Image Database Retrieval with Multiple-Instance Learning Techniques

Cheng Yang*
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
yangc@cs.stanford.edu

Tomás Lozano-Pérez†
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
tlp@mit.edu

Abstract

In this paper, we develop and test an approach to retrieving images from an image database based on content similarity. First, each picture is divided into many overlapping regions. For each region, the sub-picture is filtered and converted into a feature vector. In this way, each picture is represented by a number of different feature vectors. The user selects positive and negative image examples to train the system. During the training, a multiple-instance learning method known as the Diverse Density algorithm is employed to determine which feature vector in each image best represents the user's concept, and which dimensions of the feature vectors are important. The system tries to retrieve images with similar feature vectors from the remainder of the database. A variation of the weighted correlation statistic is used to determine image similarity. The approach is tested on a medium-sized database of natural scenes as well as single- and multiple-object images.

1. Introduction

While searching for textual data on the World Wide Web and in other databases has become common practice, search engines for pictorial data are still rare. This comes as no surprise, since it is a much more difficult task to index, categorize and analyze images automatically, compared with similar operations on text.

An easy way to make a searchable image database is to label each image with a text description, and to perform the actual search on those text labels. However, a huge amount of work is required in manually labelling every picture, and the system would not be able to deal with any new pictures not labelled before. Furthermore, it is difficult to give complete descriptions for most pictures. Consider the picture in Figure 1. One might be tempted to describe it as "river, trees and stones", but it would not be able to respond to user queries for "water", "waterfall", "clouds" or "white blobs in background". To make a real content-based image retrieval system, we need some mechanism to search on the images directly.

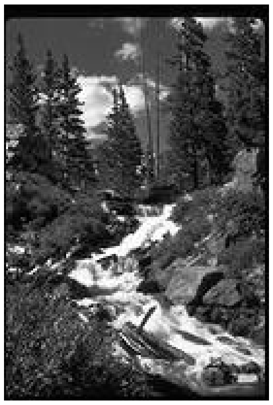


Figure 1. A sample picture

1.1. Previous work

Early approaches to the content-based image retrieval problem include the IBM QBIC (Query-By-Image-Content) System [3], where users can query an image database by average color, histogram, texture, shape, sketch, etc. The image database is preprocessed with some human assistance to facilitate the search. Some research has been done to group images into categories that capture high-level concepts, such as indoor vs. outdoor scenes and city vs. landscape scenes [18, 19]. However, image queries along these lines are not powerful enough, and more complex queries (such as "all pictures that contain waterfalls") are hard to formulate. Lipson *et al.* [9] used hand-crafted

*Current address: Computer Science Department, Stanford University, Stanford, CA 94305, USA.

†Current address: MIT AI Lab, 545 Technology Square, Cambridge, MA 02139, USA.

templates to classify natural scene images. While it has been successful in this domain, the process is difficult to automate. Recent research has paid more attention to query-by-example [1, 11, 16]. In these systems, user queries are given in terms of positive and negative examples, and sometimes salient regions are also manually indicated. The system then proceeds to retrieve images "similar" to the positive examples and "dissimilar" to the negative ones.

For images, however, "similarity" is not well-defined. Many algorithms have been proposed to compute image similarities. They typically do so by converting images into feature vectors and using feature vector distances as a similarity measure. Grosky and Mehrotra [4] experimented with a representation using object boundaries' local structural features, and they used string edit-distance as a distance measure. Mehrotra and Gary [13] used relative positions of "interest points" along object boundaries to represent shape, and used Euclidean distance as a distance measure. These methods are based on object recognition techniques. However, they are quite sensitive to noise in the images, and cannot handle images where there are no distinct objects, as in natural scenes. De Bonet and Viola [1] proposed an algorithm where images are passed through a tree of nonlinear filters to obtain feature vectors that represent "texture-of-texture" of the original images. It works well with natural scenes and single-object test sets. Maron and Lakshmi Ratan [11] used simple features like a row's mean color, color differences and color distributions among neighbors, etc., and it works well for color images of natural scenes. Ravela *et al.* [16] developed a system that uses a correlation measure to indicate similarity. It works for a variety of images, but it requires that the user manually pick the regions of interest from the images. In reality, the user may not always know which regions are most important with respect to the similarity measure used by the algorithm. Furthermore, not all pixels within a rectangular region are of equal interest, which complicates the problem.

More detailed reviews of previous literature in image classification and retrieval can be found in [8, 11].

1.2. The multiple-instance learning approach

Since the picture in Figure 1 can be viewed differently as "river", "waterfall", "trees", "clouds", etc., and multiple-object images are more common than single-object images, it is natural to have one image correspond to more than one feature vector, each one describing one particular view (or object). In this way, each positive or negative example translates into multiple feature vectors. After Maron [10], we call each of these feature vectors an *instance*, and we call the collection of instances for the same example image a *bag*.

For a positive example, at least one of the instances in the bag should be a close match to the concept the user had in mind when he or she chose the examples, but we do not know which one. The rest of the instances in the bag are irrelevant and should be regarded as noise. For a negative example, we know for sure that none of the instances in the bag corresponds to the user's concept. Given the large collection of instances from positive and negative examples, our task is to find the "ideal" feature vector that accounts for the user's concept. Furthermore, among all the feature dimensions we choose to describe an instance, only some of them may be relevant to defining the ideal concept, while the remaining ones should be ignored. When we take this into account, the problem becomes computationally much harder [2].

