Expert Systems with Applications 38 (2011) 764–775

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Variance enhanced K-medoid clustering ☆

Por-Shen Lai *, Hsin-Chia Fu

Department of Computer Science, National Chiao-Tung University, Hsin-Chu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Clustering
K-medoid
Data variance
Polygon model
Image similarity

ABSTRACT

This paper proposes new variance enhanced clustering methods to improve the popular K-medoid algorithm by adapting variance information in data clustering. Since measuring similarity between data objects is simpler than mapping data objects to data points in feature space, these pairwise similarity based clustering algorithms can greatly reduce the difficulty in developing clustering based pattern recognition applications. A web-based image clustering system has been developed to demonstrate and show the clustering power and significance of the proposed methods. Synthetic numerical data and real-world image collection are applied to evaluate the performance of the proposed methods on the prototype system. As shown as the web-demonstration, the proposed method, variance enhanced K-medoid model, groups similar images in clusters with various variances according to the distribution of image similarity values.

1. Introduction

Clustering techniques have been widely used in many applications, such as pattern recognition, data mining (Pao, Chen, Lai, Xu, & Fu, 2008), user interface design, and so on. Using clustering methods (Fu et al., 2000; Haykin, 1994) to approximate the data distribution of target objects (pattern) in feature space, the obtained data clusters can be used to recognize target objects (pattern) by checking whether a new object is in a cluster or not. Besides, clustering methods (Guyon & Elisseeff, 2003; Schapire & Singer, 1999) are also useful in finding patterns of labeled data groups for data mining application. Using *labeled* training objects, supervised clustering methods (Haykin, 1994) learn *rules* to partition data objects into clusters (Pao, Chuang, Xu, & Fu, 2008). On the other hand, unsupervised clustering methods (Comaniciu & Meer, 2002; Hastie, Tibshirani, & Friedman, 2001b, chap. 14.3) partition feature space to distribute data objects into disconnected groups.

Generally, most clustering methods apply the feature extraction techniques to extract feature vectors, which usually represent the original data objects in numerical form to represent data objects as data points in feature space. Then, data points are partitioned into clusters according to the distribution in feature space. Therefore, the performance of data clustering is highly depending on the *quality* of extracted features.

K-means (Hastie, Tibshirani, & Friedman, 2001c, chap. 14.3.6), a popular clustering algorithm, groups data objects into clusters according to the norm between extracted feature vectors. In other words, using K-means clustering method, data objects in a cluster are represented by the mean of feature vectors of data points, the cluster with the closest mean. Since K-means algorithm does not use variance information to cluster data points, thus K-means model is inappropriate to cluster data points or objects having different variances in data distribution. Mixture Gaussian model (Bilmes, 1997; Hastie, Tibshirani, & Friedman, 2001d, chap. 14.3.7) represents a data cluster by a mean vector and a covariance matrix, which, respectively, illustrate the center location and the variance of data distribution in each orientation. Therefore, the mixture Gaussian model is capable to cluster data points of non-uniform variances.

Representing data objects by their feature vectors make the design of clustering method simpler. However, extracting representative feature vectors for data objects is still difficulty. For instance, although the similarity between two images can be measured based on some visual features, such as color, texture, or spatial distribution of pixels, creating feature vectors to represent the pairwise similarities between images by the norm between two feature vectors is still difficult. Therefore, several methods (Hastie, Tibshirani, & Friedman, 2001a, chap. 14.3.10) are proposed to cluster data objects using pairwise similarities between data objects instead of measuring the distance between extracted feature vectors of data objects.

K-medoid algorithm (Hastie et al., 2001a) has been widely used to cluster data points according to their pairwise similarity values. Given similarity values between every pair of data objects, an

☆ This research was supported in part by the National Science Council under Grant NSC 95-2221-E-009-218.

* Corresponding author.
E-mail addresses: porshenlai@yahoo.com (P.-S. Lai), hcfu@csie.nctu.edu.tw (H.-C. Fu).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.07.030

iterative learning process can group data objects into clusters to maximize the sum of similarity values for each cluster. That is, when the sum of similarity values is maximized, K-medoid algorithm groups data objects of equal to or lesser than similarity value in one cluster.

The decision boundaries that are generated by a given K-medoid model are the perpendicular bisector hyperplane of the line segment from the medoid of one cluster to another. That is, the variance of data distribution in each cluster is assumed to be uniform. However, the variances of different orientations of the data distribution in a cluster may be different. In this paper, the concept of different variances is suggested to be included in the original K-medoid model. Therefore, a new variance enhanced K-medoid model is proposed to group data objects in clusters with variant variances.

In addition, in order to use the new K-medoid model to cluster similar images, a similarity measurement between two images is also proposed and briefed as follows. First, color information is used to segment image into regions of similar color. Then, the similarity between two images is measured by computing the bitmap difference in coverage region. Based on the proposed methods, a web-based prototype system (The demonstration of variance enhanced K-medoid model, http://www.csie.nctu.edu.tw/~pslai/ASKMDDemo), which displays variable-sized thumbnails of images, is developed and tested. The prototype system groups similar images in a cluster, and the image at the cluster center (medoid) is shown in larger size to depict each group. And the other similar images in one group are displayed together and shown in smaller size to save display space in screen.

This paper is organized as follows. The K-medoid algorithm is introduced in Section 2. Then, the variance enhanced K-medoid clustering is proposed and presented in Section 3. Section 4 introduces the proposed methods for the similarity measurement of a image collection. The web-based prototype system is then demonstrated and evaluated in Section 5. Finally, concluding remarks are drawn in Section 6.

2. K-medoid

Given a data point set P, the medoid of P is the point p_m with the smallest sum of distance from p_m to all the other points p_i in P. Medoid p_m of P is mathematically defined as follows:

$$p_m = \arg\min_{p_i \in P} \sum_{p_j \in P} \mathcal{N}(p_i, p_j),$$

where $\mathcal{N}(p_i, p_j)$ is the norm between two points p_i and p_j.

For one-dimensional data, medoid of P is actually the median. For a data point set P, let $Q(x)$ be the sum of distance from point x to all the other points p_i in P, $Q(x)$ can be formulated as follows:

$$Q(x) = \sum_{p_i \in P} \mathcal{N}(p_i, x).$$

Let m_d be the medoid of P, and p_i be another data point in P. According to the definition of medoid, $Q(m_d) - Q(p_i)$ must be smaller or equal to zero.

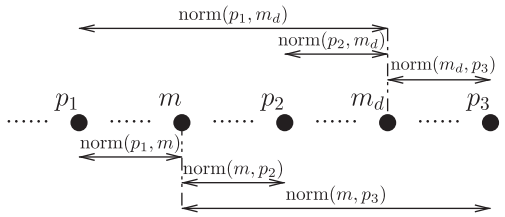

Fig. 1. The relation between medoid and median for one-dimensional data points. Let m be median and m_d be medoid. Since medoid m_d is the minimal of $Q(m_d)$, medoid m_d is equal to median m.

Fig. 1 illustrates the relation between median and medoid. Let m be the median, and m_d be the medoid of P. For each point p_i in P, we may observe the followings.

- When p_i is located between median m and medoid m_d, then $\mathcal{N}(p_i, m_d) = \mathcal{N}(m, m_d) - \mathcal{N}(m, p_i)$.
- When median m is located between p_i and medoid m_d, $\mathcal{N}(p_i, m_d) = \mathcal{N}(p_i, m) + \mathcal{N}(m, m_d)$.
- When medoid m_d is located between median m and p_i, $\mathcal{N}(m_d, p_i) = \mathcal{N}(m, p_i) - \mathcal{N}(m, m_d)$.

$Q(m_d) - Q(m)$ can be rewritten as follows:

$$\begin{aligned}
Q(m_d) - Q(m) &= \sum_{p \in (-\infty, m]} \mathcal{N}(m, m_d) - \sum_{p \in [m_d, \infty)} \mathcal{N}(m, m_d) \\
&\quad + \sum_{p \in (m, m_d)} \mathcal{N}(p, m_d) - \mathcal{N}(m, p) \\
&= \sum_{p \in (-\infty, m]} \mathcal{N}(m, m_d) - \sum_{p \in [m_d, \infty)} \mathcal{N}(m, m_d) \\
&\quad + \sum_{p \in (m, m_d)} (\mathcal{N}(p, m_d) - \mathcal{N}(m, m_d) + \mathcal{N}(p, m_d)) \\
&= \sum_{p \in (-\infty, m]} \mathcal{N}(m, m_d) - \sum_{p \in (m, \infty)} \mathcal{N}(m, m_d) \\
&\quad + 2 \sum_{p \in (m, m_d)} \mathcal{N}(p, m_d) \\
&= \mathcal{N}(m, m_d) + 2 \sum_{p \in (m, m_d)} \mathcal{N}(p, m_d)
\end{aligned}$$

Since $Q(m_d) - Q(m) \geqslant 0$, and $Q(m_d)$ is the minima (the definition of medoid), the medoid m_d must be equal to median m. The same results hold when m_d is smaller or equal to m. Therefore, for one-dimensional data, medoid is equal to median.

