsubscript{questions} = NO\textsubscript{questions} + 1
  if yes then
    merge the pair
  end if
  Update $P_{yes}(icd)$, $P_{no}(icd)$ and HF
end while

Merge the remaining clusters using a simple greedy method

Here we briefly describe the greedy algorithm, which we use after we exhaust our human resource. We start with the smallest size cluster, find it’s distance with all other clusters and merge it to the nearest cluster for which the previous constraints, which we have already got, are not violated. We continue doing this until we have only $k$ clusters.

4. Results

To evaluate our algorithm, we have tried this on the leaf dataset, as we mentioned earlier. This dataset contains 1042 leaves from 62 species. We have lots of variation in leaves from same class and sometimes leaves have very similar shapes even if they are from different classes. First, we segment all of our leaves. From the segmented leaves we build curvature histograms at different scales and use them as features. Features are 525 dimensional, so we cluster in a 525 dimensional space.

If we cluster the leaf dataset into $k = 62$ clusters directly, on average we achieve Jaccard’s coefficient 0.2675. In our algorithm CAC1, after we exhaust our human inputs, we merge the remaining clusters greedily. We use $\delta_{y} = 0.01$, $\delta_{n} = 0.02$, $w_{1} = 1.2$, $w_{2} = 5$ and $w = 0.5$ for our dataset.

We also compare our approach with that of Active PCKMeans (Basu et al., 2004) (which we refer to as APCKMeans) in our dataset and our approach outperforms APCKMeans. The initialization step of finding cluster centers using the Explore method in APCKMeans takes around 10,000 queries on average. They have provided an upper bound on the number of queries in the Explore method in their paper, which says that for $k$ clusters we may need maximum $k^{2}$ queries. This is good for a small number of clusters, but for our problem we have 62 clusters and that makes the upper bound 117,242, which is way to high for practical purposes. In figure 4 we show a comparison of CAC1 algorithm with their algorithm and we see that for less than 6000 queries APCKMeans performance does not change a lot because all of those queries are mostly used for initializing the cluster centers using farthest first travel.

To evaluate our algorithm, we can vary two parameters: the initial number of clusters and also the number of questions allowed and see how our algorithm’s performance changes. For same number of initial clusters as the number of data points within 6000 queries we get perfect clustering with Jaccard’s coefficient 1. However our motivation was to study situations in which we have fewer queries. In those cases we will
start with fewer clusters and merge them. For 248 initial clusters within 1000 questions we reach Jaccard’s Coefficient of 0.5. In figure 4, we vary the initial number of clusters and show how the Jaccard’s coefficient changes. For less initial clusters, after some questions Jaccard’s coefficient can not improve further. This is because of the error which is introduced when we over-cluster initially. However starting with same number of clusters as the number of data points can help us to achieve perfect clustering.

Figure 4. Comparison of CAC1 for different initial number of clusters and with Active PCKMeans (Basu et al., 2004) algorithm

5. Conclusion and Future Work

In this paper, we present a heuristic approach for using human resource in image clustering in a better way. This is an initial approach to using limited human resources. The heuristic function and its weights are not totally optimized, but still give reasonably good results for our dataset. There are plenty of open issues to look at. First, we would like to find an algorithm which can be applied in a more general way and also experiment with other datasets. Also, if we have multiple users labeling at the same time, then it would be interesting to determine how we could ask for pairwise constraints with minimum information overlap. We will work on these problems as part of our future research.

References


