Applying the extended mass-constraint EM algorithm to image retrieval

Daan He\textsuperscript{a,}\textsuperscript{*}, Nick Cercone\textsuperscript{b}, Zhenmei Gu\textsuperscript{a}

\textsuperscript{a} Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada
\textsuperscript{b} Faculty of Science and Engineering, York University, Toronto, ON, Canada

Received 19 July 2007; accepted 27 July 2007

Abstract

We extend the mass-constraint data clustering and vector quantization algorithm to estimate Gaussian Mixture Models (GMMs) as image features applying to the image retrieval problems. The GMM feature is an alternative method to histograms to represent data density distributions. Histograms are well known for their advantages including rotation invariance, low calculation load, and so on. The GMM maintains the rotation invariance properties; moreover, it addresses the high-dimensional problems due to which histograms usually suffer inefficiency problems. The extended mass-constraint (EMass) GMM estimation algorithm is compared with the typical Expectation–Maximization (EM) algorithm, and the deterministic annealing EM (DAEM) algorithm. The three algorithms are applied to train a GMM for a set of simulation data, and compared with the log-likelihood values. From the comparison results, we know that DAEM still has strong dependence on initial data point selection, which is the main problem we need to solve by taking advantage of the deterministic annealing methods. Thus the DAEM algorithm is not chosen to estimate GMM density functions for image retrieval. The EM and EMass algorithms are then applied to train GMMs from image RGB color features for the purpose of image retrieval. Finally the GMM features are combined with the Local Binary Pattern (LBP) features to achieve higher precision retrieval. After we compare the precision/recall curves and mean average precisions achieved by two algorithms, we conclude that the extended mass-constraint algorithm is a better solution for GMM estimation, and combining the GMM and Local Binary Pattern (LBP) provides a new promising feature for image retrieval.

\textcopyright{} 2008 Elsevier Ltd. All rights reserved.

Keywords: Image retrieval; Gaussian mixture model; EM; Deterministic annealing

1. Introduction

Image retrieval is important for a number of applications, such as publishing and advertising, multimedia database management systems and so on [1]. Image retrieval encompasses two basic steps: image indexing, and matching. We are usually given a query image and an image database with all target images. Indexing is to extract image features from each image, both query and target images. Low-level image features, such as color, texture, and shape features, are often used for retrieval matching purposes. Central to an image retrieval system is image matching, where the query image is matched to several target images in the image database. The matching criterion is based on a term: similarity, which evaluates how similar the retrieved target images and the query image are. Similarity between two images is
calculated based on the extracted indexing features. Histograms [2] are one of the most popular features for images. The histogram provides advantages such as low computation load, and rotation invariance. Grey level histogram is a small size feature, and thus retrieval applications by grey level histograms are very fast. Color histogram, however, usually is in high dimension, and retrieval is comparatively much slower. The Gaussian Mixture Model (GMM) [3–6] is introduced to approximate the histograms features, while the feature space is compact with components’ weights, means and covariance matrix.

Using GMMs to represent features has been widely followed for different applications in many research areas. The Expectation–maximization (EM) algorithm [7] is the dominant algorithm for estimating GMMs. The EM algorithm iteratively updates GMM parameters by maintaining a non-decreasing likelihood function. The EM algorithm does not guarantee to obtain the global maximum, because usually the likelihood function is not strictly concave. Under such cases, the non-decreasing EM algorithm will result in either a local maximum or a global maximum depending on the initial values.

In order to estimate a better maximum, deterministic annealing algorithm is applied to estimate the GMMs from data. The deterministic annealing method was first introduced for data clustering and vector quantization [8]. The deterministic annealing EM (DAEM) algorithm has been proposed for GMM parameter estimation problems [9]. In this paper, we make a comparison between the EM and DAEM algorithms for data modeling. Based on the comparison results, we conclude that DAEM is not an appropriate solution for data modeling by GMMs. We extend the deterministic annealing clustering and vector quantization algorithm in [8] for GMM estimation. The extended algorithm is compared with the EM algorithm by likelihood, and both are applied to and tested for image retrieval.

