Effective Image Mining by Representing Color Histograms as Time Series

Zaher Al Aghbari
Department of Computer Science, University of Sharjah, UAE
E-mail: zaher@sharjah.ac.ae
[Received December 7, 2007; accepted October 17, 2008]

Due to the wide spread of digital libraries, digital cameras, and the increase access to WWW by individuals, the number of digital images that exist pose a great challenge. Easy access to such collections requires an index structure to facilitate random access to individual images and ease navigation of these images. As these images are not annotated or associated with descriptions, existing systems represent the images by their extracted low level features.

In this paper, we demonstrate two image mining tasks, namely image classification and image clustering, which are preliminary steps in facilitating indexing and navigation. These tasks are based on the extraction of color distributions of images. Then, these color distributions are represented as time series. To make the representation more effective and efficient for the data mining tasks, we have chosen to represent the time series by a new representation called SAX (Symbolic Aggregate approXimation) [14]. SAX based representation is very effective because it reduces the dimensionality and lower bounds the distance measure. We demonstrate by our experiment the feasibility of our approach.

Keywords: image mining, image classification, image clustering, symbolic representation of color histograms, SAX based representation

1. Introduction

Recent advances in technologies such as image digitization, storage and transmission caused the number of digital images to increase tremendously. Therefore, content-based image classification and retrieval systems have been the subject of many multimedia data mining research works in recent years. Image mining deals with the extraction of implicit knowledge, image data relationships or other patterns not explicitly stored in the image databases [1]. Early systems, such as QBIC [2], VisualSEEK [3], Netra [4] and MARS [5], facilitate classification, indexing, and retrieval of images, mainly, based on low-level features of images, such as color, texture and shape. Although such systems enjoy the advantage of being fully automatic, they are hard to use by non-expert users due to the semantic gap that exist between the user’s semantic needs and low-level system requirements. To bridge this semantic gap, semantic classification and clustering of images has attracted much interest from the data mining community.

The work in [6] is one of the earliest multimedia data mining systems that includes a component for mining classes of images. However, the automatic extraction of semantic elements of images is not reliable as opposed to the low-level features. For example, the color distribution which is captured by a color histogram can be simply extracted for any image, but the automatic identification of, say, trees or sky in the image may be uncertain. For reliable identification of image elements, the work in [7] manually identifies (labels) the image element (called visual keyword) from samples of visual content domain. Then, an image instance to be classified is compared against those visual keywords to detect the specific visual keywords of the image instance. Later the visual keywords approach was integrated with the conceptual graph based representation in [8]; as a result, the paper shows that retrieval precision of family photographs is improved as compared to either approach alone.

A different approach to organize the images into semantic categories is shown in [9], in which training sample regions of images are classified by a neural network into 11 clusters (e.g. Brick, Cloud, Grass, etc.). A different classifier was created for every cluster. Further, based on the k-nearest neighbor classifier, the work in [10] classifies internet images into photographic versus synthetic images. Then, the photographic images are classified into portrait versus non-portrait. Unlike the probabilistic approach of [11] and the Bayesian framework of [12] that construct a general explicit description of the target function from training samples, the k-nearest neighbor classifier approach construct a local approximation to the target function that applies in the neighborhood of a new image query.

The approach in [11] proposes a probabilistic approach to label small areas of one set of images as either man-made images or natural, and the local areas of another set were labeled as either inside or outside. Another approach [12] proposes a Naïve Bayesian classifier to classify vacation images. Then, in [13], the authors developed an incremental learning paradigm for the probabilistic approach of [12]. It classifies images into city versus land-
scape and then further classifies the landscape images into sunset, forest and mountain classes.

In this paper, we present two image mining tasks, namely image classification and image clustering. These tasks are based on the extraction of the color distribution of images then these color distributions are represented as time series. Where time series data are measurements of a variable taken at regular intervals over time. Wide classes of practically important data are represented as time series, such as economic and social data, weather records, and sports data. Also data that does not appear to be a time series can often be usefully represented as a time series, such as DNA, Text, Video, and Images [15].

Normally, dealing with time series is difficult, mainly, due to its very high dimensionality. Therefore, it is essential to map a time series to a suitable representation that will reduce its dimensionality without sacrificing its important features. That is the representation should lower bound the original time series when computing its distance from a reference time series, say a query time series. Thus, to make the representation more effective and efficient for the data mining tasks, we have chosen to represent the time series by a representation called SAX (Symbolic Aggregate approximation) [14]. SAX based representation is very effective because of its two main characteristics: dimensionality reduction and lower bounding.

