DISTRIBUTIONAL CLUSTERING FOR EFFICIENT CONTENT-BASED RETRIEVAL OF IMAGES AND VIDEO

Giridharan Iyengar
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598.
giyengar@us.ibm.com

Andrew B. Lippman
MIT Media Laboratory
Cambridge, MA 02139.
lip@media.mit.edu

ABSTRACT

In this paper, we present an approach to clustering images for efficient retrieval using relative entropy. We start with the assumption that visual features are represented by probability densities and develop clustering algorithms for probability densities (for example, normalized histograms are crude approximations of probability densities). These clustering algorithms are then used for efficient retrieval of images and video.

1. INTRODUCTION

Large image databases are becoming increasingly common and are being used for a variety of applications ranging from security to crop prediction to art history etc. These databases typically consist of thousands of images and represent gigabytes of storage. For efficient access into such large repositories, automatic indexing and retrieval techniques are currently being investigated in a number of research groups [1, 2, 3, 4, 5, 6].

The query-by-example (QBE) paradigm [5, 6] is a powerful approach for image and video retrieval because it makes no assumptions about the nature of content. QBE works well when the database is small. For large databases, presenting the user with a small randomly chosen subset of thumbnails is not a viable choice. Also, the notion of serendipitous discovery of the “right” image/video gets weak if the initial subset is not representative of the database. In addition, QBE does not characterize the database in any particular way. It searches the entire database every time a new query is presented. While there are efficient search techniques to reduce this search complexity, the problem of not having generalized and abstracted the data remains. Minka et al.[7] address the abstraction problem by allowing the user to provide relevance feedback and propagating newly learned labels. This can be viewed as the system having the ability of “short term memory”. This system achieves the happy middle ground between pure query-by-example with no assumptions at one end and Bayesian modeling with structural assumptions at the other end of the spectrum[8].

1.1. Why clustering?

We argue that clustering is another such middle ground. Clustering is a natural way to organize massive quantities of data and build interfaces that take advantage of the cognitive abilities of the user. As the database grows in size, clustering becomes a natural solution for presenting the user with a small representative subset of the database. As opposed to showing a few randomly chosen images/video clips from the database to start off a query-by-example search, chosen images/video from each cluster can be used as the starting point. This process can be repeated ad infinitum with hierarchical clustering. This ensures that the user is presented with a representative collection at each query.

A good choice of representation for clustering should not compromise the ability to handle higher-level characterization. In particular, if the clustering is performed in the space of probabilistic representations, we retain the advantage of using higher-level probabilistic descriptors when available, such as those in Ref.[8]. In this paper, we work with discrete probability density representations of image features. A first choice of such a representation is the normalized histogram. Histograms are widely used in image databases because they outperform retrieval with other techniques based on shape and spatial composition both in terms of efficiency and robustness. Moreover normalized histograms can be made invariant to image size, translation, rotation and scale and are compact in size.

2. DISTRIBUTIONAL CLUSTERING

We describe two clustering algorithms in this paper, namely soft top-down and soft bottom-up and compare computational complexity and retrieval performance with our earlier work on hard partitional clustering[9, 1]. The hard clustering algorithm[9] assigns each sample density into one cluster. While this is useful, in many cases it is desirable to have a soft assignment, i.e., have samples belong to multiple classes and have a probabilistic association with each class. Real world examples can have multiple labels associated with them and it is desirable that the clustering algorithms exhibit similar behavior. The soft assignment process is desirable for yet another reason: The hard clustering algorithm is an iterative procedure with the potential of getting locked in a local minima. The soft assignment algorithm can be viewed as a technique for selecting a better local minima. The deterministic annealing algorithm on which our soft top-down algorithm is based was proposed as a solution for the same[10].

We assume that we have a discrete density representation of features for each image/video in our database. In this density space, the Kullback-Liebler (KL) divergence becomes a natural choice for comparing similarities. We note that KL divergence is asymmetric and therefore results in two cost criteria and correspondingly two distinct cluster solutions. We begin with the notion that each sample point (in our case, a point is a probability distribution) is associated in probability with each cluster. Hard clustering becomes a special case where the association probabilities are either 1 or 0. We quote from Ref.[10] the following which is true for all clustering (distributions or otherwise).

For a given set of centroids, the expected distortion is
\begin{align}
\langle D \rangle &= \sum_{j=1}^{N} \sum_{k=1}^{K} P(q_j \in C_k) d(q_j, p_k) \\
\end{align}

where the \( q_j \)'s are the sample points, with \( N \) such points and the \( p_k \) 's are the centroids with \( K \) such centroids, each cluster denoted by \( C_k \). The measure \( d(q_j, p_k) \) is the distortion for representing sample \( q_j \) by centroid \( p_k \). Under the assumption that we have no prior knowledge about the data, it is reasonable to apply the principle of maximum entropy. That is, given the different configurations a system can be in, the most likely configuration is the one with maximum entropy. It is relatively easy to show that maximizing the entropy under (1) results in the Gibbs distribution.