Intuitively, the medoid has higher chance to be located in the high density region in a data set than the mean can be. Therefore, medoid may be a better representation of a data set.

Similar to the popular K-means algorithm, K-medoid (Hastie et al., 2001a) can be used to cluster data points into K groups. Unlike K-means algorithm, K-medoid does not depend on the coordinates or values of each data points in feature space, the data required for medoid estimation is the norm between every pair of data elements.

Suppose data points in $P = \{p_1, \ldots, p_i, \ldots, p_N\}$ are partitioned into K clusters. By assigning each data point p_i to the cluster with minimal norm between p_i and the medoid of each cluster, a data set P can be iteratively partitioned into K clusters.

2.1. The learning algorithm of K-medoid clustering

The K-medoid cluster learning algorithm (Hastie et al., 2001a) is briefly stated as follows. Let $\mathcal{N}(p_i, p_j)$ be the norm between p_i and p_j in data set P, and K is the number of medoid.

Step 1. Given a current set of cluster centers $M^t = \{m_1^t, \ldots, m_k^t\}$, the current cluster assignments $C^t(p_i)$ of each data points $p_i \in P$ are evaluated by computing the closest cluster center for each data point p_i as follows:

$$C^t(p_i) = \arg\min_{m_j^t \in M^t} \mathcal{N}(m_j^t, p_i),$$

where $\mathcal{N}(m_j, p_i)$ is the norm between m_j and p_i, and $C^t(p_i)$ is the current cluster assignment for p_i.

Step 2. Given a set of current cluster assignments C^t, the new cluster centers $M^{t+1} = \left\{m_1^{t+1}, \ldots, m_j^{t+1}, \ldots, m_k^{t+1}\right\}$ are estimated by computing the medoid among the data points in the same assignment as follows:

$$m_j^{t+1} = \arg\min_{C^t(p_x)=m_j^t} \sum_{C^t(p_y)=m_j^t} \mathcal{N}(p_x, p_y),$$

where $C^t(p_i)$ is the cluster assignment for p_i.

Step 3. Iterate steps 1 and 2 until the assignments do not unchange.

2.2. Self-growing K-medoid

Generally speaking, asking user to assign the number of required clusters is not practical since the number of clusters is usually the natural of data distribution and is often unknown to a user.

In order to perform data clustering without knowing cluster number, the self-growing K-medoid clustering algorithm is proposed. By giving the maximal acceptable dis-similarity between a data point and its cluster center, the proposed method iteratively increases the number of clusters until the dis-similarity between any data point, and the associated cluster center is lower than a given threshold.

The basic concepts of the proposed method can be illustrated by the example shown in Fig. 2. The y-axis shows the value of weight for each points, and the x-axis is the location of data points. Based on K-medoid clustering algorithm, the point p_3 is first selected to be the first cluster center c_1. The weights of each data point are updated according to their norms with respect to the selected cluster center. Specifically, the weights of data points that are close to the selected cluster center will be decreased more than the weights of data points that are far away from the selected cluster center. Then, another data point locates in the middle of data points with larger weights is selected as the second cluster center c_2. After clustering all data points with the two selected cluster centers, the weights of data points are updated. The clustering processes are repeated to decrease the weights of data points until all weights are lower than to a given threshold.

To formulate this idea, a weighted medoid estimation is defined as follows:

$$p_n = \arg\min_{p_j \in P} \left(\sum_{p_i \in P} w_j w_i \mathcal{N}(p_j, p_i) \right).$$

The weighted medoid estimation organizes a new cluster center for data points with high weight values.

Combining the weighted medoid estimation and the standard K-medoid algorithm, the self-growing K-medoid method algorithm is introduced as follows:

Step 1. Initialize weight values w_i of each data point p_i to be 1.

Step 2. Use the weighted medoid estimation to create a new cluster center.

Step 3. Apply the standard K-medoid clustering algorithm to adjust the location of clusters centers p_m.

Step 4. Recompute the weights w_i of each data points p_i as follows:

$$w_i = 1 - \left(1 + \min_{p_m \in M} \mathcal{N}(p_i, p_m)\right)^{-1}.$$

Step 5. Iteratively repeat steps 2–4 until all weight values w_i are smaller than a given threshold.

Using the proposed self-growing K-medoid algorithm, the number of required model (medoid) can be determined to let the dissimilarity between data point and closest cluster center be less than a given threshold.

2.3. Discussion

Based on the mathematical representation of two clusters, a decision boundary can be determined to separate data points into two parts. As shown in Fig. 3(a), the decision boundary between two K-medoid clusters is the perpendicular bisectors of a line segment from one cluster center to another. Since the data variance is not considered, these decision boundaries may not separate data clusters properly. To achieve a better separation of the data points, variance values should be considered in determining the decision

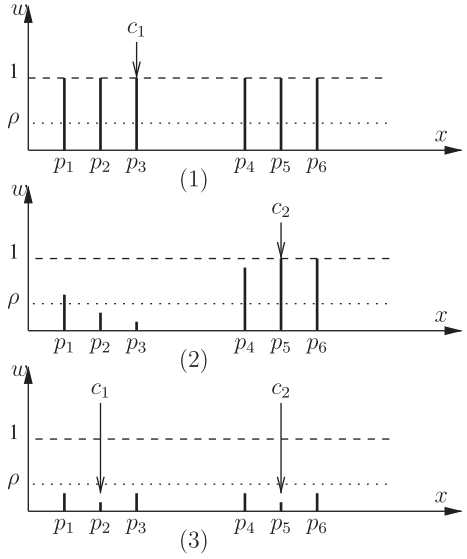

Fig. 2. A 1-D data distribution example is used to illustrate the idea of self-growing K-medoid Clustering algorithm. In each diagram, the heights (w) of a sample points represents its weight values. (1) At first, the middle point p_3 is selected as the first cluster center c_1. (2) By decreasing the weights of data points close to the first cluster center c_1, point p_5, which is the middle point of data points with large weight values, is selected to be the new cluster center c_2. (3) After refining the location cluster centers by standard K-medoid clustering algorithm, the weights (the dis-similarity between data point and the closest cluster center) of all data points are less than a given threshold.

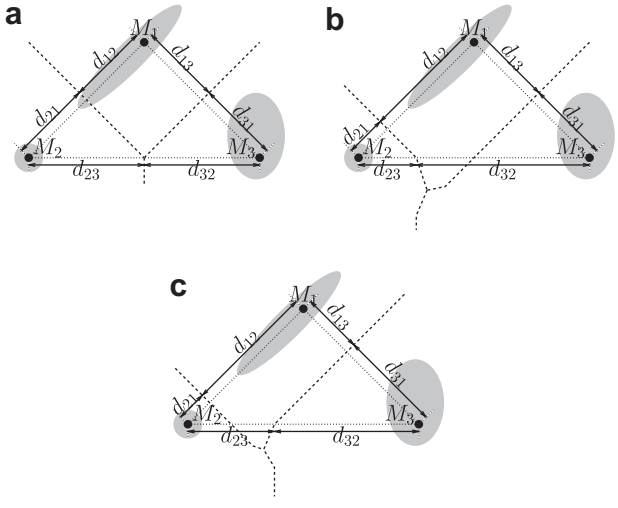

Fig. 3. The comparison among three kinds of decision boundary for clustering model. Shaded area are the distributed regions of data points. (a) The decision boundary is the perpendicular bisector of the line segment between two cluster centers. The variance of data distribution is not used in creating data clusters. (b) A single variance for every direction is used for each cluster. For two cluster, the ratio between the two variances of clusters is equal to the ratio between the distances from cluster centers to decision boundary. (c) For each two clusters, the location of decision boundary is decided according to the data variances along the line segment between two cluster centers.

boundaries. An variance enhanced K-medoid is illustrated in Fig. 3(b). By aggregating the norm between data points and its cluster center (medoid), the variance of each cluster is estimated Again, since the variance orientation of data distribution is still not considered, and the decision boundaries may not partition the feature space well enough. For instance, without considering variance orientation, a cluster with widely data distributed in one direction and with vary little data distributed in a perpendicular direction may result a large overall variance along the perpendicular direction. Fig. 3(c) shows the decision boundaries in corresponding to data cluster with multiple variances being considered along several orientations. Based on the location of data points in feature space, a K-means model can be improved by including the covariance matrix for each cluster, e.g., Gaussian mixture model. Since the location information of data points is not used in K-medoid model, improving the K-medoid model using the covariance matrix method seems impracticable. How to include the oriented variance in K-medoid method and how to estimate the variances in various orientations become essential issues for variance enhanced K-medoid clustering algorithm. The detailed description of this method will be illustrated in Section 3.

3. Variance enhanced K-medoid clustering

According to the discussion in Section 2.3, multiple variances along several orientations are required for K-medoid based clustering method. However, since the locations of data point in feature space are not available, estimating variance orientation using covariance matrix seems impractical. Therefore, multiple variances along line segments between cluster centers are proposed instead. Let us see that the variance along a line segment between cluster centers should be measured according to the data points located along the line segment. As the example shown in Fig. 4, four cluster centers M_1, M_2, M_3, and M_4 are included, and b_i's are the decision boundaries between the line segments from M_1 to M_i for $i = 2$ to 4. For instance, the variance along $\overline{M_1M_3}$ associated with cluster center M_1 is estimated by the data points located in the region of gray-scaled color. In general, decision boundaries segment the feature space into polygon-shaped regions for each cluster in a data set. And, the polygon-shaped region for a cluster can be further divided into several smaller pyramid-shape segments, whose tops are the cluster centers and bottom hyperplane are along the decision boundaries. The variance along the line segment $\overline{M_1M_3}$ is measured using data points located in the pyramid-shape segment, whose bottom edge is along the decision boundary b_3 which partitions two data clusters with centers at M_1 and M_3, respectively.

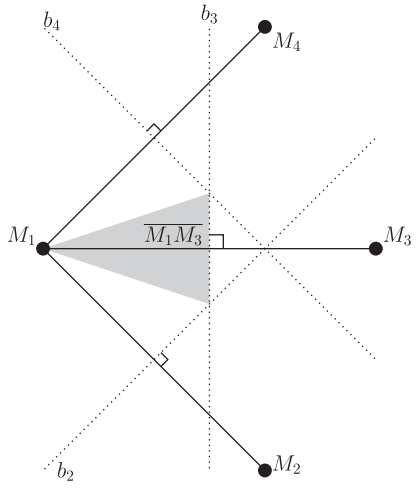

Fig. 4. Giving four clusters with centers at M_1, M_2, M_3, and M_4. For the center M_1, there are three associated decision boundaries b_2, b_3, and b_4. The variance between M_1 and M_3 is computed using the data points inside the pyramid-shape segment whose top is located at the cluster center M_1, and the bottom edge is along with decision boundary b_3.

To select appropriate data points for the computation of variance along line segments between cluster centers, a polygon-shaped data model, called Polygon descriptor (Lai & Fu, 2007, 2010), is suggested and briefly described in Section 3.1.

In the rest of this section, Polygon descriptor is introduced in Section 3.1. Then, the method that segments each cluster into sub-clusters, called side-clusters, along each line segment between cluster centers is depicted in Section 3.2. Using the projection method described in Section 3.3, the projection distribution of data points in each side cluster can be used to adjust decision boundaries between every pair clusters in the data points by the proposed method in Section 3.4. Finally, the variance enhanced K-medoid clustering method is presented in Section 3.5.

3.1. Polygon-shaped data model

As shown as Fig. 3, decision boundaries segment feature space into several polygon-shaped regions (data clusters). *Polygon descriptor* (Lai & Fu, 2007, 2010) was originally proposed for modeling a polygon-shaped data distribution.

A *Polygon descriptor* uses a reference center and N normal vectors to describe a polygon-shaped data distribution. A normal vector represents: (1) the normal direction of a hyperplane that encloses the convex unit and (2) the distance from the reference center to each hyperplane. Fig. 5(a) shows an exemplar Polygon descriptor for the representation of a 2D data distribution in a convex unit. The reference center is located at (20,20), and five normal vectors are $\begin{pmatrix}10\\0\end{pmatrix}$, $\begin{pmatrix}5\\5\end{pmatrix}$, $\begin{pmatrix}-5\\5\end{pmatrix}$, $\begin{pmatrix}-10\\0\end{pmatrix}$, and $\begin{pmatrix}0\\-10\end{pmatrix}$. By applying a 1-D probability function to each normal vectors, a polygon-shaped probability model is created as shown in Fig. 5(b). The polygon-shaped probability model (distribution) shown in Fig. 5 is centered at (20,20), and the variance corresponds to each normal vector is the length of each normal vectors, respectively.

According to the Polygon descriptor, one center point and M normal vectors are used to describe a M-side polygon. The center point is used as the reference origin of the coordinate system of a Polygon descriptor. And, M normal vectors are used to span the polygon-shaped coverage area of data points. Since normal vectors are orthogonal to the boundaries of a data distribution, the orientation of the boundaries of a data distribution can be represented by their normal vectors. Besides, the length of a normal vector indicates the data distribution variance along the normal vector.

Fig. 6 shows the components in a polygon descriptor. Data points are partitioned into clusters, called side-cluster, according to the length and orientation of each normal vectors. As shown in Fig. 6, for a data distribution containing M normal vectors, M side-clusters are determined according to the M boundary edges. The side-clusters corresponding to each boundary edges can be used to measure the variance of data points along the corresponding normal vectors.

3.2. Side-cluster segmentation

M normal vectors of a Polygon descriptor can be used to further segment a data cluster into M side-clusters. Let $\{\vec{a_1},\ldots,\vec{a_j},\ldots,\vec{a_M}\}$ be the M normal vectors of a Polygon descriptor. Given a set P of data points p_i, a sub-cluster assignment $C(p_i)$ for a data points p_i can be formulated as follows:

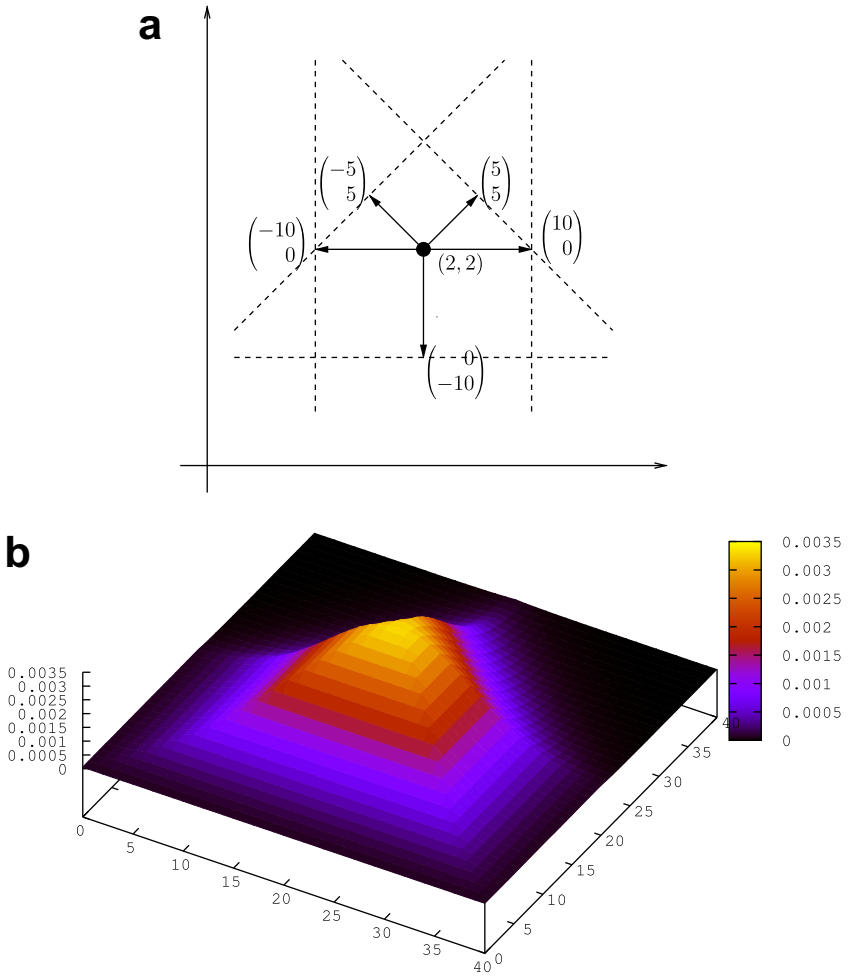

Fig. 5. (a) An example of a proposed polygon descriptor, where its center is at (20,20) and normal vectors (drawn as solid arrow) to each surrounding hyperplane are $(10,0)^T$, $(5,5)^T$, $(-5,5)^T$, $(-10,0)^T$, and $(0,-10)^T$. (b) The pentagon-shaped probability distribution for the polygon descriptor of (a).

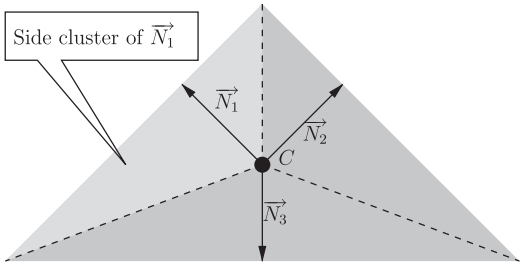

Fig. 6. A diagram shows the components of an exemplar polygon descriptor. One center point C is used as the reference origin of the coordinate system of a Polygon descriptor. $M(=3)$ normal vectors are emitted from the origin to M directions. According to the length and orientation of M normal vectors, M side-clusters are determined corresponding to each boundary edges. That is, the clustering of a side-cluster is determined by the length and orientation of normal vectors. The length and orientation of normal vectors can be estimated according to the statistical properties of data points in each clusters.

$$C(p_i) := \arg\max_{j=1}^{M} \frac{\vec{p}_i \cdot \vec{a}_j}{\vec{a}_j \cdot \vec{a}_j},$$

where $C(p_i)$ is cluster assignment for p_i, and $\vec{p}_i$ is the vector from reference center to p_i. According to the cluster assignments, there are M side-clusters, and how data points are partitioned into side clusters will be described in the following paragraph.

Fig. 7 illustrates the basic concept of how a side-cluster is formed. Let $\vec{a_1}$ and $\vec{a_2}$ be normal vectors of boundary edges b_1 adn b_2, respectively. The decision boundary between two side-clusters C_1 and C_2 is drawn in dot line, which is a hyperplane passing through the origin point O and the intersection point I of the two boundary edges b_1 and b_2. For a vector $\vec{p_1}$ from reference center to a point p_1 in C_1, its projection ratio on $\vec{a_1}$ is larger than the projection ratio on $\vec{a_2}$. That is,

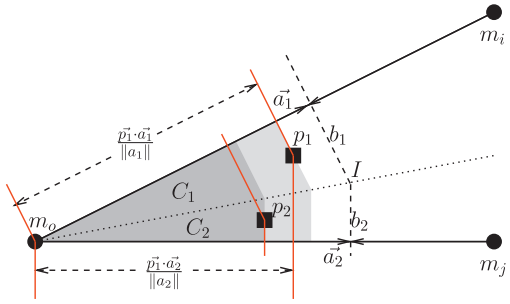

Fig. 7. Let $\vec{a_1}$ and $\vec{a_2}$ are normal vectors of side-clusters C_1 and C_2, respectively. The dash line shows the decision boundary between C_1 and C_2. For data vector $\vec{p_1}$ which is from cluster center O to data point p_1 in C_1, its projection ratio on $\vec{a_1}$ is larger than the projection ration on $\vec{a_2}$, i.e., $(\vec{p_1}\cdot\vec{a_1})/\|\vec{a_1}\|^2 > (\vec{p_1}\cdot\vec{a_2})/\|\vec{a_2}\|^2$. Similarly, the projection ratio of the data vector $\vec{p_2}$ in C_2 on $\vec{a_2}$ is larger than the projection ratio on $\vec{a_1}$, i.e., $(\vec{p_2}\cdot\vec{a_1})/\|\vec{a_1}\|^2 < (\vec{p_2}\cdot\vec{a_2})/\|\vec{a_2}\|^2$.

$$\frac{\vec{p_1}\cdot\vec{a_1}}{\|\vec{a_1}\|^2} > \frac{\vec{p_1}\cdot\vec{a_2}}{\|\vec{a_2}\|^2}.$$

By segmenting the data points of a cluster into side clusters corresponding to the line segments between cluster centers and by projecting the data points in a side cluster onto the corresponding line segment between cluster centers, an 1-D data distribution can be created. The 1-D data distribution can be used to measure the variance along the line segment between cluster centers or adjust the location of decision boundary between clusters. However, since the coordinates of data points for K-medoid model is absent, a new method is proposed in Section 3.3 to estimate the projection length without knowing the coordinate of data points.

3.3. Projections on the center-to-center link

Computing variance along normal vector using projection ratio requires the coordinate values of data points in feature space. In order to computing the projection ratio on normal vectors without knowing the coordinates of data points in feature space, a new projection method is proposed.

As shown in Fig. 8, given a point p, and two clusters with center points at m_1 and m_2. $\|\vec{m_1p_i}\|$, $\|\vec{m_2p_i}\|$, and $\|\vec{m_1m_2}\|$ are the norms from p to m_1, p to m_2, and m_1 to m_2, respectively. Let d be the projection of p on $\vec{m_1,m_2}$. $\|\vec{m_1d}\|$ is the projection length of $\vec{m_1p_i}$ on $\vec{m_1m_2}$. As shown in Fig. 8, the following relation holds:

$$\|\vec{m_1p_i}\|^2 = \|\vec{m_1d}\|^2 + \|\vec{p_id}\|^2,$$

$$\|\vec{m_2p_i}\|^2 = \left(\|\vec{m_1m_2}\| - \|\vec{m_1d}\|\right)^2 + \|\vec{p_id}\|^2.$$

Then, the length of $\|\vec{m_1d}\|$ can be derived as follows:

$$\left(\|\vec{m_1m_2}\| - \|\vec{m_1d}\|\right)^2 = \|\vec{m_2p_i}\|^2 - \|\vec{m_1p_i}\|^2 + \|\vec{m_1d}\|^2,$$

$$\|\vec{m_1m_2}\|^2 - 2\|\vec{m_1m_2}\|\|\vec{m_1d}\| = \|\vec{m_2p_i}\|^2 - \|\vec{m_1p_i}\|^2,$$

$$\|\vec{m_1d}\| = \frac{\|\vec{m_1m_2}\|^2 - \|\vec{m_2p_i}\|^2 + \|\vec{m_1p_i}\|^2}{2\|\vec{m_1m_2}\|}.$$

By projecting data points onto the line segment between two cluster centers, the 1-D data distribution along the line segment becomes available. Based on the 1-D data distribution, the decision boundary can therefore be adjusted to maximize the decision margin between two estimated clusters.

3.4. Decision boundaries adjustment

Using the projection method proposed in Section 3.3, data points in a cluster can be segmented into several side-clusters corresponding to the line segments between each pair of cluster centers. The projection of data points in a side-cluster on the corresponding line segment between cluster centers forms an 1-D data distribution. Based on the 1-D data distribution, the variance of data points along the line segment between cluster centers can be measured. Since the purpose of measuring variance is to adjust the location of decision boundary, the estimation or the measuring of variance can be avoid by directly adjusting the location of decision boundary according to projected 1-D data distribution.

As shown in Fig. 9, given two cluster centers m_1 and m_2. By projecting data points p_i (shown in black square dots) onto the line segment $\overline{m_1m_2}$, a series of 1-D location values (shown in black triangle dots) are measured. The following iterative procedure is proposed to find a decision boundary, such that the distance between the means of two partitioned clusters can be maximized. These 1-D location values are partitioned into two groups namely g_1 and g_2 according to the decision boundary b_{12}. The decision boundary b_{12} is determined to maximize the distance between μ_1 and μ_2, where μ_1 and μ_2 are the means of data groups g_1 and g_2, respectively.

Let α be the target location of decision boundary, and the projection ratio $h(p_i)$ for point p_i is defined as

$$h(p_i) = \frac{\vec{m_1p_i}\cdot\vec{m_1m_2}}{\vec{m_1m_2}\cdot\vec{m_1m_2}}.$$

Actually, $h(p_i)$ is the projection length of vector $\vec{m_1p_i}$ on $\overline{m_1m_2}$ dividing by the distance between two cluster centers. Then, α can be evaluated according to the following formula:

$$\alpha = \arg\max_{\alpha}\left(\frac{\sum_{h(p_i)>\alpha}h(p_i)}{\sum_{h(p_i)>\alpha}1} - \frac{\sum_{h(p_i)<\alpha}h(p_i)}{\sum_{h(p_i)<\alpha}1}\right),$$

where $\frac{\sum_{h(p_i)>\alpha}h(p_i)}{\sum_{h(p_i)>\alpha}1}$ is the average projection ratio that is larger than α, and $\frac{\sum_{h(p_i)<\alpha}h(p_i)}{\sum_{h(p_i)<\alpha}1}$ is the average projection ratio that is smaller than α.