This paper is organized as follows. We briefly review how to apply the EM and DAEM algorithms to Gaussian mixture model estimation in Section 2. The extended mass-constraint EM algorithm is introduced in Section 3. Experimental results of each algorithm are shown in Section 4. Conclusions and ideas for future work arising from the paper are addressed in Section 5.

2. EM and DAEM

Assume we have a random variable \( Y \) that represents the grey level distribution in an image. Given a sequence of observations \( \{y_1, \ldots, y_N\} \), where \( N \) is the sample size, and \( y_i \) is an observation of random variable \( Y \), we can easily calculate the sample statistics, e.g. sample mean and sample variance, as

\[
\mu = E[Y] = \frac{\sum_{i=1}^{N} y_i}{N}
\]
\[
\sigma^2 = E[(Y - \mu)^2] = \frac{\sum_{i=1}^{N} (y_i - \mu)^2}{N}
\]

where \( \mu \) is the mean, \( \sigma^2 \) is the variance, and \( E[.] \) is the expectation function.

The log-likelihood function \( L(Y) \) on the given grey level data \( Y \) is defined as

\[
L(\Phi) = \log g(y; \Phi) = \log \prod_{i=1}^{N} g(y_i; \Phi) = \sum_{i=1}^{N} \log g(y_i; \Phi) \tag{1}
\]

where \( g(.\;) \) is a density function that describes the grey level data distribution in the image, and \( \Phi \) is the function parameter set.

Our problem is how to estimate a set of parameters \( \Phi \) of the density function \( g(y; \Phi) \) that best models the given image data. One of the popular solutions is to estimate the density function by maximizing the log-likelihood function. Assume that the \( g(y; \Phi) \) is a Gaussian distribution, and thus \( \Phi \) consists of two parameters, the mean \( \mu \) and variance \( \sigma^2 \). We know that the set of parameters of the likelihood maxima is the sample mean and variance.

However, if we examine the grey level histogram, we may notice that the histogram usually is not a distribution with only one Gaussian component, but a distribution with several Gaussian components mixed together. For example, an image with two colors may have two peaks in the histogram, which is modeled better by a GMM with two
Gaussian components. One solution is to introduce a random variable $Z$ to indicate from which Gaussian component an observation $y_i$ of $Y$ is generated. The data are noted as $X = \{Y, Z\} = \{y_1, z_1, \ldots, y_N, z_N\}$, where $z_i$ is a vector of length $K$, and $K$ is the total number of components in the GMM. $z_i(k), k = 1, \ldots, K$ is a value between 0 and 1, indicating that $y_i$ is decided by the $k$th component with weight $z_i(k)$. Note that the summation of all elements $z_i(k), k = 1, \ldots, K$ in $z_i$ is 1. The log-likelihood is defined as:

$$L_c(\Phi) = \sum_{i=1}^N \log f(y_i, z_i; \Phi).$$  \hfill (2)

Our problem remains that the hidden variable $Z$ is not observed. Thus mechanisms to account for hidden variables are necessary to incorporate with the maximum likelihood method [10].

2.1. EM algorithm

The Expectation–Maximization (EM) algorithm [7] is a general maximum likelihood algorithm to estimate a model from incomplete data, i.e., data with missing values, or with latent variables. The EM algorithm calculates the conditional expectation of the log-likelihood of the complete posteriori densities, and updates the model parameters by maximizing the log-likelihood.

Given a set of data in $d$ dimension $Y \in \mathcal{Y}$, and a set of probabilistic density functions $\mathcal{F}$ on $\mathbb{R}^d$. We are looking for a density function $g(y) \in \mathcal{F}$, which best describes the given data. If $g(y)$ is depending on a set of parameters $\Phi$, $g(y)$ is noted as $g(y; \Phi)$.

We are looking for a set of parameters maximizing the log-likelihood

$$L(\Phi) = \sum_{y \in \mathcal{Y}} \log g(y; \Phi)$$ \hfill (3)

where $\mathcal{Y}$ is alphabet of $Y$.

As mentioned above, the information that indicates which component generates each datum is missing. We denote this missing information as $Z \in \mathcal{Z}$, where $\mathcal{Z}$ is the alphabet set as all vectors whose elements are summed to 1. We also use a corresponding $X \in \mathcal{X}$ including the observed $Y$ and missing $Z$. Random variable $X$ is referred to as complete data, $Y$ is incomplete data, and $Z$ is the unobserved data. The complete log-likelihood is defined as,

$$L_c(\Phi) = \sum_{y \in \mathcal{Y}} \sum_{z \in \mathcal{Z}} \log f(y, z; \Phi)$$ \hfill (4)

where $f(y, z; \Phi)$ is the joint density function of the variables $Y$ and $Z$.

The EM algorithm obtains a set of parameters $\Phi$ that maximizes $L(\Phi)$ by making use of the complete log-likelihood. The parameter set is iteratively updated by maximizing the expectation of complete log-likelihood $Q(\Phi'|\Phi)$, where

$$Q(\Phi'|\Phi) = E[\log f(y, z; \Phi')|y; \Phi] = \sum_{y \in \mathcal{Y}} \int_{z \in \mathcal{Z}} \log f(y, z; \Phi') k(z|y; \Phi) dz$$ \hfill (5)

where $\Phi$ is the current parameter, $\Phi'$ is the parameter to be estimated, and $k(z|y; \Phi)$ is the conditional probability function of the hidden variable $z$ given the observed $y$, and denoted as

$$k(z|y; \Phi) = \frac{f(y, z; \Phi)}{\int_{z \in \mathcal{Z}} f(y, z; \Phi) dz}. \hfill (6)$$

Thus the E step and M step of the EM algorithm are:

E step:
$$Q(\Phi|\Phi^{(t)}) = E[\log f(y, z; \Phi)|y; \Phi^{(t)}]$$

M step:
$$\Phi^{(t+1)} = \arg \max_{\Phi} Q(\Phi|\Phi^{(t)})$$ \hfill (7)

where $t$ is the iteration index.
Paper [7] indicates that the likelihood is non-decreasing at each iteration of the EM algorithm, i.e., each iteration of the EM algorithm is a monotonically increasing process. If the expected complete log-likelihood \( L_c(\phi) \) has several maxima, the EM algorithm converges to either a global or local maximum depending on the choice of initial values.

2.2. Deterministic annealing EM algorithm

The deterministic annealing method is applied to avoid local maxima [9]. The maximum likelihood estimation is reformatted as optimizing the expected complete log-likelihood with some constraints. The deterministic annealing EM (DAEM) algorithm defines the problem as one of maximizing the log-likelihood,

\[
Q(\phi'|\phi) = E[\log f(y, z; \phi')|y; \phi] = \sum_{y \in Y} \int_{z \in Z} \log f(y, z; \phi) k(z|y; \phi^{(t)}) dz
\]

and also maximizing entropy

\[
H = -\sum_{y \in Y} \int_{z \in Z} k(z|y; \phi) \log k(z|y; \phi) dz
\]

under the constraint:

\[
\int_{z \in Z} k(z|y; \phi) dz = 1.
\]

The optimization is to maximize the Lagrangian

\[
F = Q + TH + \lambda \left( \int k(z|y; \phi) dz - 1 \right)
\]

where \( T \) and \( \lambda \) are Lagrange multipliers.

By maximizing the Lagrangian with respect to the density function \( k(z|y) \), we have

\[
k(z|y) = \frac{1}{Z} f(y, z; \phi)^{\frac{1}{T}}
\]

(8)

where \( Z \) is the partition function that satisfies

\[
Z = \int_{z \in Z} f(y, z; \phi)^{\frac{1}{T}} dz.
\]

The DAEM algorithm is:

1. Set limits: Maximum parameter \( T_{\text{max}} \) and minimum \( T_{\text{min}} \).
2. Set \( T = T_{\text{max}} \);
3. Set \( \Phi^{(0)} \), and \( t = 0 \);
4. Iterate the E and M steps until convergence.

E-step:

\[
U_T(\phi|\phi^{(t)}) = E[-\log f(y, z; \phi)|y; \phi^{(t)}]
\]

(9)

M-step

\[
\phi^{(t+1)} = \arg \max_{\phi} U_T(\phi|\phi^{(t)}).
\]

(10)

Set \( t = t + 1 \);

5. Decrease \( T = \alpha T, 0 < \alpha < 1 \);
6. If \( T > T_{\text{min}} \), goes back to step 5; otherwise stop.

Two special cases of \( T \) are \( + \infty \) and 1. If \( T = + \infty \), the posterior probability is a uniform distribution. If \( T = 1 \), the posterior probability is the same as the EM algorithm. When \( T \) changes from \( + \infty \) to 1, the effects of the original posterior density on the parameter estimation is increasing. We follow such changes because the posterior density is not reliable at the early training stages, but getting stable while we are approaching the true solution.
3. Extended mass-constraint EM algorithm

In this section, GMM estimation problem for a group of data is mathematically defined by minimizing an expected distortion or cost function. The basic idea comes from the mass-constraint algorithm [8], which uses the deterministic annealing method as the DAEM algorithm. However, the mass-constraint algorithm uses different expectation functions with different constraints from the DAEM algorithm.

The mass-constraint algorithm has been applied to data clustering and vector quantization problems [8]. To cluster a group of data, they obtain a list of centroid vectors as the clustering results, with each vector as the centroid of a cluster. To formulate our problem, we consider a GMM as a soft clustering result of a group of data. Each Gaussian component is a cluster, and clusters are overlapped with the nearby clusters. Under this consideration, we extend the mass-constraint algorithm to estimate a GMM density function from the data, other than the cluster centroids by the mass-constraint algorithm. The resulting GMM density function consists of several Gaussian components with their weights, mean vectors, and covariance matrix.

The extended mass-constraint (EMass) algorithm for GMM density estimation is derived as follows. Let \( y \) denote a data vector, and \( g_i \) (\( i = 1 \ldots K \)) denote the Gaussian components to be estimated, where the Gaussian mixture models have at most \( K \) components. We write the expected distortion [8] as:

\[
D = \sum_y \sum_i p(y, g_i) d_{LL}(y, g_i) = \sum_y p(y) \sum_i p(g_i|y) d_{LL}(y, g_i)
\]

where \( p(y, g_i) \) is the joint probability between an observed \( y \) and a Gaussian component \( g_i \), and the conditional probability \( p(g_i|y) \) is the association probability of a component \( g_i \) given observed data \( y \). \( d_{LL}(y, g_i) \) is the distance from a data vector \( y \) to a Gaussian component \( g_i \). Our objective is to minimize the \( D \) subject to a level of randomness \( p(y, g_i) \). The level of randomness is measured by the Shannon entropy

\[
H = -\sum_y \sum_i p(y, g_i) \log p(y, g_i).
\]

The optimization is equivalent to minimizing a Lagrangian

\[
F = D - TH
\]

where \( T \) is the Lagrange multiplier.

Similar to the derivation in [8], two steps are iteratively performed to minimize \( F \) as:

Step 1. Fix the component parameters, and update association probabilities as:

\[
p(g_i|y) = \frac{\exp\left(-\frac{d_{LL}(y, g_i)}{T}\right)}{Z_y}
\]

where \( Z_y \) is the partition function as:

\[
Z_y = \sum_i \exp\left(-\frac{d_{LL}(y, g_i)}{T}\right).
\]

Step 2. Fix the association probabilities, and optimize the component parameters according to

\[
\frac{1}{N} \sum_y p(g_i|y) \frac{d}{dg_i} d_{LL}(y, g_i) = 0
\]

\[
d_{LL}(y, g_i) = \frac{1}{2} \left(d \log(2\pi) + \log |v_i| + (y - m_i)' v_i^{-1} (y - m_i)\right)
\]

where \( d \) is the dimension of vector \( y \), \( m_i \) is mean, and \( v_i \) is covariance matrix of the component \( g_i \).

3.1. A small change in the mass-constraint algorithm

Unlike clustering or vector quantization [8], which returns a representative or mean vector for each group, the GMM estimation returns parameters with both the mean vectors and the covariance matrix. In many cases, the covariance
matrix encounters singular problems, which make the algorithm fail in an unfinished estimation result. The singular matrix happens when all elements in a component are too similar to each other, which makes the component narrow in a small gap, and more like a constant. We propose to avoid this problem by singular checking at each iteration, maintaining a list with all singular matrix components, removing all singular component related data, and updating the GMM components parameters with the remaining data.

The EMass algorithm for estimating the Gaussian mixture density function is as follows.

1. Input: data $y_{n \times p}$, where $n$ is the number of all data vectors, $p$ is the dimensionality of data vectors, the number of mixture components is $K_{\text{max}}$, and the minimum temperature is $T_{\text{min}}$

2. Initialize: $T > 2\lambda_{\text{max}}(C_y)$, $K = 1$, $m_1 = \sum_y y p(y)$, $v_1 = \text{cov}(y)$, and $p(g_1) = 1$.

3. Update for $i = 1, \ldots, K$

   \[
   m_i = \frac{\sum_y y p(y) p(g_i | y)}{p(g_i)}
   \]

   \[
   v_i = \frac{\sum_y (y - m_i)^T (y - m_i) p(y) p(g_i | y)}{p(g_i)}
   \]
Fig. 2. Simulation experimental results by the DAEM algorithm. Left: $\beta$ starts from 0.3; right: $\beta$ starts from 0.6.

Fig. 3. Sample images from each category in Wang's image dataset.

Fig. 4. Precision-recall graph of the retrieval by Gaussian mixture models. Left: RGB GMM by EM or EMass algorithms; Right: combining LBP with RGB EMass features.
Table 1
Differences among the EM, DAEM, EMass algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Object function</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM</td>
<td>Expected complete log-likelihood</td>
<td>None</td>
</tr>
<tr>
<td>DAEM</td>
<td>Expected complete log-likelihood</td>
<td>Posterior probability entropy</td>
</tr>
<tr>
<td>EMass</td>
<td>Expected log-likelihood</td>
<td>Joint probability entropy</td>
</tr>
</tbody>
</table>

where

\[ p(g_i | y) = \frac{p(g_i) e^{(-d_{LL}(y, g_i) / T)}}{\sum_{j=1}^{K} p(g_j) e^{(-d_{LL}(y, g_j) / T)}} \]

\[ d_{LL}(y, g_i) = \frac{1}{2} (d \log(2\pi) + \log |v_i| + (y - m_i)'v_i^{-1}(y - m_i)) \]

\[ p(g_i) = \sum_y p(y) p(g_i | y). \]

(4) Singular check on covariance matrix \( v_i, i = 1 \ldots K \). If \( v_i \) is singular, save \( i \)th component to singular component list, and decrease \( K \) and \( K_{max} \). The weights, mean vectors, covariance matrix are reset to initial values, and the algorithm goes back to the beginning and starts over again.

(5) Convergence test, if not passed go to step (3).

(6) If \( T \leq T_{min} \), STOP.

(7) Cooling step: \( T = \alpha T, 0 < \alpha < 1 \).

(8) If \( K < K_{max} \), check condition for phase transition for \( i = 1, \ldots, K \). If critical \( T \) [8] is reached for component \( g_j \), we add a new component with parameters set as, \( m_{K+1} = m_i + \delta, v_{K+1} = v_i \), \( p(g_{K+1}) = p(g_j) / 2, p(g_j) = p(g_j) / 2 \), and increment \( K \).

(9) Go to step (3).

The EMass algorithm differs from the mass-constraint algorithm at the fourth step. We add a singularity check step before updating the parameter set to the next iteration round. If a singular component is detected, we save it in a list, and update the data by removing all data related to the singular component. All parameters are reset to initial value, and the algorithm goes back to the beginning to train a model based on the remaining data.

The results obtained from the EMass algorithm are a Gaussian mixture model, together with several singular components if existing. As either the control parameter \( T \) or the number \( K_{max} \) is decreased at each round of iteration, the algorithm will converge to a result while \( T \) and \( k \) both reach the predefined \( T_{min} \) and \( K_{max} \).

3.2. Three algorithms

The three algorithms, EM, DAEM, and EMass are introduced for estimating a GMM from a group of data. Their differences lie in two aspects listed in Table 1.

The first aspect is that the three algorithms have different objective functions. The EM algorithm maximizes the expected complete likelihood function to obtain the resulting model parameters. DAEM also maximizes the expected log-likelihood function, with the constraint of maximizing the posterior probability entropy function. EMass algorithm maximizes the expected log-likelihood, with the constraint of maximizing the joint probability entropy function. The second difference is that different constraints are applied while maximizing or minimizing the objective functions. From Table 1, we can see that the EM algorithm is a pure maximum likelihood method, while the DAEM and EMass algorithms apply different constraints on the objective function to avoid local maxima problems. We will compare their performance in the following experiment section.
4. Experiments

4.1. Simulations

In order to compare how the EM, DAEM, and EMass algorithms work, we first test the three algorithms by simulation experiments. The simulation test data are 1600 two-dimensional data generated by a GMM with four Gaussian components. The true values of the GMMs, initial parameters and trained results are shown in Table 2, and Fig. 1.

Fig. 1 shows the resultant GMM with four components by the EM, DAEM, and the EMass algorithms respectively. The data are shown as red dots in the figure, and each GMM component is represented by an eclipse, with the center set by the component mean, and size set by the component covariance matrix. For the results using the EM algorithm in Fig. 1(a), two components of four show the first group of data (group 1), and the other two components cover the remaining three groups of data. Such results are caused by improper initial data with two starting components with mean vectors as \((-1, 0), (0, 0)\). These two components are both closer to the first group of data, and thus the results with the EM algorithm cannot distinguish one component from the other in group 1. This difficulty is the problem that we address by applying annealing methods. For both the DAEM and EMass algorithms result in Fig. 1(b) and (c), each component of the four GMM components shows one group of data.

From the log-likelihood results in Table 2, we can see that the DAEM and EMass algorithms both obtain higher likelihood than the EM algorithm. DAEM achieves the highest log-likelihood in the three algorithms. However, we also note that the DAEM algorithm is very sensitive to the choice of the initial parameter $\beta = 1/T$. The following
two examples in Fig. 2 show that when $\beta$ is starting at 0.3 and 0.6, we obtain less accurate GMMs with lower log-likelihoods. The results show that DAEM has strong dependence on initial parameter selection, which is the main problem we need to solve by taking advantage of the annealing methods. As there is no available prior information to set the initial value of $\beta$, we consider the DAEM is not suitable for estimating the density of the GMM features for image retrieval applications.

4.2. Retrieval results

We test the EM algorithm and EMass algorithms by applying both algorithms to train GMMs from RGB color features for image retrieval. The test dataset (Wang) [11] contains 1000 generic images in 10 categories, with 100 images in each category. These ten categories are: Africa, Beach, Building, Bus, Dinosaur, Elephant, Flower, Horse, Mountain, and Food. Ten sample images selected from each category are shown in Fig. 3. The images are of the size of either $384 \times 256$ or $256 \times 384$.

First, a GMM is trained from each image’s RGB color features by the EM and EMass algorithms. The average log-likelihood for all images by the EM algorithm is $-2.0005e+005$, and the average log-likelihood by the EMass algorithm is $-1.9795e+005$, which is higher than the likelihood by the EM algorithm.

In the retrieval step, each image serves as a query image, and the performance is compared by average precision-recall for all retrievals. Target images similar to the query image are retrieved by calculating and sorting the distance values between the query image GMM and all target images GMMs. Given a query image GMM as $R = \sum_{i=1}^{L_1} p_i r_i$, 

Fig. 6. Example 1: query image, and target images retrieved by the EMass algorithm.
Fig. 7. Example 1: query image, and target images retrieved by the LBP and EMass algorithms.

Table 2
Parameters estimated by EM, DAEM and EMass algorithms in simulation experiments

<table>
<thead>
<tr>
<th>P</th>
<th>True values</th>
<th>Initial</th>
<th>EM</th>
<th>DAEM</th>
<th>EMass</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1$</td>
<td>0.1875</td>
<td>0.25</td>
<td>0.318</td>
<td>0.188</td>
<td>0.189</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>0.3125</td>
<td>0.25</td>
<td>0.347</td>
<td>0.306</td>
<td>0.307</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>0.25</td>
<td>0.25</td>
<td>0.335</td>
<td>0.257</td>
<td>0.254</td>
</tr>
<tr>
<td>$\alpha_4$</td>
<td>0.25</td>
<td>0.25</td>
<td>0.335</td>
<td>0.250</td>
<td>0.250</td>
</tr>
<tr>
<td>$m_1$</td>
<td>(0 0)’</td>
<td>(−1 0)’</td>
<td>(−0.194 0.060)’</td>
<td>(0.013 −0.008)’</td>
<td>(0.014 0.008)’</td>
</tr>
<tr>
<td>$m_2$</td>
<td>(0 5)’</td>
<td>(0 0)’</td>
<td>(0.133 −0.048)’</td>
<td>(−0.093 5.023)’</td>
<td>(−0.105 5.037)’</td>
</tr>
<tr>
<td>$m_3$</td>
<td>(2 7)’</td>
<td>(2 6)’</td>
<td>(0.852 5.934)’</td>
<td>(1.986 7.030)’</td>
<td>(2.019 7.051)’</td>
</tr>
<tr>
<td>$m_4$</td>
<td>(4 0)’</td>
<td>(1 0)’</td>
<td>(4.025 −0.050)’</td>
<td>(4.025 −0.050)’</td>
<td>(4.026 −0.050)’</td>
</tr>
<tr>
<td>$v_1$</td>
<td>(0.5 0)</td>
<td>(1 0)</td>
<td>(0.396 −0.005)</td>
<td>(0.489 −0.029)</td>
<td>(0.487 0.026)</td>
</tr>
<tr>
<td>$v_2$</td>
<td>(0.5 0)</td>
<td>(1 0)</td>
<td>(0.498 −0.007)</td>
<td>(0.496 −0.045)</td>
<td>(0.464 −0.062)</td>
</tr>
<tr>
<td>$v_3$</td>
<td>(0.5 0)</td>
<td>(1 0)</td>
<td>(1.576 1.050)</td>
<td>(0.502 0.061)</td>
<td>(0.447 0.017)</td>
</tr>
<tr>
<td>$v_4$</td>
<td>(0.5 0)</td>
<td>(1 0)</td>
<td>(0.488 0.016)</td>
<td>(0.488 0.016)</td>
<td>(0.484 0.015)</td>
</tr>
<tr>
<td>LL $^b$</td>
<td>−6.1655e+003</td>
<td>−6.0558e+003</td>
<td>−6.0586e+003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$^a$ This group of results give the best results with initial parameter $\beta = 1/T = 0.5$. Note that $\beta$ varies from 0 to 1.

$^b$ LL stands for log-likelihood.
and a target image GMM as $T = \sum_{j=1}^{L_2} q_j t_j$, the distance between the two GMMs are calculated by their approximation likelihood [3]. The likelihood is a weighted summation of the distance for each component in the query GMM to the closest component in the target GMM.

\[
\text{Likelihood} = \sum_{i=1}^{L_1} p_i (\log(r_i(m_l)))
\]

where $m_l$ is the mean of the closest component in the target image GMM to the $i$th component in the query image GMM. $l$ is found by

\[
l^* = \arg \min_j (m_j - m_l)' v_i^{-1} (m_j - m_l)
\]

where $l \in \{1, 2, \ldots, L_2\}$, and $v_i$ is the covariance matrix of the $i$th component in the query GMM.

The retrieval average precision-recall results are shown in the left part of Fig. 4. We compare the GMM results to the retrieval results by the local binary pattern (LBP) features [12], which is a 256-bin histogram with each element in the histogram as the frequency of an LBP level in the image. The LBP is a color-based feature, and its details can be found in [12]. We also propose to interleave the RGB GMMs and LBP equally.

The interleaving is performed by the following steps. Given a query image and all target images in the data set, we first retrieve and sort all target images separately by their GMM and LBP features. Then we fetch the largest distance in each features results, and normalize all the distances into a value between 0 and 1. Thus, we obtain two distance
values for each target image, and we add them together as the final distance to the query image. Finally, we sort all target images by their final distance values. The overall comparison is shown on the right side of Fig. 4.

GMM features by the EM and EMass algorithms are also compared by the mean average precision (MAP) over all retrievals. The MAP is the average precision at which each relevant image is retrieved. Two images are relevant when they are in the same category, and are irrelevant otherwise. The MAP is defined in Eq. (13):

\[
\text{MAP} = \frac{\sum_{r=1}^{N} (\text{pre}(r) \times \text{rel}(r))}{\text{number of relevant documents}}
\]

where \( r \) is the rank of the retrieved image, \( N \) is the total number of all retrieved images, \( \text{rel}(\cdot) \) is a binary function indicating whether the retrieved image at a given rank \( r \) is relevant to the query image, and \( \text{pre}(\cdot) \) is the precision at a given rank \( r \). This method favors the features that can retrieve relevant images earlier. The MAP comparison results are shown in Table 3.

Table 3
MAP comparison results

<table>
<thead>
<tr>
<th>Features</th>
<th>LBP</th>
<th>EM</th>
<th>EMass</th>
<th>LBP&amp;EMass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.3875</td>
<td>0.3888</td>
<td>0.4000</td>
<td>0.4527</td>
</tr>
</tbody>
</table>

Fig. 9. Example 2: query image, and target images retrieved by the EMass algorithm.
From Fig. 4 and Table 3, we observe that the GMM trained by the EMass algorithm achieves better precision-recall and MAP results than the GMMs trained by the EM algorithm. The performance is improved again when we combine the GMM features and LBP features. The LBP features describe the relationship between a pixel with eight neighboring pixels, which provide the texture information. The GMM features provide the color information. The combined feature takes advantage of both the texture and color information in an image, thus achieves better performance than the performance of either method by itself.

4.3. Initial retrieval example

We show two examples to compare the performance of the GMM features trained by the EM and EMass algorithms. Two examples list the top 23 target images in all images that are retrieved given a query image. One query image is a human image, and the other query image is a flower image. In both examples, the query image is shown at the top left corner, and marked by Q. The rest of the images are retrieved target images, and they are ranked in an order from left to right, from top to bottom. A positive image, which means the target image belongs to the same category as the query image, is marked by Y. A negative image is marked by X. We expect that more positive images are ranked higher than negative images. The more positive images we obtain, the better are the retrieval results.

The first set of examples is shown in Figs. 5–7. The query image on the left top corner is a human image in the Africa category. Fig. 5 shows the results by the EM algorithm. Among top 23 retrieved images, we have 15 positive images, and 8 negative images. The first negative image is the 4th retrieved image. Fig. 6 shows the results by the EMass algorithm. Among the top 23 images, we have 15 positive images, and 8 negative images. Compared to the
EM algorithm result, the first negative image is the 7th retrieved image, which is better than the EM algorithm results. In Fig. 7, 22 positive images are retrieved, and the only negative image is at 17th position. Thus, the comparison of three methods shows that EM is the worst, EMass is in the middle, and the combined method is the best one.

The second set of retrieval examples uses a flower image as a query image, and the results are shown in Figs. 8–10. All images are listed in the same way as in the first set of examples. The query image is a red flower image in the Flower category. Fig. 8 shows the results by the EM algorithm. Among these top 23 images, we have 15 positive images, and 8 negative images. The first negative image is the 8th retrieved image. Fig. 9 shows the results by the EMass algorithm. Among the top 23 images, we have 16 positive images, and 7 negative images. Compared to the EM algorithm result, the first negative image is the 12th retrieved image, which is better than the EM algorithm results. From the example, we observe that using RGB GMM features trained by the EM and EMass algorithms retrieve target images with similar red colors. Combined with LBP in Fig. 7, the retrieval is more accurate, as all retrieved 23 images are positive. Thus, the combined algorithm has the best precision performance. The two query images are in different categories, but the comparison results are consistent.

5. Future work

In this paper, we introduced an extended mass-constraint algorithm for training Gaussian mixture models from a set of data. The EMass algorithm has the promising property of avoiding local maxima problems, and achieves a higher log-likelihood than the EM algorithm. The experimental results show that the EMass algorithm also provides higher average precision when applied to image retrieval applications. Our future work will focus on providing detailed performance analysis on the relationship between log-likelihood and precision.

A tighter approximation for calculating the distance between two Gaussian Mixture models is also open for solution. This calculation can be accurately approximated by the Monte Carlo method [13] but with a heavy computational overhead. Both the effectiveness and the efficiency remain as important issues to be solved in the future work in this part.

References