A SAX representation of time series is a compact sequence of symbols (say, alphabets) that can be efficiently compared and consequently speeds up the clustering and classification tasks. We used the K-nearest neighbor method and the K-means method to classify and cluster the SAX representations, respectively.

The rest of this paper is organized as follows. In Section 2, we discuss the background of this research and explain the SAX method. In Section 3, we introduce our classification and clustering algorithms based on the SAX representation of time series that represents the color distributions of images. In Section 4, we discuss the results of our experiments and show the feasibility of SAX representation of image data. Finally, we conclude the paper in Section 5.

2. Background

Before introducing our algorithms, we briefly review the SAX (Symbolic Aggregate approximation) representation of a time series as presented in [14]. Although there are many representations for time series in the literature, however SAX is the only one that reduces the dimensionality and lower bounds the $L_p$ norms (distance functions) [15]. Where lower bounding means that the distance between any two vectors in the SAX representation is smaller than, or equal to, the distance between these two vectors before converting them to their SAX representation. We begin by a formal definition of time series:

**Definition 1.** A time series $T = d_1, d_2, \ldots, d_n$ is an ordered set of $n$ real-valued variables.

Before dealing with time series, we have to remove the distortions that could greatly affect the results of the data mining tasks. Thus, we normalize each time series to have a mean of zero and a standard deviation of one, since it is well understood that it is meaningless to compare two time series with different offsets and amplitudes. The common distortions that we remove by the normalization process are: 1. **Offset translation** (Fig. 1) and 2. **Amplitude scaling** (Fig. 2).

The following formula is used to normalize the time series.

$$ T = \frac{T - \text{mean}(T)}{\text{std}(T)} $$  

(1)
Table 1. Meaning of the symbols used in the paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>Time series</td>
</tr>
<tr>
<td>( n )</td>
<td>Length of time series</td>
</tr>
<tr>
<td>( w )</td>
<td>Number of PAA segments</td>
</tr>
<tr>
<td>( p )</td>
<td>Number of alphabets used to represent the time series</td>
</tr>
<tr>
<td>( \beta_i )</td>
<td>A breakpoint under the under a Gaussian curve that divide the curve into ( p ) equi-probable regions.</td>
</tr>
<tr>
<td>( I_{DB} )</td>
<td>Training dataset for the K-Nearest Neighbor algorithm</td>
</tr>
<tr>
<td>( I )</td>
<td>An instant, or image, in the training dataset</td>
</tr>
<tr>
<td>( K )</td>
<td>Number of neighbors</td>
</tr>
<tr>
<td>( K^n )</td>
<td>The ( K^{th} ) neighbor</td>
</tr>
<tr>
<td>( I_{Q} )</td>
<td>Test dataset for the K-Nearest Neighbor algorithm</td>
</tr>
<tr>
<td>( I_K )</td>
<td>An instant, or query image, in the test dataset</td>
</tr>
</tbody>
</table>

where \( T \) is the time series, \( \text{mean}(T) \) is the mean value of time series variables, and \( \text{std}(T) \) is the standard deviation of the time series variable.

2.1. Converting Time Series to SAX Representation

After normalizing the time series, it is converted to a Piecewise Aggregate Approximation (PAA) then to the symbolic representation SAX (see Fig. 3). Table 1 explains the symbols used in this paper.

The conversion process is performed by the time series to SAX representation algorithm, which takes as input the time series \( T \) that’s stored in an array, it’s length \( n \), the number of PAA segments \( w \) and the number of alphabets used to represent the time series \( p \).

Algorithm: time series to SAX representation

Input: \( T, n, w, p \)

Output: SAX representation

1. Normalize \( T \) using Eq. (1).
2. Verify that \( n \) is divisible by \( w \) and \( p \) is between 3 and 10 inclusively.
3. Transform \( T \) into PAA coefficients using Eq. (2).
4. Convert the PAA coefficient into SAX symbols

The details of the time series to SAX representation algorithm, are explained below:

- An input time series \( T \) of length \( n \) is normalize using Eq. (1).
- Verify that the values of \( w \) and \( p \) satisfy the following conditions:
  - The length \( n \) of \( T \) must be divisible by \( w \).
  - The value of \( p \) must be between 3 and 10 inclusively.