The detail processes of estimating the decision boundary between two clusters are described as follows. Data points are clustered using the K-medoid method proposed in Section 2.2. For every pair of clusters C_i and C_j, for $j > i$, data points in C_i and C_j are projected onto $\vec{m_im_j}$, where m_i and m_j are medoids of C_i and C_j, respectively. Then, the line segment between C_i and C_j is partitioned into 10 intervals. By counting the number of points locating in one interval, a histogram is drawn. Based on the histogram, the decision boundary can be properly located so that the distance between two means of data sets, which are separated by the decision boundary, is maximized. All data points are then redistributed using the adjusted decision boundaries. The process repeats until the location of decision boundaries converged, such that the distance between μ_1 and μ_2 is maximized.

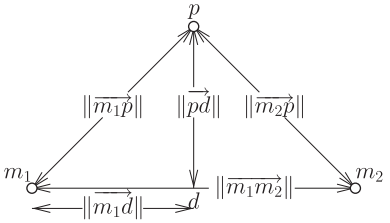

Fig. 8. m_1 and m_2 are two cluster centers. p is a data point. d is the projection point on $\overline{m_1m_2}$.

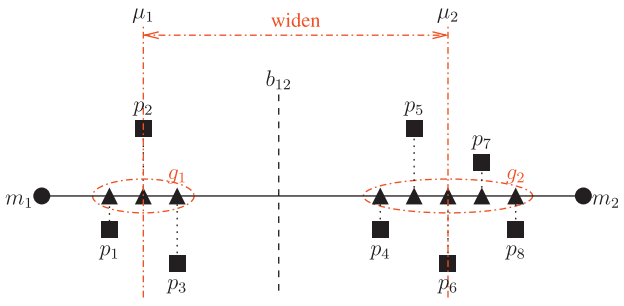

Fig. 9. Data points p_i are projected to the line segment between cluster centers m_1 and m_2. The decision boundary is located at the place which partition the projected data points (triangle points) into two groups with farthest mean location μ_1 and μ_2.

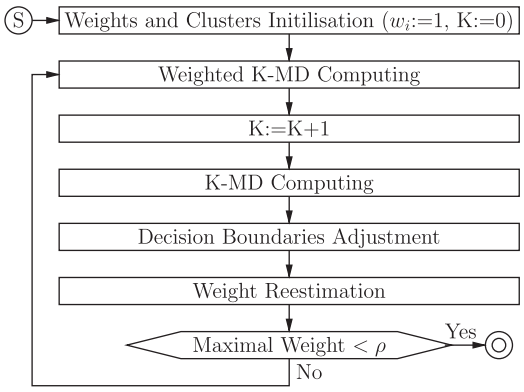

Fig. 10. The flow chart of variance enhanced K-medoid clustering. At first, general K-medoid algorithm is applied. According to the resulted clusters of general K-medoid algorithm, the cluster-to-cluster variance is estimated. Then, the weights for each data points are updated and new cluster is estimated based on the weighted data points. These steps are repeated until all weights are smaller than a given threshold ρ.

3.5. Variance enhanced K-medoid clustering method

The flow chart of variance enhanced K-medoid clustering is shown in Fig. 10. At first, one medoid is estimated. Then, the number of medoid is gradually increased until the distance from each data point to the closest cluster center is smaller than a given threshold. For each medoid, its positions are estimated using general K-medoid algorithm first. Then, the 1-D data distributions can be established by projecting data points in side-clusters to the corresponding line segment between cluster centers. According to these 1-D data distributions, the location of decision boundaries is adjusted to redistributed data points to each clusters. After the position of medoid converged, weights w_i of each points p_i is calculated as the following formula:

$$w_i = 1 - \left(1 + \min_{p_m \in M} \mathrm{sim}(p_i, p_m)\right)^{-1},$$

where M is a set of estimated medoids, and sim (p_i,p_m) is the projected similarity defined as follows:

$$\mathrm{sim}(p_i, p_m) = \arg\max_{p_j \in M, p_j \neq p_m} \frac{\vec{p_i} \cdot (\vec{p_j} - \vec{p_m})}{\alpha(\vec{p_j} - \vec{p_m}) \cdot \alpha(\vec{p_j} - \vec{p_m})},$$

where $\alpha(\vec{p_j} - \vec{p_m})$ is the vector from p_m to the decision boundary between the clusters associated to p_m and p_j. Based on these weights, a new medoid is created to include high weighted data points. By repeating these processes, the number of medoid gradually increasing until the maximal weight is lower than a given threshold.

Using the proposed variance enhanced K-medoid, the location of decision boundaries between each pair of cluster centers is related to the data distribution along a line segment between cluster centers now. A real-world application, photo album preview with variable-sized thumbnails, is proposed and implemented in Section 4 to illustrate the performance of the proposed variance enhanced K-medoid clustering method.

4. Real-world application

In this section, variance enhanced K-medoid is used to intelligently create image thumbnails, where "intelligently" means that the size and the display order of image thumbnail are decided without human intervention. Generally, image gallery (Pao et al., 2008) uses thumbnail to provide a quick preview for each photo (Pao et al., 2008). However, even though the thumbnail is downsized from the original photos, the display area is often still too small to contain all thumbnails. To contain all thumbnails of same size can be very difficult in selecting a proper size for all the thumbnails. Sometimes, it may be too crowded to clearly represent the original image, or it may be too large to put all thumbnails in a page. Therefore, variable-sized thumbnail seems a better alternative to fit all thumbnails into a page without losing too much details.

Since most of images in an image gallery may be similar to each other, these images can be partitioned into groups according to their similarity. For each group, a most representative image can be displayed by a normal sized thumbnail, and the rest of similar images can be shown by smaller thumbnails.

The detail description of the proposed system is depicted in the following sections. Color information of an image is used to create four bitmaps to show the distribution of dominant color components. Based on these bitmaps, the similarity between two images is measured. Then, the proposed variance enhanced K-medoid is applied to cluster all images into groups.

4.1. Color clustering

Giving an image I, K's dominant color components are selected to establish K's bitmaps which show the coverage of data distribution of dominant color components. K-means algorithm is used to cluster the color value of each pixel $p_{x,y} \in I$. Let $K = 4$, then four dominant colors m_i, for $i = 1,\ldots,4$, are estimated by K-means algorithm. Then, four bitmaps $I(x,y)_i$ are created as follows:

$$I(x,y)_i = \begin{cases} 1 & , \text{if } \arg\min_{m_i} \mathrm{norm}(m_i, c_i) = m_i \\ 0 & , \text{if } \arg\min_{m_i} \mathrm{norm}(m_i, c_i) \neq m_i \end{cases} \text{ for } i = 1, \ldots, 4;$$

where $I(x,y)_i$ is the bitmap created using dominant color m_i, and c_i is the ith color component at (x,y).

An example is shown in Fig. 11, the size of each image is normalized to 80×80 at first. Then, K-means algorithm is used to cluster pixels of original image to four color component groups according to the color values of pixels. And, four bitmaps are established to show the distribution of data points in each groups. These four bitmaps will be used to measure the similarity between images (in Sections 4.2 and 4.3).

4.2. Similarity measurement of two bitmaps

Giving two normalized bitmaps that show the coverage area for data distribution of specified color components, the similarity $S(A,B)$ between two bitmaps A and B is defined as follows:

$$S(A,B) = \frac{\|\mathcal{A} \cap \mathcal{B}\|}{\|\mathcal{A} \cup \mathcal{B}\|},$$

where $\mathcal{A}$ and $\mathcal{B}$ are the sets containing the pixels with value 1 in bitmap A and B, respectively. As shown as Fig. 12, the similarity is defined according to the ratio between overlapped (δ) and overall $(\alpha \cup \delta \cup \beta)$ area of $\mathcal{A}$ $(\alpha \cup \delta)$ and $\mathcal{B}$ $(\beta \cup \delta)$, where α, β, and δ represent the shaded areas in Fig. 12.

Since four bitmaps are extracted to represent the color characteristics of an color image, the similarity between two images can be estimated based on the group similarity S_g between two group of bitmaps. The computation of group similarity can be performed on a best match between the elements in bitmap groups. The best match can be searched by the method proposed in Section 4.3. According to the best match, the group similarity measurement S_g is defined as follows:

$$S_g = \sum_i \frac{\|\mathcal{A}_i\| + \|\mathcal{B}_i\|}{\sum_j (\|\mathcal{A}_j\| + \|\mathcal{B}_j\|)} \times S(A_i, B_i),$$

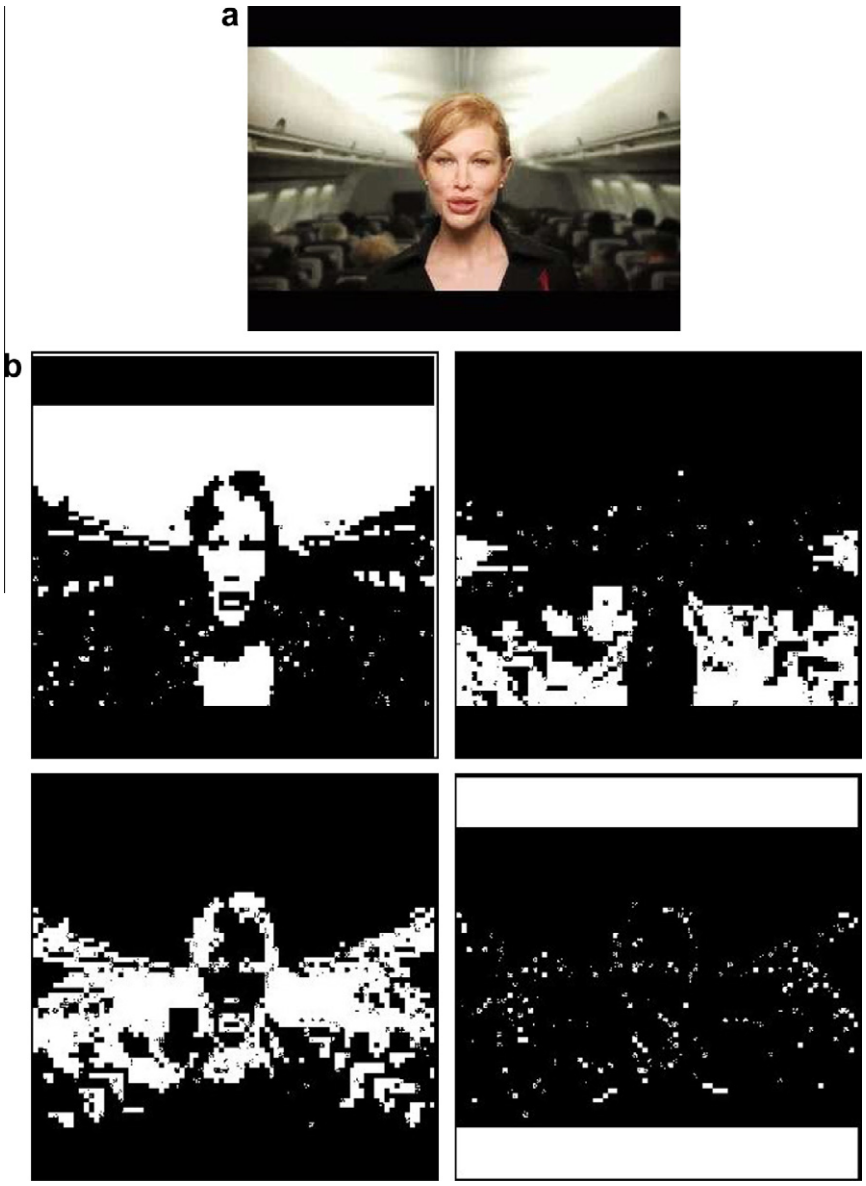

Fig. 11. Using K-means algorithm, pixels of original image, for example in (a), are clustered into four dominant groups according its color value. Then, four bitmaps, for example (b), are established to show the data distribution of pixels in each dominant groups.

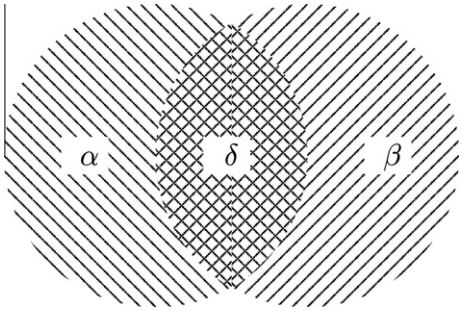

Fig. 12. Giving two coverage area of data distribution α and β, the similarity between these two coverage area is defined according to how much portion of these two coverage area are overlapped (δ).

where $\mathcal{A}_i$ and $\mathcal{B}_i$ are the sets containing the pixels with value 1 in bitmap A_i and B_i, respectively. That is, the group similarity measurement S_g is the weighted sum of similarities between each pair of bitmaps, among the total number of pixels in each pair of bitmaps.

4.3. Similarity matching between bitmap groups

For each image, four bitmaps are extracted to represent the color characteristics of an image. To measure the similarity between two images, a one-to-one and onto match (Liu, 1985) between elements of two bitmap groups can be performed by the following proposed method.

Giving two bitmap groups $\mathcal{A} = \{A_1, \ldots, A_N\}$ and $\mathcal{B} = \{B_1, \ldots, B_N\}$, where A_i and B_i are bitmaps. An $N \times N$ matrix M is formed in the following manner:

$$M = \begin{bmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \ddots & \vdots \\ m_{N1} & \cdots & m_{NN} \end{bmatrix}, \quad \text{where } m_{ij} = S(A_i, B_j).$$

Since an element m_{ij} represents the similarity value when A_i and B_i are considered to be matched with each other. Finding a one-to-one and onto match between $\mathcal{A}$ and $\mathcal{B}$ is equivalent to selecting N elements from each column of M, and no two elements are selected from the same row. To have the largest overall similarity $S_g(\mathcal{A},\mathcal{B})$ of a one-to-one and onto match between two bitmap groups $\mathcal{A}$ and $\mathcal{B}$, the N elements should be selected such that the sum of selected elements is maximized. In order to regularize the computing procedure, row exchanges are performed to have the N selected elements aligned on the diagonal position. Thus, the similarity value between bitmap groups can be computed as the sum of diagonal elements.

Fig. 13 shows the flow chart of the proposed method that maximizes the sum of diagonal elements by properly exchanging rows. Giving a similarity matrix M. First, find an element m_{ij} with the largest value in a similarity matrix M. Then, swap the ith row and jth row such that the element m_{ij} is at (j,j). Then, mark all elements in jth row and jth column. Then, find elements with maximal value from un-marked elements and swap the corresponding raw and column, such that the maximal element is moved to diagonal position. Repeat the same process, until all elements are marked.

Although the procedures stated above are efficient, its results may not be good enough. Therefore, a checking step is applied on each rows and columns to see whether any possible row exchange can further maximize the sum of diagonal elements in the similarity matrix M. The checking step is described as follows:

> IF $m_{ii} + m_{jj} > m_{ij} + m_{ji}$
> THEN exchange the ith and jth rows.

This checking step is performed repeatedly until no further row exchange can be performed to increase the sum of diagonal elements. In other words, the near maximum similarity value between bitmap group $\mathcal{A}$ and $\mathcal{B}$ is achieved.

An example of the one-to-one onto match method is shown in Fig. 14. At first, since $m_{11} = 178$ is the largest value in the 5×4 matrix, the elements in first row and first column are marked. Since the value $m_{23} = 169$, the largest value among the unmarked elements is located in the third row, and the second row and the third row are exchanged. Then, the elements in second row and second column are marked too. Similar process is repeated until all columns are marked. According to the checking step, the second and the fourth rows are exchanged, since 80 + 143 > 60 + 162.

In order to evaluate the performance of the proposed one-to-one and onto match algorithm, synthesis testing data sets are generated by a random number generator. To evaluate different size of similarity matrices, 10 groups of test data sets with row dimension from 2 to 11 are generated. For each group, random seeds from 0 to 5999 are used to generate test data.

The testing results are listed in Table 1. Label OST (optimizing sorting test) is performed by the proposed method, and label TAT (Testing All Test) indicates an exhaustive search method that validates all the possible solutions. The TAT results are considered as

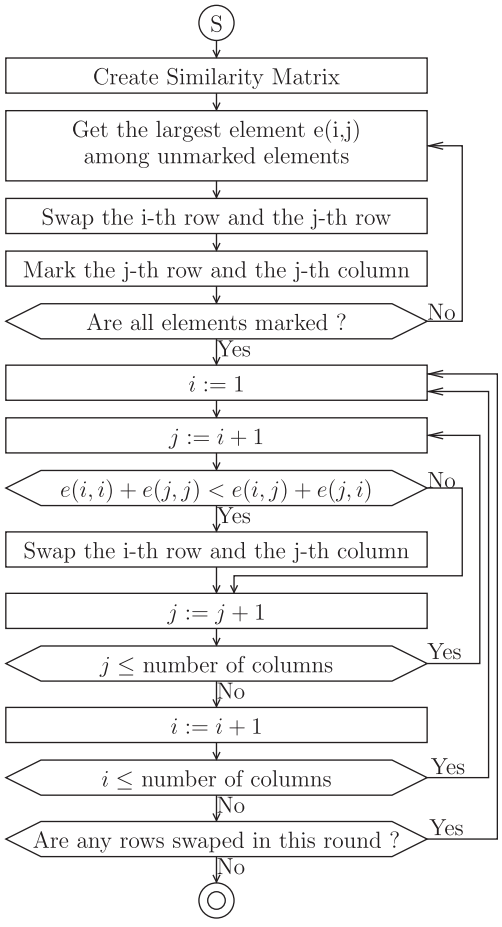

Fig. 13. The flow chart of the proposed matching method. At the beginning, a similarity matrix is constructed. The order of rows is then initialized by greedily exchanging the largest element to diagonal position. Then, further checking steps are applied on the diagonal elements to maximize the sum of diagonal elements.

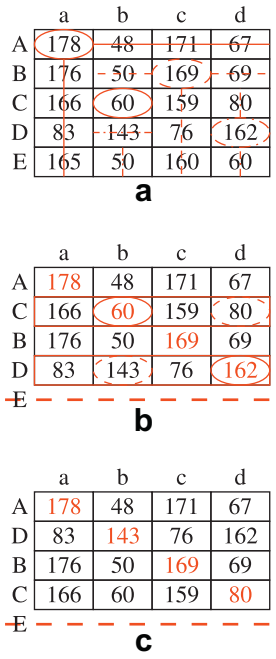

Fig. 14. An example of the proposed matching method. The similarity matrix that is constructed from the pair-wise similarity values is shown in (a). At first, by selecting an element with the largest value among unmarked elements in a similarity matrix, swapping the selected element to diagonal position, and marking the elements in the same row and column iteratively until all elements are marked. The result is shown as (b). Then, the checking steps are applied to see whether any row exchanges can be applied to maximize the sum of diagonal elements. The final results are shown in (c).

Table 1
The performance comparison of the proposed one-to-one match algorithm (OST) and the exhaustive search method (TAT). To evaluate different size of similarity matrices, 10 groups of test data sets with row dimension from 2 to 11 are generated. The data listed in table are the averaged computing time in milliseconds spent by the matching process. TAT/OST is the ratio of the spending time by these two methods.

Matrix dimension	Computing time (ms)		TAT/OST
	TAT	OST	
2	206	175	1.2
3	295	214	1.4
4	855	216	4.0
5	5440	228	23.9
6	40,853	232	176.1
7	349,117	255	1369.1
8	3,582,596	301	11902.3
9	39,197,125	327	119868.9
10	469,318,748	362	1296460.6
11	6,219,635,476	408	15244204.6

the ground truth. The performance is measured in milliseconds of the total computing time to process all the test data. Apparently, the proposed method is quite efficient. Besides, based on the 60,000 (10 group of 6000 random seeds) testing data, the proposed method generates the same results as the ground truth.

4.4. Images clustering by enhanced K-medoid method

Giving an image set $\mathcal{I} = \{I_1, \ldots, I_N\}$, a matrix $\mathcal{M}$ is constructed by measuring the similarity between two images using the methods proposed in Sections 4.1, 4.2, and 4.3. The image similarity matrix $\mathcal{M}$ for image comparison is defined as follows:

$$\mathcal{M} = \begin{bmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \ddots & \vdots \\ m_{N1} & \cdots & m_{NN} \end{bmatrix}, \quad \text{where } m_{ij} = S_g(I_i, I_j).$$

Then, the proposed *variance enhanced K-medoid* model clusters a set of images using the pair-wise image similarities S_g represented in the image similarity matrix $\mathcal{M}$. The image corresponding to the medoid of a data cluster is selected as the representative image for each image cluster.

5. System demonstration

A web-based prototype system (http://www.csie.nctu.edu.tw/pslai/ASKMDDemo) is implemented to demonstrate the proposed method for public trial and testing. Fig. 15 shows a web-based user interface for image clustering. As shown in Fig. 15, a user may select a test media (http://www.youtube.com/watch?v = MgpzUo_kbFY) set from the selector at the top-right corner of the web interface. The threshold, which represents the maximal allowed difference between data objects and its cluster center, is selected by clicking at a proper position on the ruler at the top-left corner. Then, the images in the test media set are clustered. And, the resulted clusters are presented in the order of cluster size. For each cluster, the represented image of each cluster is displayed in larger size, and the rest of images in a cluster are shown in smaller-sized thumbnail.

Figs. 15–17 show the testing results using three different thresholds on the same media data set. In general, using higher threshold may reduce the number of clusters. More specifically, enlarging similarity threshold may reduce the similarity among data elements in certain clusters, so as to receive more *not so similar* data elements in a cluster. That is, the maximum allowed variance is increased. However, for data clusters containing quiet similar data elements, enlarge the similarity threshold will only slightly increase the number of less similar data elements, thus the maximal allowed variance has little inference on these data clusters. In addition, noisy data elements are clustered into certain less-similar clusters instead of spreading to all clusters.

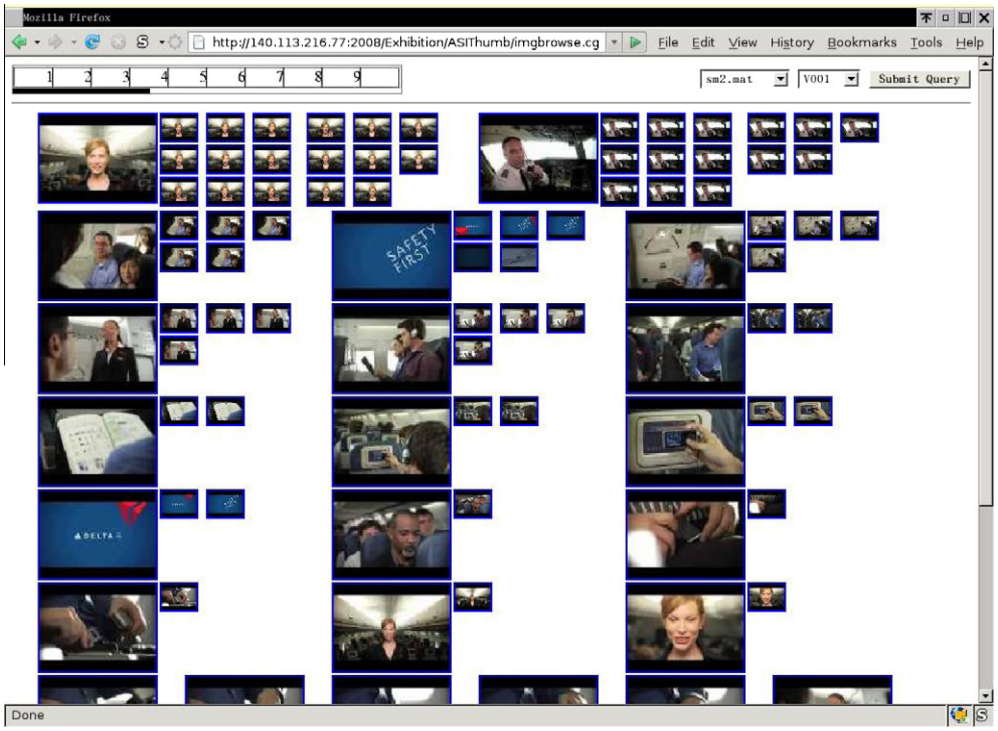

Fig. 15. According to the web-based image clustering prototype system, the proposed method successfully clusters similar images together. In this demonstration, the similarity threshold is set as 0.35. That is, the similarity between a data object and the representative object (cluster center) of a cluster should be larger than 0.65(=1 − 0.35).

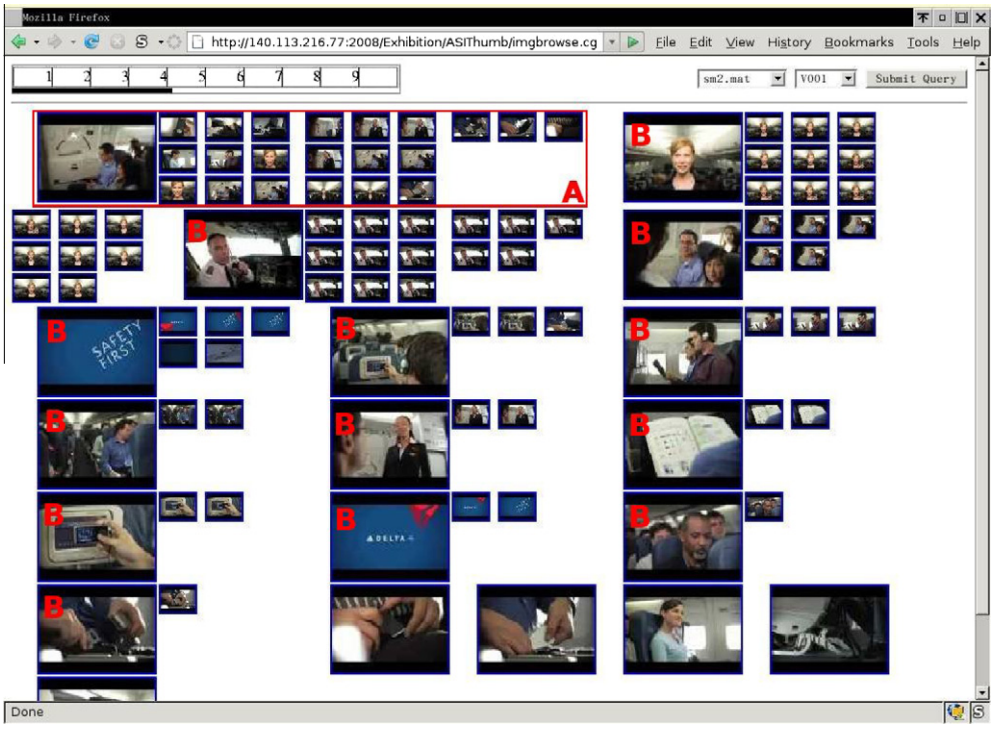

Fig. 16. By increasing the similarity threshold to 0.4, certain data clusters receive several *not so similar* data objects (red frame A). However, the data clusters, which are previously contained quiet similar data elements, still have almost the same similarity among each other (clusters marked with B). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

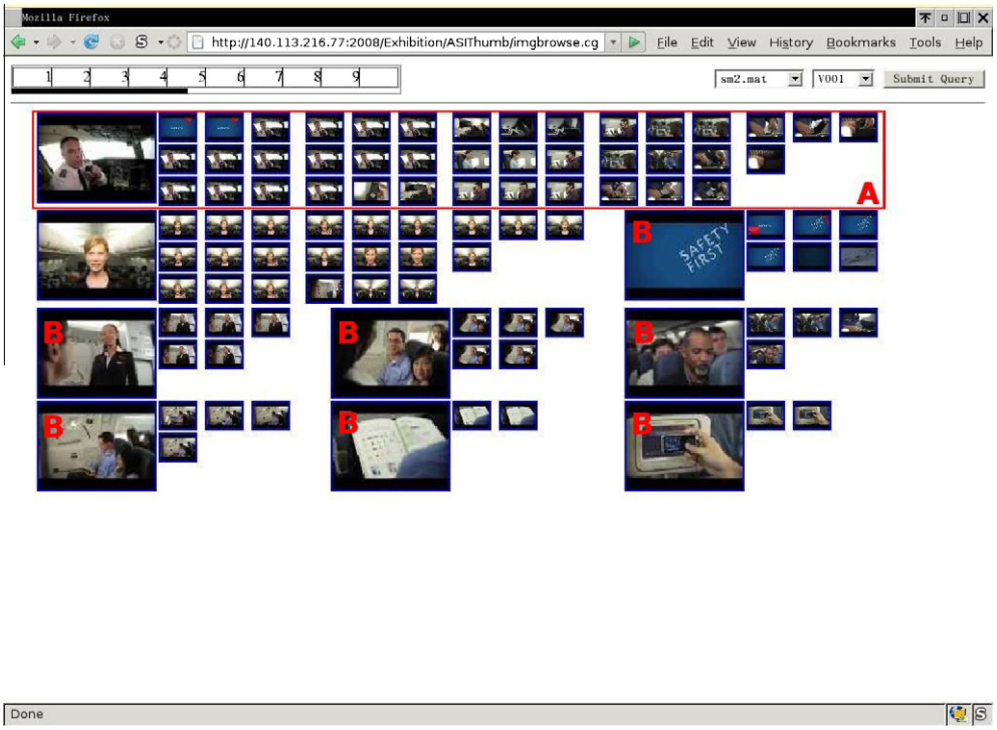

Fig. 17. By further increasing the similarity threshold to 0.45, data clusters previously contained some *not so similar* objects will grow to accept more dis-similar objects (red frame A). However, these data clusters, which previously contained quiet similar objects, will still have very similar objects (clusters marked with B). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

6. Conclusion

Generally speaking, grouping similar data together is useful to help user browse data efficiently. However, the similarity of data are difficult to define. Often, the definition of similarity between data objects depends on their applications. On the other hand, most clustering algorithms are developed for data sets which are represented as data points in feature space. That is, before

clustering similar data, a feature extraction process is applied to each data objects to represent data objects as points in feature space. However, since the definition of similarity is different for various applications, and mapping the data objects into a feature space reasonably can be difficult and also complicated. Most of the time, proper feature extraction becomes a bottleneck for application system development.

Instead of mapping data objects into points in feature space, another way to show the similarity among data objects is to measure the pairwise similarity among data objects. Describing the similarity between two data objects is more instinctively than mapping data objects into a feature space since computing similarity between pair of data objects only need the special relationship between data objects.

A popular clustering algorithm, called K-medoid, is often used to cluster data objects using the pairwise similarities between data objects. K-medoid clustering algorithm groups data objects into clusters by maximizing the sum of similarity between each data objects and their cluster center. That is, the variance of each cluster still has the same value.

In this paper, variance enhanced K-medoid clustering is proposed to introduce the idea of data variance into the original K-medoid algorithm. Besides, multiple variances are used to describe data variances along line segments between cluster centers. That is, the variance of cluster in different orientation is considered.

An intelligent image thumbnail system is prototyped to demonstrate the proposed variance enhanced K-medoid clustering method. For each image, the covered regions of four dominant colors are segmented at first. Between two images, the differences in four dominant color regions are measured as the similarity between images. Then, the variance enhanced K-medoid model is applied on these pairwise image similarities to group images into clusters. And, the image at cluster center (medoid) is used as the key image for this group. The web-based thumbnail demonstration show that the system successfully groups similar images together, and the key image is a good representative of the images in the cluster. By increasing the similarity thresholds of the proposed clustering method, the number of clusters will decrease. And, the clusters that contained quiet similar images before the changes will only changes slightly, i.e., receive a few more images, although the cluster that contained some *not so similar* images will grow bigger due to more *outlier images* are received. Such properties of the proposed model are useful such as selecting representative frame-sets from video for various video summary applications.

References

Bilmes, J. (1997). *A gentle tutorial on the em algorithm and its application to parameter estimation for Gaussian mixture and hidden markov models*, Tech. rep., ICSI-TR-97-021, University of Berkeley.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603–619.

Fu, H.-C., Lai, P.S., Lou, R.S., & Pao, H.T. (2000). Face detection and eye localization by neural network based color segmentation. In: *Neural networks for signal processing X, 2000. Proceedings of the 2000 IEEE signal processing society workshop* (Vol. 2, pp. 507–516).

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157–1182.

Hastie, T., Tibshirani, R., & Friedman, J. (2001a). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2001b). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2001c). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2001d). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation*. Macmillan College Publishing Company, Inc..

Lai, P.-S. & Fu, H.-C. (2007). A polygon description based similarity measurement of stock market behavior. In: *IEEE congress on evolutionary computation, 2007 (CEC 2007) Singapore* (pp. 806–812).

Lai, P.-S., & Fu, H.-C. (2010). A polygon description based similarity measurement of stock market behavior. *Expert System with Applications, 37*(2), 1113–1123.

Liu, C. L. (1985). *Elements of discrete mathematics*. New York: McGraw-Hill. Ch. 4, p. 126.

Pao, H., Chen, Y., Lai, P., Xu, Y., & Fu, H.-C. (2008). Constructing and application of multimedia tv-news archives. *Expert Systems with Applications, 35*, 1444–1450.

Pao, H., Chuang, S., Xu, Y., & Fu, H.-C. (2008). An em based multiple instance learning method for image classification. *Expert Systems with Applications, 35*, 1468–1472.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 80–91.

The demonstration of variance enhanced K-medoid model. Available from http://www.csie.nctu.edu.tw~pslai/ASKMDDemo/.

The test video. Available from http://www.youtube.com/watch?v=MgpzUo_kbFY.