\[
P(q_j \in C_k) = \frac{e^{-\beta d(q_j, p_k)}}{Z_{q_j}}
\]

where \( Z_{q_j} \) is the partition function

\[
Z_{q_j} = \sum_k e^{-\beta d(q_j, p_k)}
\]

It is also reasonable to assume that the probabilities relating the different samples to clusters are independent. Hence the total partition function is \( Z(q) = \prod_j Z_{q_j} \). The parameter \( \beta \) can be considered inversely proportional to temperature, given the analogy of annealing. Minimizing the distortion subject to maximizing the entropy is equivalent to minimizing the Gibbs free energy which is defined as

\[
F(p) = -\frac{1}{\beta} \log Z(q)
\]

The set of centroids \( p \), that minimize the free energy satisfy \( \frac{\partial}{\partial p_k} F = 0 \) which results in

\[
\sum_j P(q_j \in C_k) \frac{\partial}{\partial p_k} d(q_j, p_k) = 0
\]

While this technique works with any distortion function, It can additionally be shown that for convex distortion functions, there is a unique solution that minimizes \( \sum_j d(q_j, p) \) [10]. That is, at \( \beta = 0 \) there is exactly one cluster. At \( \beta > 0 \), a set of vectors corresponding to a local minima in free energy can be obtained. We now apply this to top-down clustering of discrete densities. Later (in section 2.2), we present a simplifying approximation which results in a computationally simpler bottom-up clustering.

### 2.1. Soft top-down clustering

Using densities places the additional constraint that each resulting centroid solution be a valid distribution. The additional constraint leads to the following minimization formulation [1].

\[
\frac{\partial}{\partial p} F(p) = \frac{\partial}{\partial p} \left( -\frac{1}{\beta} \log Z(q) + \lambda(1 - \sum_i p) \right) = 0
\]

To account for the asymmetry of KL divergence, we get the following two expressions for the cumulative distortion.

\[
J = \sum_j \sum_k (P(q_j \in C_k) KL(q_j || p_k) + \lambda(\sum_i p_k(i) - 1)
\]

where \( P(q_j \in C_k) \) is as defined in (Eq. 2), and

\[
J = \sum_j \sum_k (P(q_j \in C_k) KL(p_k || q_j) + \lambda(\sum_i p_k(i) - 1)
\]

Minimizing Eq. 7 with respect the centroid densities, we get a centroid expression

\[
p_k(i) = \frac{\sum_j P(q_j \in C_k) q_j(i)}{\sum_j P(q_j \in C_k)}
\]

and minimizing Eq. 8, we get a centroid expression

\[
p_k(i) = C \times \prod_j q_j(i)^{P(q_j \in C_k)}
\]

where \( C \) is the constant introduced to satisfy the Lagrange constraint. We note here that in one case, we get the weighted arithmetic mean as the centroid expression and the weighted geometric mean in the second case.

The top-down clustering algorithm can now be detailed in the following steps, with the appropriate expression for centroid estimation:

1. Set \( \beta = 0 \) and find the global solution.
2. Replace each centroid in the solution set with many copies, each copy randomly perturbed. Increase \( \beta \) per annealing schedule.
3. Repeat probabilistic assignment and centroid computation till convergence. This is similar to the EM algorithm.
4. Discard duplicate centroids (i.e., the ones that converge to the same point). This is the solution set at the new \( \beta \).
5. Repeat from step 2 till desired number of clusters are reached.

### 2.2. Soft bottom-up clustering

The above algorithm is computationally inefficient since minimization has to be repeated over the entire dataset for each change in \( \beta \) value. If the dataset is large, this computation can be prohibitively expensive. Ideally, if one level of the hierarchy can be related to the next level without reference to the underlying data, then the resulting algorithm will be computationally efficient since we deal with (a possibly) much smaller collection of clusters at each level as opposed to the entire dataset. We attempt to do this with bottom-up clustering.

The clustering model we use in this formulation is shown in Eq. 11. Since the quantities we are dealing with are distributions, we assume that \( X \) represents the underlying random variables that these distributions pertain to.

\[
P(X) = \sum_{i=1}^{c_i} \pi^i p(X | z_i^l = 1, M_i)
\]

where \( l \) is the level in the hierarchy the model represents, with \( l = 0 \) being the coarsest level. \( M_i \) is the model at level \( l \), \( C_i \) the number of centroids at that level and \( \pi^i \), the corresponding priors. The variable \( z_i^l \) is an indicator that takes the value 1 iff the sample \( X \) is drawn from the \( i \)th component. This model is a standard likelihood model in the EM literature [11].

At each level, the collection of centroid distributions from the previous level is treated as data for estimating the new model. In addition, at each level the clustering is soft. A straightforward implementation would be to draw some samples from the model at level \( l + 1 \) and simply run the EM algorithm to estimate the new model. This would be computationally expensive and be contrary to our goal. Let us assume that instead of generating real samples from the cluster model at level \( l + 1 \), we perform a Monte
Figure 1: Sample images in the test database.

Carlo simulation and extract a collection of virtual samples. Once these virtual samples are generated, we can find a compact expression for relating the two levels that is independent of the virtual samples themselves. The mechanics of this proof is skipped in this paper and the reader’s attention is drawn to Ref. [1, chapter 4]. The idea of using virtual samples for estimating EM parameters was proposed for learning mixture parameters by Vasconcelos [12]. We have shown[1] that we get the following update equation, considering the Lagrange constraint.

\[ p_i^k(l) = \frac{\sum h_{ij} N_k p_{ij}^{k+1} (l)}{\sum h_{ij} N_k} \]  \hspace{1cm} (12)

Which is very closely related to the update equation (9) in the top-down soft clustering algorithm. Instead of a simple weighted average, we now have the additional weighting factor that is dependent on the number of virtual samples that we create. The responsibility \( h_{ij} \) is given as

\[ h_{ij} = \frac{\pi_i e^{-D(p_i^{j+1} \mid \hat{l} p_j^k)}}{\sum_k \pi_k e^{-D(p_i^{j+1} \mid \hat{l} p_k^k)}} \]  \hspace{1cm} (13)

We note here that because of the directional constraint in our virtual sample generation, a particular form of the KL divergence gets chosen and gives us only one expression for the centroid update equation.

3. EXPERIMENTS

We now present some experimental results with a 1000 sample images chosen from the Corel photo database. Figure 1 shows some sample images in the database. The experiments that we perform are two-stage query experiments. That is, the test image is first queried with respect to the cluster centroids using KL divergence as the ranking criterion. The closest cluster that is returned in this first step is now considered as the database and the images within cluster are ranked using KL divergence. In the tables, Rank 1 implies that the correct match came up as the first match in this 2-stage retrieval. Rank 5 implies that the correct match appeared within the top 5 matches and similarly Rank 20 implies that the correct match appeared in the top 20 matches. In addition, to provide a baseline, the same query experiment is performed with the entire database. It is easy to show that using KL divergence for retrieval is equivalent to performing maximum-likelihood [1]. In our tables, the Hard AM and Hard GM refer to the hard versions of these clustering algorithms presented earlier [9, 1].

In each set of experiments, the query image is subject to a different distortion prior to query. The distortions that we used are: scaling, sub image, rotation, contrast shift, gaussian noise addition and random pixel distortion. In the scaling experiment, the query image is randomly scaled either 0.5 or 2.0 times the original image dimensions. For sub-image experiments, the top \( \frac{2}{3} \) of the image is used to perform the query. In rotation experiments, an affine warp of 45 degrees is applied to the query images. For contrast shift, the dynamic range of the pixels is reduced to 75% of the original.

Figure 2: Sample Rotated and Noisy Images. Bottom right image is noise-distorted.

In the next two experiments, we distort the query images by a large extent. For the contrast and illumination change experiment, we reduce the dynamic range of the pixels to 75% of the original and for the gaussian noise experiment, we add a gaussian distributed random number to every pixel in the image. Table 3 shows the re-

Figure 3: Sample image and its 25% sub-image used for query.
results of the contrast change experiment. We note here that we used the Ohta color space and did not normalize to make them illumination invariant. We therefore expect a considerable degradation in retrieval accuracy. Table 4 shows the results for the gaussian noise experiment.

We find that compared to using the entire database, there is a small drop in performance when limiting the search to only the nearest cluster. However, the time savings of a cluster based query are significant. Each cluster based query requires approximately \( \frac{1}{N_c} \times \% \) of the computation time, where \( N_c \) is the number of clusters. In general, the soft clustering algorithms perform best. Interestingly, the percentage of correct retrievals in Rank 1 is higher for the clustered retrieval compared with normal retrieval in Tables 3 and 4. This is possibly because of fewer distractors in the clustered cases and therefore less confusion. However, for Rank 20 performance, the full database query always performs better. This implies that if the correct cluster is identified at the first stage, then having fewer distractors results in a better rank. However, there are a number of instances where the correct clusters are not identified in the first stage and therefore we do not get similar improvements in Rank 20. To a smaller degree, this effect can be observed in Rank 5 performance.

### 4. SUMMARY

Both the soft algorithms performed better than the Hard variants. However, both these algorithms are significantly more computationally complex than the Hard variant. Using soft algorithms to initialize a hard clustering as we did for these experiments is perhaps not the best way to use these algorithms. While such an approach improves the hard clustering, the significant additional computation may not justify the improvement in performance. If we use the soft clusterings without freezing, there might be additional improvement in performance. However, it does not result in computational savings during retrieval since in soft clustering every sample belongs to every cluster (with only the degree of belongingness changing). Soft clusterings as such only influence the order of search and do not restrict the search set. This is a good property to have – e.g., if the user is willing to wait longer (but not as long as it takes to search the entire database), these soft clusterings can be used to search the database in the order of decreasing likelihood, thereby reducing the expected time retrieval. A possible way to achieve the computational savings with soft clustering is to limit the search set to a fixed percentage of the database images. Since each cluster in the soft algorithms implies a different search order, this thresholding strategy might offer the computational savings with possibly higher retrieval accuracy.

### 5. REFERENCES