- Transform \( T \) into the Piecewise Aggregate Approximation (PAA) by dividing its length \( n \) into \( w \) equal-sized “frames”. The mean value of the data falling within a frame is calculated using Eq. (2) and a vector of these values (PAA coefficients) becomes the data-reduced representation.

\[
\bar{x}_i = \frac{w}{n} \sum_{j=\frac{i-1}{p}+1}^{\frac{i}{p}} x_j \quad \ldots \quad \ldots \quad \ldots \quad (2)
\]

- Since normalized time series follow the Gaussian distribution [14], breakpoints are determined so that they produce equal-sized areas under Gaussian curve. Breakpoints are a sorted list of numbers \( B = \beta_1, \ldots, \beta_{p-1} \) such that the area under a Gaussian curve from \( \beta_i \) to \( \beta_{i+1} = 1/p \) \( (B_0 \text{ and } B_p \text{ are defined as } -\infty \text{ and } \infty, \text{ respectively}) \), see Fig. 3. These breakpoints can be taken from a look-up table [15] and they divide the amplitude values of the time series into \( p \) equi-probable regions as follows:
  - Depending on the value of \( p \), the breakpoints are determined.
  - All the PAA coefficients that are below the smallest breakpoint are mapped to the symbol “a”, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol “b”, and so on.
  - After mapping all PAA coefficients to their corresponding symbols, we get a SAX representation of the input time series. For example, in Fig. 3 the SAX representation of the time series \( T_i \) is bbcbaacba.
2.2. SAX Distance

The distance between two time series is approximated by the distance between their SAX representations. Given two SAX representations \( \hat{Q} \) and \( \hat{C} \), the following formula returns the minimum distance between them:

\[
MINDIST(\hat{Q}, \hat{C}) = \frac{n}{w} \sqrt{\sum_{i=1}^{w} (|dist(\hat{q}_i, \hat{c}_i)|)^2}.
\] (3)

The distance between two SAX representations of a time series requires looking up the distances between each pair of symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate \( n/w \).

The \( dist() \) function is implemented using a lookup table [15]. The value in cell (row, column) for any lookup table can be calculated by the following expression where \( \beta(r,c) \) is the breaking point value:

\[
cell_{rc} = \begin{cases} 
0, & \text{if } |r-c| \leq 1 \\
\beta_{\max(rc)} - \beta_{\min(rc)}, & \text{otherwise.}
\end{cases}
\]

3. Classification and Clustering Images Time Series

In this Section, we present the algorithms for the image mining tasks: classification and clustering.

3.1. Classification

We use the K-Nearest Neighbor algorithm to classify the images into a fixed number of predefined classes based on their SAX representations. We can think of an image as a time series by using the histogram of the image’s color distribution. Then, this histogram is converted into a SAX representation.

Algorithm: K-Nearest Neighbor

Input: \( I_{DB}, K, IQ \)

Output: Class to which \( IQ \) is assigned

1. \( L_K = 0 \)
2. for each \( t \in I_{DB} \) do
3. \( \text{compute } MINDIST(t, IQ) \) using Eq. (3)
4. if \( L_K \) contains \(< K \) items
5. \( L_K = L_K \cup t \)
6. else if \( MINDIST(t, IQ) < \) distance between \( IQ \) and \( K^{th} \) neighbor so far
7. \( L_K = L_K - K^{th} \)
8. \( L_K = L_K \cup t \)
9. Assign \( t \) to the majority class in \( L_K \)

Generally, the K-Nearest Neighbor algorithm works as follows:

- The user specifies the value of \( K \), which represents the number of clusters to which the images will be divided into.
- Randomly select \( K \) images to be used as centroids (centers for the clusters).
- For each image in the database (query image), compute the SAX distance between the query image and each of the \( K \) centroids, and associate the query image with the closest cluster (most similar).
- For each cluster \( C_i \), compute its new mean.
- Repeat the previous two steps until all the centroids are no longer moving (the difference between previous centroid’s location and current location is smaller than a specified threshold), or until we reach a specified number of iterations, which is entered by a user to prevent the program from going into an infinite loop.

3.2. Clustering

For clustering, we use the K-means algorithm to cluster the images into \( K \) groups based on the SAX representation of the images, where \( K \) is determined by a user. Generally, the algorithm works as follows:

Algorithm: K-means

Input: \( I_{DB}, K \)

Output: \( K \) clusters of images

1. Randomly assign \( K \) images to be the centers (means) for the \( K \) clusters
2. repeat
3. for each \( IQ \in I_{DB} \) do
4. compute \( MINDIST(K_i, IQ) \) using Eq. (3)
5. assign \( IQ \) to the closest cluster
6. calculate the new means of the clusters
7. until convergence criteria is met.