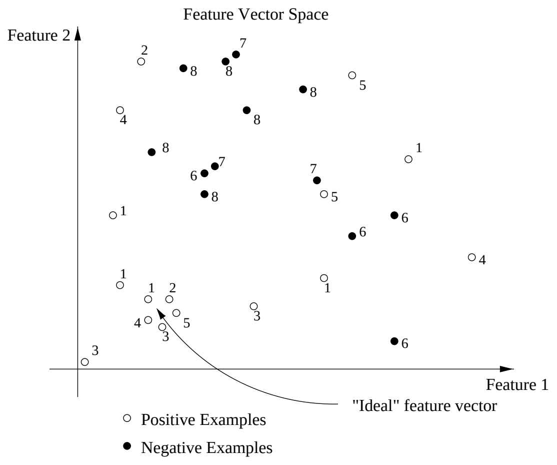


Figure 2. A multiple-instance learning algorithm: Diverse Density

This kind of problem is known as a *Multiple-Instance Learning* problem [2, 10, 12]. One way to solve this type of problem is to examine the distribution of these instance vectors, and to look for a feature vector that is close to a lot of instances from different positive bags and far from all the instances from negative bags. Such a vector is likely to represent the concept we are trying to learn. This is the basic idea behind the *Diverse Density* algorithm, proposed by Maron and Lozano-Pérez [10, 11, 12], and is illustrated in Figure 2. In Figure 2, there are five positive examples (bags) labelled 1 to 5 and three negative examples (bags) labelled 6 to 8. Each bag has several instances. The feature vector space is 2-dimensional. The "ideal" feature vector is where there is a high concentration of positive instances from different bags.

Maron and Lakshmi Ratan [11] have applied the Diverse Density technique to image retrieval problems by using image features such as color statistics and color distribution patterns. They converted each picture into an 8×8 matrix of "color blobs", and used feature vectors such as row color vectors, column color vectors, color patterns of specific neighborhoods, etc. This representation works well for retrieving color natural scene images.

In this paper, we improve their method to deal with a broader range of images including object images. Similar attempts have been made by Lakshmi Ratan *et al.* [6], who used image segmentation techniques and more complex features such as a combination of color, texture and simple shapes. They used special filters to find circles and other simple shapes from the images, in an attempt to learn object concepts. Our approach differs from theirs in that we do not pre-define or depend on any templates for object shapes, but use a much simpler feature representation. For object images that can be modeled by existing templates, the method given in [6] is more suitable, but our method can potentially work with a larger class of images.

We define an image similarity measure as the correlation coefficient of corresponding regions after smoothing and sampling, and further refine it by allowing different weight factors for different locations when comparing for similarity. Based on this, we develop a feature vector representation for images where we can use weighted Euclidean distance to reflect the distance defined by our weighted similarity measure. For each example image, a bag of multiple instances are obtained by choosing different sub-regions of the image and generating a feature vector for each region.

In section 2 below, we introduce the Diverse Density algorithm. Section 3 discusses our correlation similarity measure, its corresponding feature representation and a weight factor controlling method. Section 4 gives experimental details and results.

2. The Diverse Density algorithm

In this section, we give a brief introduction to the multiple-instance learning problem and the Diverse Density (DD) algorithm. A more elaborate treatment can be found in [10, 11, 12].

2.1. The multiple-instance learning problem

Machine learning algorithms provide ways for computer programs to improve automatically with experience [14]. In a typical machine learning problem, the task is to learn a function

$$y = f(x_1, x_2, ..., x_n)$$

given some examples. In traditional *Supervised Learning*, the examples are given in terms of $(y_i, x_{i1}, x_{i2}, ..., x_{in})$ tuples, where i is the index of examples: $i = 1, 2, 3, ...$ That is, each set of input values $(x_{i1}, x_{i2}, ..., x_{in})$ is tagged with the correct label y_i. In *Multiple-Instance Learning*, however, input vectors $(x_{i1}, x_{i2}, ..., x_{in})$ (called *instances*) are not individually labelled with its corresponding y_i value; rather, one or more instances are grouped together to form a *bag*, and they are collectively labelled with a y value of 1 (TRUE) or 0 (FALSE). If the label is TRUE, it means that at least one of the instances in the bag must correspond to y_i=TRUE, while others may correspond to either TRUE or FALSE. If the label is FALSE, it means that all of the instances in the bag must correspond to FALSE.

In terms of the image retrieval problem, each positive example selected by the user corresponds to a bag labelled TRUE, and each negative example selected by the user corresponds to a bag labelled FALSE. A feature vector consists of n numbers (features), each of which partially describes the image in some way, for example, pixel values, color statistics, edge locations, etc. Redundant or irrelevant features are allowed. Since the pictures are inherently ambiguous, we generate more than one feature vector (instance) to describe each picture. We expect that one of these feature vectors for each positive example would account for the concept the user had in mind when picking the examples, and that none of them in the negative examples would coincide with the user's concept.

We would like to train the system so that it can make predictions for new examples: given a new example image (a bag of instance vectors), it should determine whether it corresponds to TRUE or FALSE. To allow for uncertainty, the system may give a real value between 0 (FALSE) and 1 (TRUE).

We make a simplifying assumption that the user's concept can be represented by a single "ideal" point in the n-dimensional feature space. A bag is labelled TRUE if one of its instances is close to the ideal point. A bag is labelled FALSE if none of its instances is close to the ideal point. The "ideal" point is where there is a high concentration of positive instances from different bags. The confidence of a bag being TRUE can be measured by the distance from the ideal point to the closest instance vector in the bag. [10, 12] developed an algorithm called *Diverse Density*, which is able to find such a point. Not all dimensions of feature vectors are equally important, so the distance here is not restricted to normal Euclidean distance, but may be defined as a weighted Euclidean distance where important dimensions have larger weights. The Diverse Density algorithm is capable of determining these weight factors as well.

2.2. Diverse Density

Following the same notations as in [10, 11, 12], we denote the positive bags as $B_1^+, B_2^+, ..., B_n^+$ and the negative bags as $B_1^-, B_2^-, ..., B_m^-$. The j^{th} instance of bag B_i^+ is written as B_{ij}^+, while the j^{th} instance of bag B_i^- is written as B_{ij}^-. Each bag may contain any number of instances, but every instance must be a k-dimensional vector where k is a constant.

Not all k dimensions contribute equally to defining the ideal concept, so we need to give a weight to each dimension. We want to look for a point in the weighted k-dimensional space near which there is a high concentration of positive instances from different bags. It is important that they are from *different* bags, since a high concentration of instances from the same bag is effectively the same as one instance at that point. In other words, we are looking for a point where there is a high *Diverse Density* of positive instances.

For any point t in the feature space, the probability of it being our target point, given all the positive and negative bags, is $\Pr(t|B_1^+, ..., B_n^+, B_1^-, ..., B_m^-)$. So the point we are looking for is the one that maximizes this probability, that is

$$\arg\max_t \Pr(t|B_1^+, ..., B_n^+, B_1^-, ..., B_m^-)$$

Using Bayes' rule, assuming a uniform prior over the concept location $\Pr(t)$ and conditional independence of the bags given the target concept t, the above equals

$$\arg\max_t \prod_i \Pr(t|B_i^+) \prod_i \Pr(t|B_i^-)$$

This is a formal definition of maximizing Diverse Density. We use the "noisy-or" assumption (see Maron [10] for motivation and discussions) that

$$\begin{aligned} \Pr(t|B_i^+) &= 1 - \prod_j (1 - \Pr(B_{ij}^+ = t)) \\ \Pr(t|B_i^-) &= \prod_j (1 - \Pr(B_{ij}^- = t)) \end{aligned}$$

and make the following assumption:

$$\Pr(B_{ij} = t) = \exp(-||B_{ij} - t||^2)$$

where $||B_{ij} - t||$ is the distance between the two vectors. This is a Gaussian bump centered on the feature vector. As we mentioned before, not all dimensions are equally important, so we define the distance to be a weighted Euclidean distance:

$$||B_{ij} - t||^2 = \sum_k w_k^2 (B_{ijk} - t_k)^2$$

where B_{ijk} is the k^{th} dimension in the vector B_{ij}. w_k^2 is a non-negative weight. (We use w_k^2 rather than w_k in order to force the weights to be non-negative.) Now we need to maximize Diverse Density over both t and w. By introducing weights, we have actually doubled the number of dimensions over which we are trying to maximize Diverse Density.

2.3. Finding the maximum

The problem of finding the global maximum Diverse Density (DD) is difficult, especially when the number of dimensions is large. The DD algorithm makes use of a gradient ascent method with multiple starting points. It starts from every instance from every positive bag and performs gradient ascent from each one to find the maximum. The idea is that, at least one of the positive instances is likely to be close to the maximum. So if we do hill-climbing from every positive instance, it is very likely that we will hit the maximum DD point.

3. Adapting Diverse Density

The definition of DD requires the definition of feature vectors for images, where a weighted Euclidean distance can be used as a measure of "similarity". We want to use the pixels themselves as the features and correlation coefficient as a similarity measure.

3.1. The correlation similarity measure

Given two series of sampled signals $f_1(t)$ and $f_2(t)$, $t = 1, 2, ..., n$, there is a standard way to find out how correlated they are with respect to each other: we can compute their *correlation coefficient* [17]. In its simplest form, the correlation coefficient r is defined by

$$r = \frac{\frac{1}{n}\sum_{t=1}^{n}(f_1(t) - \overline{f_1})(f_2(t) - \overline{f_2})}{\sigma_{f_1}\sigma_{f_2}}$$

where $\overline{f_1}$, $\overline{f_2}$ are the average values of $f_1(t)$ and $f_2(t)$, and σ_{f_1}, σ_{f_2} are the standard deviations of $f_1(t)$ and $f_2(t)$, respectively:[1]

$$\overline{f_1} = \frac{1}{n}\sum_{t=1}^{n} f_1(t), \quad \overline{f_2} = \frac{1}{n}\sum_{t=1}^{n} f_2(t)$$

$$\sigma_{f_1} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(f_1(t) - \overline{f_1})^2}, \quad \sigma_{f_2} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(f_2(t) - \overline{f_2})^2}$$

When $r = 1$, the two signals are perfectly correlated. When $r \approx 0$, there is little or no correlation between the two. When $r = -1$, the two signals are perfectly inversely correlated. If we only count positive correlations as "similar", then r can be used as a direct measurement of similarity: as r increases, similarity increases.

[1] Strictly speaking [15], in the definitions for σ_{f_1}, σ_{f_2} and r given here, $\frac{1}{n}$ should be replaced by $\frac{1}{n-1}$. But it does not matter to us. Both definitions work the same way in the derivations in this paper, and we choose to use $\frac{1}{n}$ which is more convenient.

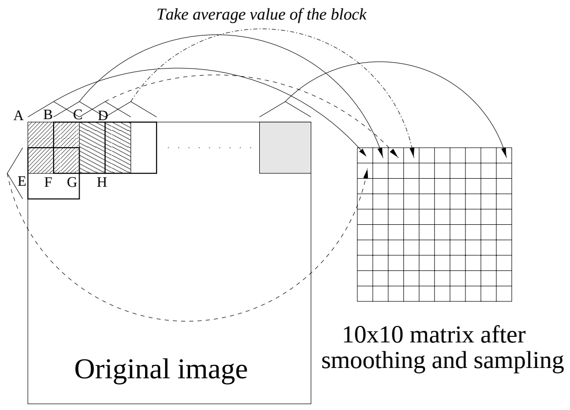

Take average value of the block
A
B
C
D
E
F
G
H
Original image
10x10 matrix after smoothing and sampling

Figure 3. Illustration of smoothing and sampling process

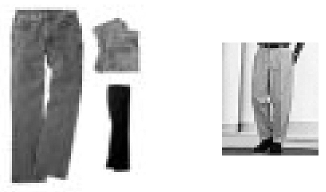
Since an image region of size $u \times v$ can be treated as an uv-dimensional vector of gray-scale values, this correlation coefficient is often used to measure similarities between image regions [5, 16].

If we apply the correlation formula to the original images directly, on a pixel-by-pixel basis, a shift in the image by one pixel would cause a relatively big change in the correlation value, which is not desirable. To avoid this effect, we smooth and sample the $m \times n$ image down to a low-resolution $h \times h$ matrix. In most of the experiments in this paper, we choose $h = 10$. Specifically, we smooth the $m \times n$ image with a $\frac{2m}{h+1} \times \frac{2n}{h+1}$ averaging kernel and then sub-sample it to get an $h \times h$ matrix. In other words, each entry in the resulting $h \times h$ matrix is the average gray-scale value of a corresponding block region in the original image, as illustrated in Figure 3. In Figure 3, the average value of block $AEGC$ goes into the 1st entry of the 10×10 matrix, the average value of block $BFHD$ goes into the 2nd entry (1st row, 2nd column) of the matrix, and so on. Each block has a 50% overlap with any of its neighbors. The large overlap is intended to reduce sensitivity to the choice of block border locations.

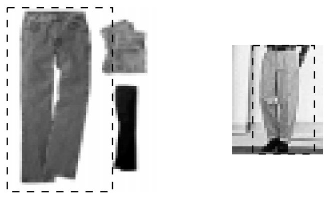
3.2. Region selection

With the above smoothing and sampling scheme and $h = 10$, the correlation coefficient is a good indication of similarity for two single-object images. However, this would not generalize to more complex cases such as multiple-object images, where the object (or feature) of interest may not be at the same position in all pictures.

In a more complex image, the object (or feature) of interest does not occupy the whole image, but only a sub-region of the image. We would not be able to get satisfactory results if we compared the two entire images in Figure 4(a) using the correlation similarity measure, but we may have better luck if we compare a region in one image against a region in the other. For example, the correlation coefficient of the two entire images in Figure 4(a) is 0.118, while the correlation coefficient of the two marked regions in Figure 4(b) is 0.674, indicating similarity.



(a) Correlation coefficient of these two images is 0.118.



(b) Correlation coefficient of the two marked regions is 0.674.

Figure 4. More complex images

Now the question is, how do we choose the regions? In fact, we do not know which regions we should pick, since the pictures are inherently ambiguous, and any region might become the region of interest, depending on the user's concept. This is exactly where multiple-instance learning can help us: we can simply pick all possible regions and let the learning algorithm take care of finding the “right” region for us.

Figure 5 shows 20 possible regions (as shaded areas). Conceptually, there is an unlimited number of possible regions. When deciding the actual number of regions to consider, there is a trade-off between the chance of hitting the “right” region and the amount of noise introduced. This will be discussed further in Section 4.2.

In most of this paper, we only consider the 20 possible regions shown in Figure 5. Consequently, if some images contain objects that are much smaller than one quarter of the entire image, their details may not be visible to the algorithm. For each region, we consider both the original image in that region and the left-right mirror image of that region, since left-right mirror images occur very frequently in image databases and we would like to regard them as

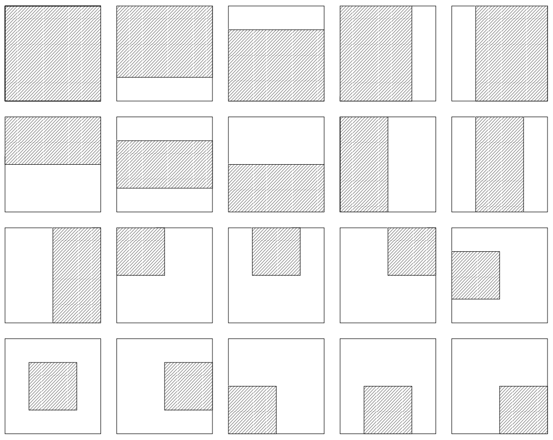


Figure 5. Possible regions to consider

the same. Therefore, there are a total of 40 sub-pictures to consider. This translates into 40 instances per bag in the multiple-instance learning framework. Here, we do a little optimization to throw out regions whose variances are below a certain threshold, since low-variance regions are not likely to be interesting. For each sub-picture, we process it with smoothing and sampling as illustrated in Figure 3, to get an $h \times h$ matrix which we treat as an h^2-dimensional feature vector.

3.3. Weighted correlation coefficient

Not all dimensions in the feature vector are equally important. For example, some of them may correspond to the background in the image, and we do not want them to carry the same weights as other dimensions. Therefore, we extend our correlation similarity measure to allow different dimensions to have different weight factors. We define a weighted correlation coefficient for two n-dimensional feature vectors f_1 and f_2 as:

$$r' = \frac{\frac{1}{n}\sum_{k=1}^{n} w_k^2 (f_1(k) - \overline{f_1})(f_2(k) - \overline{f_2})}{\sigma'_{f_1}\sigma'_{f_2}}$$

where w_k^2 is the non-negative weight for the k^{th} dimension, $\overline{f_1}$, $\overline{f_2}$ are defined as before, and σ'_{f_1}, σ'_{f_2} are the "weighted" standard deviations of $f_1(k)$ and $f_2(k)$, respectively:

$$\sigma'_{f_1} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} w_k^2 (f_1(k) - \overline{f_1})^2}$$

$$\sigma'_{f_2} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} w_k^2 (f_2(k) - \overline{f_2})^2}$$

3.4. Fitting into Euclidean space

Our similarity measure is defined as the weighted correlation coefficient on feature vectors, rather than Euclidean distance. This does not fit directly into the Diverse Density framework. However, there is a simple way to transform the vectors, so that we can use weighted Euclidean distance directly to reflect the weighted correlation coefficients of the original feature vectors.

Suppose that A_{ij} is the n-dimensional feature vector we have obtained for the i^{th} bag, j^{th} instance. w_k^2 is the weight factor for the k^{th} dimension. Define

$$B_{ij} = \frac{A_{ij} - \overline{A_{ij}}}{\sigma'_{A_{ij}}}$$

where $\overline{A_{ij}}$ is the average of A_{ij} entries, and $\sigma'_{A_{ij}}$ is the "weighted" standard deviation of A_{ij} entries:

$$\overline{A_{ij}} = \frac{\sum_{k=1}^{n} A_{ijk}}{n}, \quad \sigma'_{A_{ij}} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} w_k^2 (A_{ijk} - \overline{A_{ij}})^2}$$

With this definition, we are going to show that, comparing or ranking A_{ij} vectors based on weighted correlation coefficients is the same as comparing or ranking B_{ij} vectors based on weighted Euclidean distances in reverse order. This is formally stated as follows:
Claim For any i, j, l, m, p, q, u, v and weight factors $\{w_k^2\}$,

1. $Corr(A_{ij}, A_{lm}) > Corr(A_{pq}, A_{uv})$ if and only if $||B_{ij} - B_{lm}|| < ||B_{pq} - B_{uv}||$
2. $Corr(A_{ij}, A_{lm}) = Corr(A_{pq}, A_{uv})$ if and only if $||B_{ij} - B_{lm}|| = ||B_{pq} - B_{uv}||$
3. $Corr(A_{ij}, A_{lm}) < Corr(A_{pq}, A_{uv})$ if and only if $||B_{ij} - B_{lm}|| > ||B_{pq} - B_{uv}||$

where $Corr(\alpha, \beta)$ means the weighted correlation coefficient of α and β, and $||\alpha - \beta||$ means the weighted Euclidean distance between α and β.
Lemma For any i, j,

$$\sum_{k=1}^{n} w_k^2 B_{ijk}^2 = n$$

Proof of the lemma is straightforward given the definitions above.
Proof of Claim

$$\begin{aligned} & ||B_{ij} - B_{lm}|| \\ = & \sum_{k=1}^{n} w_k^2 (B_{ijk} - B_{lmk})^2 \end{aligned}$$

$$\begin{aligned}
&= \sum_{k=1}^{n}(w_k^2 B_{ijk}^2 + w_k^2 B_{lmk}^2 - 2w_k^2 B_{ijk}B_{lmk}) \\
&= n + n - 2\sum_{k=1}^{n} w_k^2 (\frac{A_{ijk} - \overline{A_{ij}}}{\sigma'_{A_{ij}}})(\frac{A_{lmk} - \overline{A_{lm}}}{\sigma'_{A_{lm}}}) \\
&= 2n - 2nCorr(A_{ij}, A_{lm})
\end{aligned}$$

Similarly,

$$||B_{pq} - B_{uv}|| = 2n - 2nCorr(A_{pq}, A_{uv})$$

and the Claim follows.

3.5. Bag generation and image retrieval

Now we are ready to put everything together. For every image in our database, we do the following pre-processing:

1. If it is a color image, convert it into a gray-scale image.
2. Select some regions from the image, according to Section 3.2. Throw out regions whose variances are below a certain threshold.
3. Extract two sub-pictures from each region: one as the image itself in the region, and the other as its left-right mirror image. For each sub-picture, perform smoothing and sampling as illustrated in Figure 3 to get an $h \times h$ matrix. Treat this as an h^2-dimensional feature vector.
4. Transform each feature vector into a new one according to Section 3.4, i.e., subtract its mean from it and then divide it by its standard deviation. (All weights are 1 to start with.)
5. For each image in our database, we have obtained a number of feature vectors (after the transformation). Treat each one as an instance and put them together to form a bag for the image.

After these steps, our image database is ready to respond to user queries. The user is asked to select several positive and negative examples. The system puts together the corresponding image bags of multiple-instance data and feeds them into the DD algorithm. The DD algorithm returns an "ideal" point in the feature space as well as a set of feature weight values which maximize Diverse Density. Then the system goes to the image database and ranks all images based on their weighted Euclidean distances to the ideal point. (To find the distance from an image to the ideal point, it computes the distances of all of its instances to the point, and then picks the smallest one.) It then retrieves images in the ranked order. If the retrieval results are not satisfactory, the user may obtain better performance by picking out false positives and/or false negatives, adding them to the examples and training the system again.

3.6. Controlling feature weight factors

The DD algorithm finds an "ideal" feature vector t and a set of weights w to maximize Diverse Density. However, in the presence of few negative instances, it tends to push most of the weight factors towards zero, leaving only a few large weight values, which means that we are only using a small fraction of pixels to classify and retrieve images. Since we have very little training data, a too-simple concept based on a few pixels is likely to work well on the training set. However, it is not likely to generalize well, especially for complex image concepts. To address this issue, we impose a constraint on the sum of weight factors, as discussed below.

Without loss of generality, we require that all weight factors be between 0 and 1: $0 \leq w_k \leq 1, k = 1, 2, ..., h^2$. ($h^2$ is the number of dimensions in the feature vectors.) We can limit the change in weight factors w_k by imposing the following constraint, which sets a lower bound for the sum of weights:

$$\sum_{k=1}^{h^2} w_k \geq \beta \cdot h^2$$

where β is a constant between 0 and 1. When $\beta = 0$, there is no restriction on the weights, and we are back to the original DD algorithm. When $\beta = 1$, we are forcing all weight factors w_k to be equal to 1. The restrictions on weight factors are easily controlled by changing β values. For example, when $\beta = 0.5$, the average of weight factors must be greater than 0.5, so no more than half of the weight factors can be close to zero.

The simple unconstrained maximization algorithm used in the original DD method would no longer work to find the maximum with this new constraint. We switch to a more powerful algorithm called CFSQP (C code for Feasible Sequential Quadratic Programming) [7], which is capable of handling maximization problems with constraints. As will be shown in Section 4, this approach works well on a wide variety of situations.

4. Results

We have tested our system on two different image databases. One is a natural scene image database, consisting of 500 pictures, 100 each for waterfalls, mountains, fields, lakes/rivers, and sunsets/sunrises. These are taken from the COREL library, the same database as used in [11]. The other one is an object image database, consisting of 228 pictures from 19 different categories, such as cars, airplanes, pants, hammers, cameras, etc. These are downloaded from the websites of AVIS Car Rental (www.avis.com), Bicycle Online (www.bicycle.com), Continental Airlines (www.flycontinental.com), Delta Airlines (www.delta-air.com), J. Crew (www.jcrew.com), JCPenney

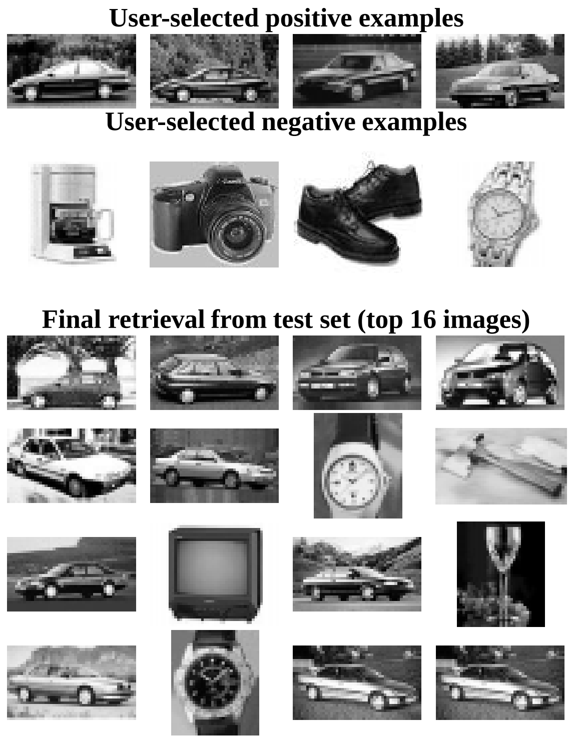

User-selected positive examples
User-selected negative examples
Final retrieval from test set (top 16 images)

Figure 6. A sample run with 3 rounds of training: retrieving cars

(www.jcpenney.com), Ritz Camera (www.ritzcamera.com), Sears (www.sears.com) and Sony (www.sony.com).

4.1. Experimental setup

To simulate user feedback while minimizing user intervention, we followed the same experimental method as used in [11]:

The entire image database is split into a small *potential training set* and a larger *test set*. The correct classifications for all images in the potential training set are known to the system. After the user selects positive and negative image examples, we generate corresponding bags and run DD algorithm once, and then use the results to rank images from the potential training set. Since their correct classifications are already known, the system can evaluate its own performance on these images without asking the user. It can pick out some false positives and/or false negatives and add them to the examples to train itself again. This process can be repeated more than once, and it effectively simulates what a user might do to obtain better performance. In the experiments of this section, 20% of images from each category are placed in the potential training set. The system picks out top 5 false positives from the potential training set and

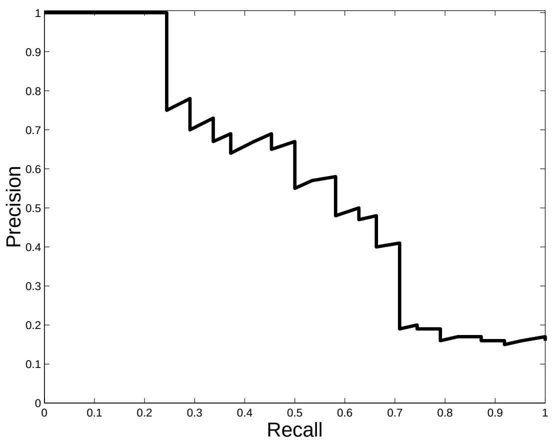


Figure 7. Precision-recall curve for Figure 6

adds them to the negative examples for a second round of training, and then picks out another top 5 false positives and trains for a third time. Finally it retrieves images from the larger test set. A sample run of the image retrieval system is shown in Figure 6, where the user wants to retrieve images that contain cars.

One way to evaluate image retrieval performance is to use precision-recall curves. Precision is the ratio of the number of correctly retrieved images to the number of all images retrieved so far. Recall is the ratio of the number of correctly retrieved images to the total number of correct images in the test database. In a precision-recall curve, we plot precision values against recall values. Figure 7 shows the precision-recall curve for the retrieval result in Figure 6. In this graph, precision is around 0.5 when recall is 0.6, which means: in order to obtain 60% of all waterfalls, about 50% of the images retrieved are true waterfalls.

4.2. Comparisons

We now study the effects of changing various parameters in the learning algorithm.

- Adjusting weight factor control

  The β value in the inequality constraint affects performance very much. In Figure 8, we show the results of varying β when retrieving sunset images. For each β, the precision-recall curve is shown. As β moves towards 0, the precision-recall curve tends to move close to that of the original DD algorithm. As β moves towards 1, the precision-recall curve tends to move close to that of forcing all weights to be identical. This is consistent with our analysis in Section 3.6.

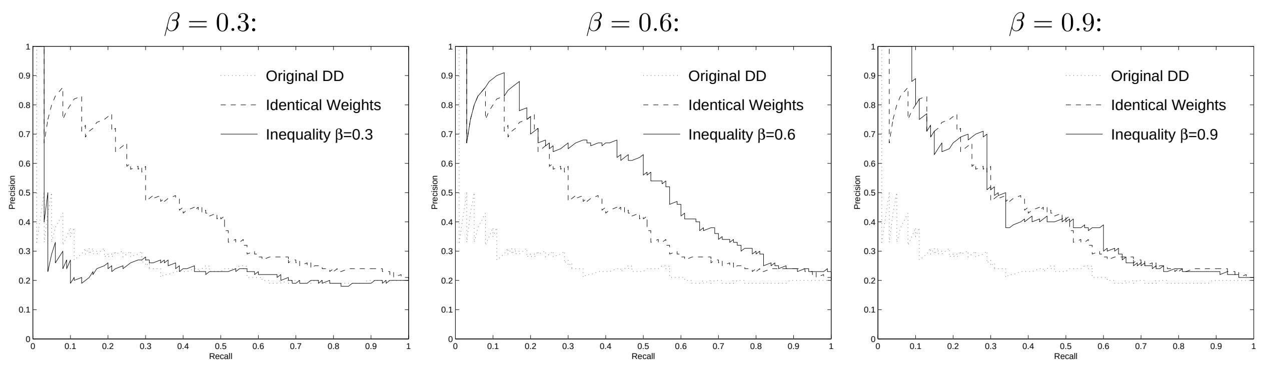


Figure 8. Precision-recall curves for different β in the inequality constraint

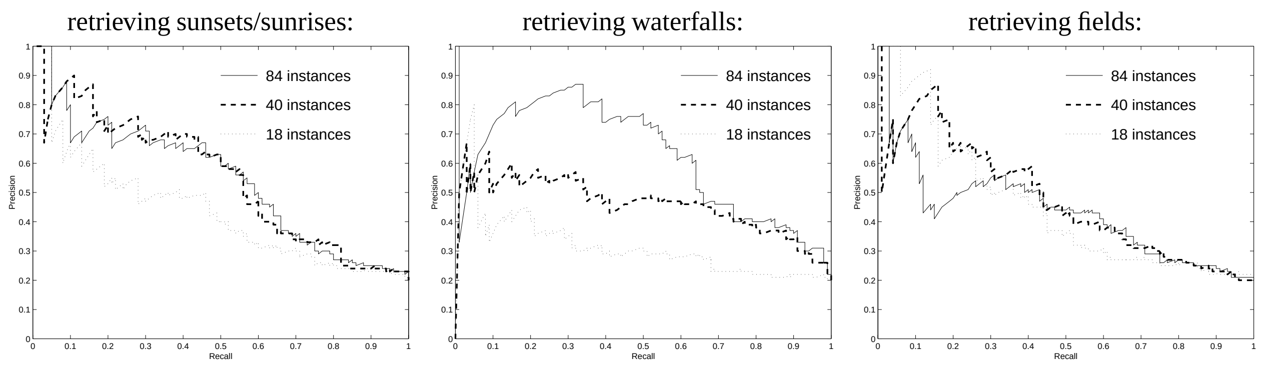


Figure 9. Precision-recall curves for different number of instances per bag

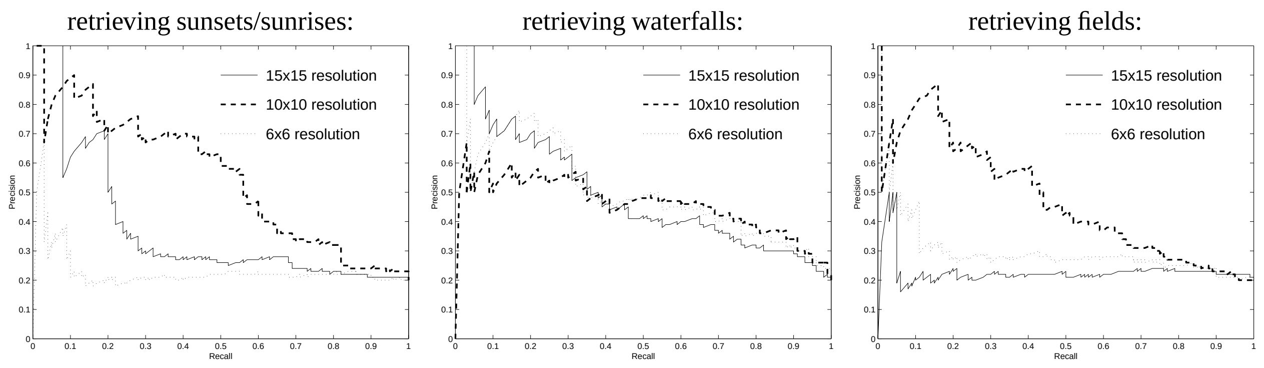


Figure 10. Precision-recall curves when smoothing and sampling at different resolutions

- Choosing different number of instances per bag

  Most of the experiments have been done with up to 40 instances per bag, by picking 20 different regions and taking mirror images. Figure 9 shows the effects of using fewer and more instances per bag. In general, having more instances per bag means a higher chance of hitting the "right" region. However, it also means introducing more noise which affects DD performance. Therefore, more instances per bag do not guarantee better performance. This is supported by Figure 9.

- Changing feature vector dimensions

  In most experiments, we smoothed and subsampled each image region to a low-resolution 10×10 matrix (a 100-dimensional feature vector) before comparing them against each other. We can use other resolutions (i.e., feature vector dimensions) as well. Figure 10 shows the effects of doing so. In many cases, as we increase the resolution, performance first rises, then declines. The problem with a very low resolution is that it does not give much information to compare for similarity. The problem with a very high resolution is that it makes our correlation similarity measure very sensitive to image shifts, and a higher resolution brings more noise. The "ideal" resolution which gives the best performance is highly dependent on the actual images.

- Comparing with a previous approach

  Now we compare our system with a previous approach developed by Maron and Lakshmi Ratan [11], which used DD algorithm with image feature vectors of color statistics and color distribution patterns. With a natural scene database, the performance of our system is very close to that of [11], as shown in Figure 11. The approach in [11] was targeted to retrieving color natural scene images, and would not work with object images. Our system makes use of only gray-scale information from the images, and has obtained comparable results on the natural scene database. Furthermore, it works with a wider range of image databases including object images.

5. Conclusions and future work

We have presented a new approach to the problem of content-based image database retrieval, using a weighted correlation similarity measure and the Diverse Density multiple-instance learning techniques. We have built and tested a system which allows users to select positive and negative example images and then automatically retrieves similar pictures from a medium-sized database. As has been shown in the test results, this approach performs reasonably well on both natural scenes and object images.

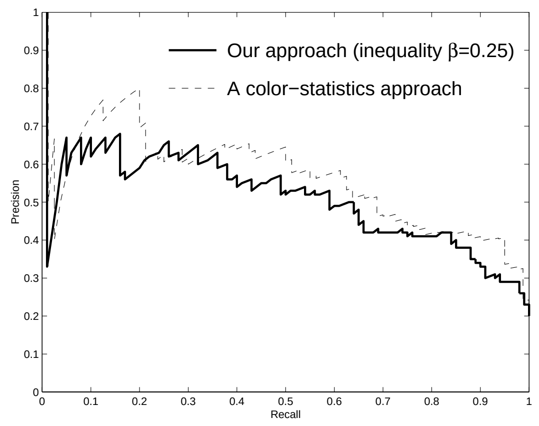


Figure 11. Comparison with a color-statistics approach

Compared with a previous approach in [11] which was targeted to retrieving natural scenes, our approach performs very close to theirs. Furthermore, our approach works well on object image databases, which [11] was not designed to handle.

We have studied the effects of putting more or fewer instances in each bag (by choosing more or fewer regions from each picture), and the effects of changing the number of feature vector dimensions (by smoothing and sampling image regions at different resolutions). Having more instances per bag does not guarantee better performance. Although the chance of hitting the "right" region increases as we put more instances into each bag, more irrelevant instances lead to more noise, which makes it more difficult for DD algorithm to find the ideal point. On the other hand, as we increase the number of dimensions of each feature vector, performance first rises and then drops down in many cases. This is because a very low resolution does not give enough information to compare for similarity, while a very high resolution adds noise and also makes our correlation similarity measure very sensitive to image shifts.

The treatment of feature space weight factors in the Diverse Density algorithm has significant effects on the performance of our system. The original Diverse Density algorithm gives the maximization process too much freedom, which drives most of the weight factors towards zero, leaving only a few large values. This is not desirable in the image retrieval domain. We experimented with imposing different inequality constraints on the sum of weights. The system is quite sensitive to these changes.

In Section 4.2, we discussed the effects of changing the β value in the inequality constraint. As a future direction, one might want to study how to choose β automatically to get optimal performance.

All experiments shown in this paper have been done on gray-scale images. Some attempts have been made to make use of color information in color natural scene images. We used RGB values separately and used a similar approach as we did with gray-scale images, tripling the number of dimensions of feature vectors. No significant improvements have been observed in this case. One other possible future direction would be to explore the effects of alternate color representation schemes, and to test on a larger variety of color images.

Also, one might want to try to use different feature vector representations and/or other similarity measures. We have attempted to preprocess the images with edge detection, and to use line and corner features in the feature vectors. However, the results we have got are not satisfactory.

Although our system is able to handle scaling changes across images, it is not designed to handle rotations. The correlation similarity measure can tolerate small rotations, but large rotations of the same object would be treated as dissimilar. One way to handle rotations would be to add more instances to represent different angles of view for each image region, although this would mean a significant increase in the number of instances per bag. There may be better ways, and this is yet another possible future direction.

6. Acknowledgements

Funding for this work is provided in part under the ONR Multidisciplinary University Research Initiative (MURI) through contract N00014-95-1-0600. We would like to thank Aparna Lakshmi Ratan and Oded Maron for providing part of the program code and data, and for their help throughout this work. This paper is based on Cheng Yang's M.S. thesis [20] at Massachusetts Institute of Technology, supervised by Professor Tomás Lozano-Pérez.

References

[1] J. S. De Bonet and P. Viola, "Structure driven image database retrieval", in *Advances in Neural Information Processing*, Vol. 10, 1997, pp. 866-872.

[2] T. G. Dietterich, R. H. Lathrop and T. Lozano-Pérez, "Solving the multiple-instance problem with axis-parallel rectangles", *Artificial Intelligence Journal*, Vol. 89, Nos. 1-2, 1997, pp. 31-72.

[3] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker, "Query by image and video content: the QBIC system", *IEEE Computer*, Vol. 28, No. 9, Sept. 1995, pp. 23-30.

[4] W. I. Grosky and R. Mehrotra, "Index-based object recognition in pictorial data management", *Computer Vision, Graphics, and Image Processing*, Vol. 52, No. 3, 1990, pp. 416-436.

[5] R. Jain, R. Kasturi and B. G. Schunck, *Machine Vision*, McGraw-Hill, 1995.

[6] A. Lakshmi Ratan, O. Maron, W. E. L. Grimson and T. Lozano-Pérez, "A framework for learning query concepts in image classification", *Computer Vision and Pattern Recognition*, 1999, pp. 423-429.

[7] C. T. Lawrence, J. L. Zhou and A. L. Tits, *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints*, Institute for Systems Research TR-94-16r1, University of Maryland, College Park, 1997.

[8] P. Lipson, *Context and Configuration Based Scene Classification*, Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[9] P. Lipson, E. Grimson and P. Sinha, "Configuration based scene classification and image indexing", in *Computer Vision and Pattern Recognition*, 1997, pp. 1007-1013.

[10] O. Maron, *Learning from Ambiguity*, Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[11] O. Maron and A. Lakshmi Ratan, "Multiple-instance learning for natural scene classification", in *Machine Learning: Proc. 15th International Conference*, 1998.

[12] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning", in *Advances in Neural Information Processing Systems*, Vol. 10, 1997, pp. 570-576.

[13] R. Mehrotra and J. E. Gary, "Similar-shape retrieval in shape data management", *IEEE Computer*, Vol. 28, No. 9, Sept. 1995, pp. 57-62.

[14] T. M. Mitchell, *Machine Learning*, WCB/McGraw-Hill, 1997.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, 2nd Edition, Cambridge University Press, 1992.

[16] S. Ravela, R. Manmatha and E. M. Riseman, "Scale-space matching and image retrieval", *Proc. Image Understanding Workshop*, Vol. 2, 1996, pp. 1199-1207.

[17] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, 2nd Edition, Vol. 1, Academic Press, 1982.

[18] M. Szummer and R. W. Picard, "Indoor-outdoor image classification", in *IEEE International Workshop on Content-based Access of Image and Video Databases*, Bombay, India, 1998.

[19] A. Vailaya, M. Figueiredo, A. Jain and H. Zhang, "Content based hierarchical classification of vacation images", in *Proc. IEEE International Conference on Multimedia Computing and Systems*, Vol. 1, 1999, pp. 518-523.

[20] C. Yang, *Image Database Retrieval With Multiple-Instance Learning Techniques*, M.S. thesis, Massachusetts Institute of Technology, 1998.
http://www.ai.mit.edu/people/cheng/thesis/image-mi.ps.gz