Generally, the K-means algorithm works as follows:

- The algorithm maintains an ascending order list \( L_K \) that keeps the \( K \) nearest neighbors found so far.
- Each image in the database \( IQ \) is then compared with each image \( t \) in the training set \( I_{DB} \) by computing their SAX distance (Eq. (3)).
- If the list \( L_K \) contains less than \( K \) images from the training dataset, add image \( t \) into the list, otherwise, if the distance between \( IQ \) and image \( t \) is less than the distance between \( IQ \) and the \( K^{th} \) neighbor in \( L_K \), remove the \( K^{th} \) neighbor from the \( L_K \) and add \( t \) to \( L_K \).
- Finally, the query image \( IQ \) is classified to the majority class in the retrieved \( K \) nearest images in \( L_K \).
4. Experiments and Evaluation

To show the feasibility of the SAX representation of images, we compare the SAX representation against the normal histogram representation of the color distribution of images for both the classification and clustering results. The comparison is based on the Recall measure as defined below:

\[
\text{recall}_i = \frac{\text{number of relevant & retrieved images for class } i}{\text{total number of relevant images of class } i}. \tag{4}
\]

The images are of size 256 × 256 and stored as portable pixmap (ppm) type images. The collection of 420 images contains 12 different classes (Beach, Garden, Desert, Snow, Sunset, Rose, Banana, Tomato, Copper, Tiger, Wood, Gorilla). These images, which are used in our experiments, were taken from the Corel image collection at UCI Machine Learning Repository [18]. The Corel image collection contains various categories of general image types. This image collection has become a benchmark for testing image databases and has been used by many research papers. Fig. 4 shows representative images from each class.

The K-nearest neighbor algorithm is used to classify the image collection into its 12 classes. We selected 30% of the images in each class as a training dataset and the other 70% of the images are used for testing. The training dataset is selected so that it represents the various images of each class. By comparing the classification results of the SAX and normal histogram representations, we found that SAX representation is more accurate than the normal histogram representation. Fig. 5 shows the Recall of all the classes in the two representations. As shown in Fig. 5, the recall results is higher for the SAX representation for almost all the classes except for the two classes “Banana”, and “Gorilla”. However, the difference in recall between the two representations for the Banana and Gorilla classes is very small.

The normal histogram representation is that each image is represented by the frequency count of the colors in the image after quantizing the image into 256 colors. The color model used in the normal histogram representation is the HSV model. The HSV color space is chosen because it encapsulates information about the colors that are more familiar to humans [16] as compared with other color spaces. The K-nearest neighbor algorithm is used to classify the images of this representation and the Euclidean distance is used to compute the distance between color histograms (vectors of 256 dimensions).

We notice that working with approximations of the image data, such as the SAX approximations to represent the image data, improves the classification results as compared with the normal histogram representation of image data. Approximations of high dimensional vectors are more robust to classification.

By comparing the clustering results of the SAX and normal histogram representations, both representations gave different results as can be seen from Fig. 6. Depending on the randomly chosen centroids, the resulting clusters may reach a convergence state that is slightly different from the previous convergence states. However, the SAX representation is almost always more accurate than normal histogram representation. Fig. 6 shows the recall of both representation for a sample run, where the number of clusters is chosen to be 6. In the Fig. 6, we named the
produced cluster according to the majority images of the cluster.

It is shown in the literature of image representation that representing the image data approximations gives better results than representing the image data with raw color histograms. In [17], the authors show that initializing the clusters with low dimensional vectors (approximations of the data) improves the quality of the clustering results. Therefore, our technique improves the clustering results because the clusters are initialized with the SAX representation images.

It is also obvious that using approximations of the image data not only improves the quality of classification and clustering, but also significantly improves the efficiency of classification and clustering. It has been shown in [14] that SAX representations of image data improve the efficiency by at least 2 orders of degrees on the average.

5. Conclusion

This paper presented a novel approach of two image mining tasks, namely image classification and image clustering. The approach is based on a new image representation called SAX representation, which gives more accurate results in classification and clustering image databases as compared against the color histogram representation of images. The SAX representation of images renders itself as a better alternative to the common histogram representation specially for image mining tasks. The improvement in results is mainly due to the fact that the SAX representation reduces the dimensionality while lower bounding the distance measure. That is lowering the dimensionality would reduce the unnecessary details and lower bounding would emphasize the main features in the representation. Thus, the results become more effective and yet more efficient.

References:


