Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Relevance feature mapping for content-based multimedia information retrieval

Guang-Tong Zhou [a], Kai Ming Ting [b], Fei Tony Liu [b], Yilong Yin [a,]*

[a] School of Computer Science and Technology, Shandong University, Jinan 250101, China
[b] Gippsland School of Information Technology, Monash University, Victoria 3842, Australia

ARTICLE INFO

Article history:
Received 22 March 2010
Received in revised form
10 July 2011
Accepted 24 September 2011
Available online 1 October 2011

Keywords:
Content-based multimedia information retrieval
Ranking
Relevance feature
Relevance feedback
Isolation forest

ABSTRACT

This paper presents a novel ranking framework for content-based multimedia information retrieval (CBMIR). The framework introduces relevance features and a new ranking scheme. Each relevance feature measures the relevance of an instance with respect to a profile of the targeted multimedia database. We show that the task of CBMIR can be done more effectively using the relevance features than the original features. Furthermore, additional performance gain is achieved by incorporating our new ranking scheme which modifies instance rankings based on the weighted average of relevance feature values. Experiments on image and music databases validate the efficacy and efficiency of the proposed framework.

1. Introduction

We have witnessed a substantial progress in the acquisition and storage of digital media such as images, video and audio. With the rapid increase of digital multimedia collections, effective and efficient retrieval techniques have become increasingly important. Many existing multimedia information retrieval systems index and search the multimedia databases based on textual information such as keywords, surrounding text, etc. However, the text-based search suffers from the following inherent drawbacks [1,2]: (i) the textual information is usually nonexistent or incomplete with the emergence of massive multimedia databases; (ii) the textual description is not sufficient for depicting subjective semantics since different people may describe the content in different ways; and (iii) some media contents are difficult to be described in words.

To address these problems, content-based multimedia information retrieval (CBMIR) is proposed and has attracted a lot of research interest in recent years [1,3–6]. In a typical CBMIR setting, a user poses a query instance to the system in order to retrieve relevant instances from the database. However, due to the semantic gap [3,4] between high-level concepts and low-level features, the list returned by the initial search may not be good enough to satisfy the user's requirement. Thus, relevance feedback [7,8] is usually employed to allow the user to iteratively refine the query information by labeling a few positive instances as well as negative instances in each feedback round.

The performance of a CBMIR system relies on the accuracy of its ranking results. Thus, ranking is the central problem in CBMIR, and many researchers have endeavored to design a fast and effective ranking method [1,4,5]. A key ingredient in ranking is the measure used for comparing instances in the database with respect to the query. Many existing methods (e.g., [9–11,2]) use distance as the core ranking measure.

This paper presents a novel ranking framework for CBMIR that does not use distance as the ranking measure, which is fundamentally different from the above-mentioned methods. Our framework uses some form of ranking models to produce a relevance feature space. It first builds a collection of ranking models and the output of each model forms a relevance feature. Then, the models are used to map every instance from the original feature space to a new space of relevance features. Finally, the ranking and retrieval of instances, based on one query and relevance feedbacks, are computed in the new space using our proposed ranking scheme, which ranks instances based on the weighted average of relevance feature values.

Our analysis shows that the power of the proposed framework derives primarily from the relevance features and secondarily from the ranking scheme. The framework has linear time and

* Corresponding author. Tel./fax: +86 531 88391367.
E-mail addresses: zhouguangtong@gmail.com (G.-T. Zhou), kaiming.ting@monash.edu (K.M. Ting), tony.liu@monash.edu (F.T. Liu), ylyin@sdu.edu.cn (Y. Yin).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.09.016

space complexities with respect to the database size. The on-line processing time is constant when the number of relevance features is fixed, no matter how many original features are used to represent an instance. These characteristics enable the proposed framework to scale up to large databases. In addition, our framework has a good tolerance to irrelevant features.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces our framework, followed by a detailed description in Section 4. Section 5 reports empirical studies, and Section 6 discusses related issues. Finally, this paper concludes in Section 7.

2. Related work

Many ranking methods employ distance as the core ranking measure [1,4,5]. In the case of retrieval with one query without relevance feedback, the majority of previous works have focused on different variants of distance metrics. The simplest way is to use a single distance metric, e.g., Euclidean distance or Manhattan distance. Here instances that lie near to a given query are ranked higher than instances far away from the query. However, these distance metrics are global measures and they might not produce the best results for all queries. Thus, researchers have investigated distance metrics that can be tailored to each query. For example, based on the manifold ranking algorithm [12], He et al. [9] have proposed the MRBIR method which implicitly learns a manifold metric to produce rankings.

In relevance feedback, the additional information provided by the user offers more flexibility in the design of effective ranking methods. Here the query and positive feedbacks are usually considered as positive instances, and negative feedbacks are negative instances. The refinement can be done in three ways. First, the distance metric for the initial query session can be refined based on pair-wised distance constraints derived from positive and negative instances. Commonly used techniques include distance metric learning [13,14], kernel learning [15], and manifold learning [16,17].

Second, instead of refining the distance metric, we can also tackle the problem by designing appropriate ranking schemes. For example, MARS (Multimedia Analysis and Retrieval System) [18] employs a query-point movement technique which estimates the "ideal query point" by moving it towards positive instances and away from negative ones. The ranking is produced by measuring distance with respect to the ideal query after the movement. Giacinto and Roli [10] proposed the InstRank method based on the idea that an instance is more likely to be relevant if its distance to the nearest positive instance is small, while an instance is more likely to be irrelevant if its distance to the nearest negative instance is small. Qsim [11] advocates ranking instances based on the query-sensitive similarity measure, which takes into account the queried concept when measuring similarities. Note that these methods are all based on some predefined or learned distance metrics.

Third, some methods transform the CBMIR problem into a classification problem, and solve it using classification techniques such as support vector machine [19] and Bayesian method [2]. A representative method called BALAS [2] first estimates the probability density function of positive and negative classes, and then the ranking is produced within a Bayesian learning framework. However, most classification methods are designed to classify instances into a fixed number of classes and are not designed for ranking instances. Thus, the ranking results might be suboptimal.

This paper proposes to rank instances through a new framework that does not require distance calculation—a computationally expensive process. This is fundamentally different from most existing methods. Our framework is able to deal with retrieval tasks with one query as well as in relevance feedback. In contrast, most of the above-mentioned methods were designed to be used in relevance feedback only, e.g., InstRank, Qsim and BALAS.

Note that meta-search [20] employs an ensemble of ranking models for information retrieval. However, this technique aims at improving the retrieval performance by combining the ranking results returned by multiple search engines. This is a different problem from the one we addressed. It is also worth noting that Rasiwasia et al. [21] proposed the query-by-semantic-example method which maps and retrieves instances in a semantic space. Here a set of semantic-level concepts has to be predefined in order to construct the semantic features. On the contrary, the relevance features used in this paper are automatically generated—users do not need to specify them.

3. The proposed framework

Generally speaking, a CBMIR system is composed of four parts [22]: (i) a given multimedia database $\mathcal{D}$; (ii) a query $\mathcal{Q}$; (iii) a model $\mathbb{F}(\mathcal{Q},\mathcal{D})$ to model the relationships between instances in $\mathcal{Q}$ and $\mathcal{D}$; and (iv) a ranking scheme $R(\mathcal{D}|\mathcal{Q})$ which defines an ordering among the database instances with respect to $\mathcal{Q}$. On the other hand, a ranking system consists of three components: (i) a given data set $\hat{\mathcal{D}}$; (ii) a model $\hat{\mathbb{F}}(\hat{\mathcal{D}})$ to model the relationships between instances in $\hat{\mathcal{D}}$; and (iii) a ranking scheme $\hat{R}(\hat{\mathcal{D}})$ which produces an ordering for all the instances in $\hat{\mathcal{D}}$. Ranking in CBMIR are typically provided by distance metrics. In this work, we show an alternative method, that is more suitable for CBMIR, using an ensemble of ranking systems.

Here, we propose to map the database $\mathcal{D}$ from the original d-dimensional feature space $\mathbb{R}^d$ into a new space $\mathbb{R}^t$ to form a new database $\mathcal{D}'$ by using an ensemble of t ranking models, i.e., $\tilde{\mathbb{F}} = [\hat{\mathbb{F}}_1, \hat{\mathbb{F}}_2, \ldots, \hat{\mathbb{F}}_t]$. Each ranking model is regarded as a feature descriptor, and the ranking output is the feature value; for an instance, the t ranking outputs from the t ranking models constitute the new t-dimensional feature vector. Given a query $\mathcal{Q}$, we first map it into the new space to obtain $\mathcal{Q}'$, and then we employ a ranking scheme $R'(\mathcal{D}'|\mathcal{Q}')$ to rank the instances in $\mathcal{D}'$. Note that R' can be any existing ranking scheme. But we propose a new ranking scheme based on the weighted average of relevance feature values to avoid the costly distance or similarity calculation. We show in this paper that the ensemble of ranking models, i.e., $\tilde{\mathbb{F}}$, can be implemented using an anomaly detector called Isolation Forest, or iForest [23].

iForest builds an ensemble of isolation trees (or iTrees) to detect anomalies. Each iTree is constructed on a fix-sized random sub-sample of the given data set. The tree growing process recursively random-partitions the sub-sample along axis-parallel coordinates until every instance is isolated from the rest of the instances or a specified height limit is reached. Each iTree is a ranking model which describes a data profile from the view of the underlying sub-sample and produces a ranking output in terms of path length for any test instance. The ranking output can be interpreted as: a short path length indicates irrelevance to the profile because an instance, which has different data characteristics from the majorities, is easily isolated by a few random partitions; on the other hand, a long path shows relevance to the profile. For anomaly detection tasks, instances identified to be irrelevant to the various profiles modeled by a number of iTrees are deemed to be anomalies, and instances relevant to the profiles are normal points. The algorithms to produce iTree and iForest are provided in Appendix A.

In our framework, we first build an iForest, which is composed of t iTrees, to map instances from the original feature space to the relevance feature space, i.e., $\mathbb{R}^d \rightarrow \mathbb{R}^t$. Different iTrees profile different aspects of the multimedia database. We treat each iTree as a feature descriptor, and the feature value (i.e., path length) is a measure of relevance with respect to the profile modeled by the iTree. The representation of an instance in the new space is a vector of relevance features; hence the name *relevance feature mapping*. To implement $R'(\mathcal{D}'|\mathcal{Q}')$, we have also designed a new ranking scheme based on the weighted average of relevance feature values. We call our framework ReFeat which refers to the retrieval based on *Re*levance *Feat*ure mapping.

4. ReFeat

ReFeat has two stages. The first off-line modeling stage builds an iForest to perform relevance feature mapping and the second on-line retrieval stage ranks instances with respect to the query. We first describe the two stages in the next two subsections, followed by explaining why our ranking scheme works in Section 4.3. We then provide our treatment for relevance feedback in Section 4.4. The algorithmic complexity is analyzed in the last subsection.

4.1. Off-line modeling and relevance feature mapping

In off-line modeling, we build an iForest from the given database $\mathcal{D}$. Here t iTrees are constructed, each built on a sub-sample of randomly selected ψ instances from $\mathcal{D}$. After iForest is built, $\mathcal{D}$ is mapped to $\mathcal{D}'$ as follows.

Let $\ell_i(\mathbf{x})$ denotes the path length of an instance $\mathbf{x} \in \mathcal{D}$ on an iTree T_i ($i \in \{1,2,\ldots,t\}$). We map $\mathbf{x}$ to the relevance feature space as: $\mathbf{x}' = [\ell_1(\mathbf{x}), \ell_2(\mathbf{x}), \ldots, \ell_t(\mathbf{x})]^T$. All the instances in $\mathcal{D}$ are mapped through the relevance feature mapping to form a new database $\mathcal{D}' = \{\mathbf{x}' | \forall \mathbf{x} \in \mathcal{D}\}$. Note that this stage does not require any user intervention. Thus, $\mathcal{D}'$ is generated off-line to accelerate the following on-line retrieval process.

4.2. On-line retrieval with one query

Given a query instance $\mathbf{q}$, ReFeat maps it to $\mathbf{q}' = [\ell_1(\mathbf{q}), \ldots, \ell_t(\mathbf{q})]^T$. To retrieve instances relevant to $\mathbf{q}$, we first assign a weight to each feature due to $\mathbf{q}$: a high weight is assigned to a feature which signifies that $\mathbf{q}$ is relevant to the profile modeled by the feature; otherwise, a low weight is assigned. Then the ranking score for every instance in the database is computed using a weighted average of its relevance feature values. The instances having the highest scores are regarded to be the most relevant to the query. To implement this, we define a weight for feature i as:

$$w_i(\mathbf{q}) = \frac{\ell_i(\mathbf{q})}{c(\psi)} - 1. \tag{1}$$

$c(\psi)$ is a normalization term which estimates the average path length of a ψ-sized iTree. The $c(\cdot)$ function is defined as follows [23]:

$$c(n) = \begin{cases} 2(\ln(n-1) - (n-1)/n + E) & \text{if } n > 1, \\ 0 & \text{if } n = 1, \end{cases} \tag{2}$$

where $E \approx 0.5772$ is the Euler's constant.

Finally, the ranking score of an instance $\mathbf{x}$ with respect to the query $\mathbf{q}$ is given by the weighted average of feature values:

$$Score(\mathbf{x}|\mathbf{q}) = \frac{1}{t} \sum_{i=1}^{t} (w_i(\mathbf{q}) \times \ell_i(\mathbf{x})). \tag{3}$$

Eq. (3) gives high scores to instances which have long path lengths on many highly weighted features induced by the query; and it produces low scores to instances which have short path lengths on many lowly weighted features. $Score(\mathbf{x}|\mathbf{q})$ can be negative. If required, strictly positive scores can be produced by using an exponential mapping. For the rest of this paper, we refer to the ranking based on the weighted average of feature values as our ranking scheme.

It is worth noting that the off-line modeling of iForest utilizes no distance or similarity measure [23], and the proposed on-line ranking scheme also avoids distance or similarity calculation through Eqs. (1) and (3). This characteristic differentiates ReFeat from most existing methods which are based on certain distance or similarity measures.

4.3. Understanding the ranking scheme

Our ranking scheme is based on the idea that similar instances share many relevance features with long path lengths from iTrees; whereas dissimilar instances have many relevance features with short path lengths.

A region defined by a long path length in an iTree has many same splitting conditions, where each condition is defined by an internal node along the path from the root to the external node. Thus, intuitively, instances falling into each of these regions (defined by long path lengths) are likely to be more similar than those instances falling into other regions. This explains why we use Eq. (1) to assign high weights to iTrees where the query is estimated to have long path lengths—a big contribution to the relevance score through Eq. (3) if the test instance also achieves long path lengths on these iTrees. On the other hand, if an instance is estimated to have a short path length on an iTree, then it is most likely to be different from the instances falling into the regions defined by long-path-length external nodes. Thus, Eq. (1) assigns negative weights to the iTrees in which the query has short path lengths—via Eq. (3) to penalize the test instances with long path lengths in these iTrees. In addition, if the query is estimated to have a path length around $c(\psi)$, then we simply assign a small or zero weight because instances having similar path lengths are likely to be in different regions.

In the following paragraphs, we first provide the topologically distinct iTree structures in the setting we have used in our experiment. Then, we show that the majority of iTrees produced from a database have distinct long and short path lengths that allow our scheme to identify similar instances from dissimilar ones through ranking.

The parameters we have used in the experiment are: the sub-sample size $\psi = 8$ and the height limit $h = \lceil \log_2 \psi \rceil = 3$. This produces a total of 17 topologically distinct tree structures as shown in Fig. 1. To obtain the path length of an instance $\mathbf{x}$ from an iTree, $\mathbf{x}$ traverses from the root of the iTree to an external node; and the path length is computed as the number of edges traversed plus the estimated average path length of an unbuilt subtree from a sample of *Size* instances which is $c(Size)$, where *Size* is the number of sub-sample instances at the external node and $c(\cdot)$ is defined in Eq. (2). Note that out of the 17 structures depicted in Fig. 1, structures (a)–(g) all have the minimum path length equal to 1; and structures (h)–(p) have the minimum path length equal to 2. These structures have the maximum path lengths vary from $3+c(5)$, $3+c(4)$, $3+c(3)$ to $3+c(2)$. Only structure (q) is a balanced tree which gives the same path length for all instances.

An iTree is only useful if it is imbalanced and provides long and short path lengths that differentiate similar and dissimilar instances. It is also preferred to have the maximum path length in only one external node that uniquely identifies the neighborhood region. A total of 10 structures, i.e., (a)–(f) and (h)–(k), satisfy this essential property,[1] where the maximum path lengths are

[1] Structure (d) is an exception but it still stipulates the neighborhood region by at least two splitting conditions. We include (d) here to facilitate the following analysis.

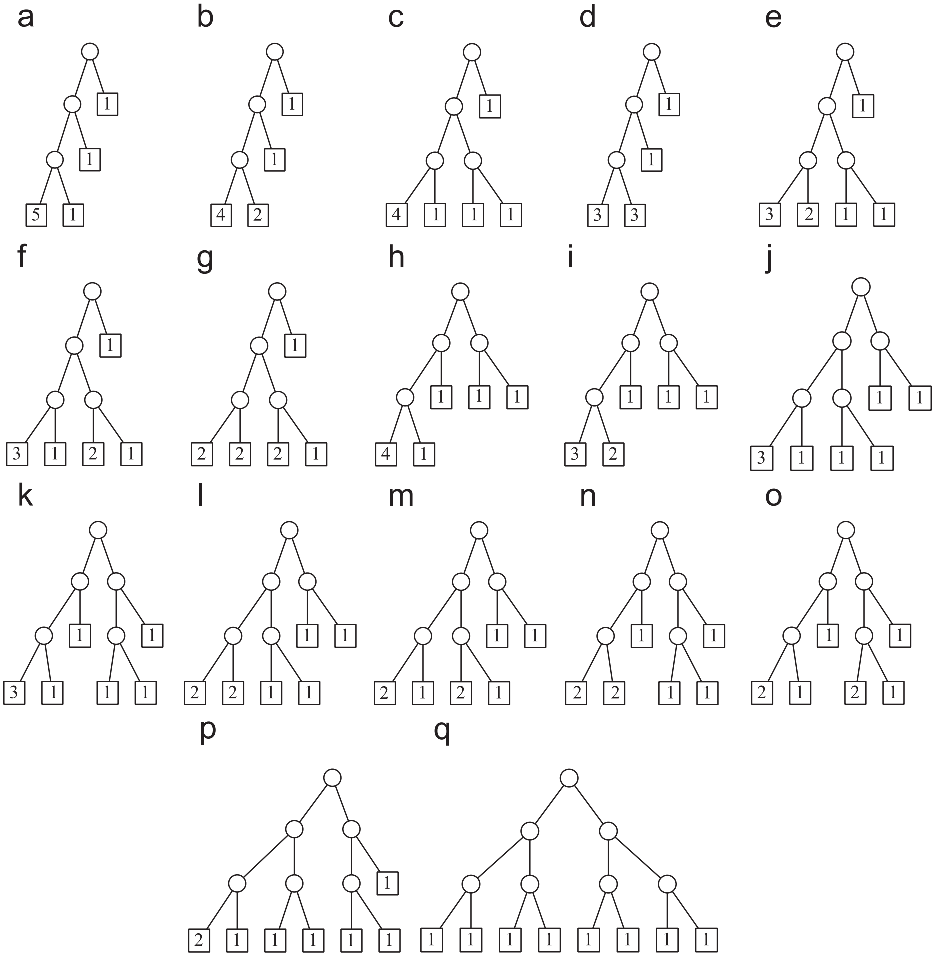

Fig. 1. The 17 unique iTree structures with $\psi = 8$ and $h = 3$. Circles (○) denote internal nodes, and squares (□) are external nodes. The figure in an external node indicates the number of sub-sample instances split in the node, i.e., the "*Size*" of the node.

$3+c(5)$, $3+c(4)$ and $3+c(3)$. A total of six structures, i.e., (g) and (l)–(p), are also good by providing short path lengths. An iTree like structure (q) which gives the same path length for all instances is useless for our purpose.

We employ δ, which is the difference between the maximum path length and the minimum path length of an iTree, to indicate how imbalance the iTree is. Out of the 17 topologically distinct tree structures, we have only eight δ values: 0, $1+c(2)$, $2+c(2)$, $1+c(3)$, $1+c(4)$, $2+c(3)$, $2+c(4)$, and $2+c(5)$, which range from balanced tree (q) to highly imbalanced tree (a).

Using the COREL image database [24], we generate 1000 iTrees and then tally the number of trees for each δ value. Fig. 2(a) shows the result: more than 75% of the iTrees have $\delta \geq 1+c(3)$ which represents the 10 imbalanced iTree structures (a)–(f) and (h)–(k). The near-balanced trees (having $0 < \delta \leq 2+c(2)$) constitute about 23% of the iTrees which represents the six structures (g) and (l)–(p). The balanced iTrees constitute less than 1%. The result shows that the majority of the generated iTrees are useful for identifying similar instances as well as dissimilar instances.

To further enhance the understanding, we provide statistics of the path lengths in the following case study. We select a rose image (Fig. 2(b)) from the COREL database as a query. Another rose image (Fig. 2(c)) is considered as relevant, and a beach image (Fig. 2(d)) is treated as irrelevant. We estimate the path lengths of the three images on the above-generated 1000 iTrees. Considering the 17 distinct iTree structures, we have seven possible path length values ranging from the longest to the shortest: $3+c(5)$, $3+c(4)$, $3+c(3)$, $3+c(2)$, 3, 2, and 1. We then divide the 1000 iTrees into seven categories based on the query's path lengths. In each category, we calculate the proportion of iTrees that produce different path lengths for the relevant image and the irrelevant image, and the results are provided in Table 1. It shows that: on highly weighted iTrees (in which the query has long path lengths, shown in top rows in Table 1(a) and (b)), the relevant image has significantly more long path lengths than the irrelevant image; on negatively weighted iTrees (in which the query has short path lengths, shown in bottom rows in Table 1(a) and (b)), the relevant image has noticeably less long path lengths. This explains why the relevant image scores larger than the irrelevant one through Eq. (3) in our ranking scheme. In this case, the scores for relevant and irrelevant images are 1.14 and 0.89, respectively.

Also notice that the similarity between the relevant image and the query is implied by the high proportion of iTrees when the path length is matched between the two images (see the numbers in the diagonal of Table 1(a)). The corresponding proportions of iTrees are significantly less between the irrelevant image and the query image, shown in Table 1(b).

4.4. On-line retrieval in relevance feedback

If feedbacks are available, we use them to refine the retrieval result by modifying the feature weights. Here the query is denoted by $\mathcal{Q} = \mathcal{P} \cup \mathcal{N}$, where $\mathcal{P}$ is the set of positive feedbacks and the initial query; and $\mathcal{N}$ is the set of negative feedbacks. Begin with the initial query $\mathbf{q}$, they are initialized as follows: $\mathcal{P} = \{\mathbf{q}\}$ and

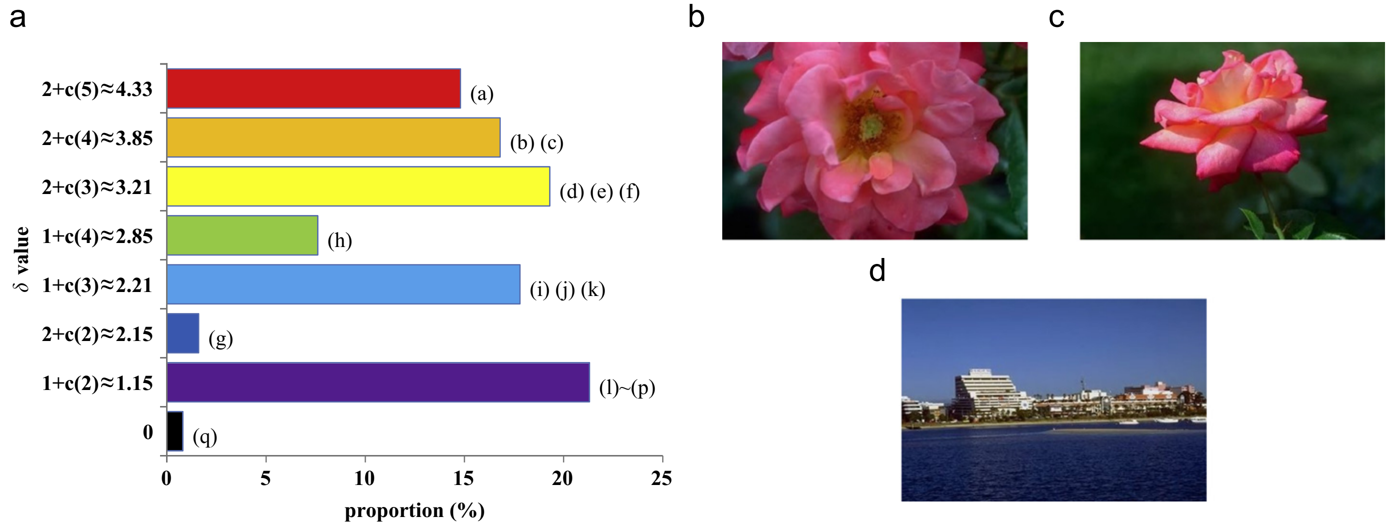

Fig. 2. Statistics of iTrees and the sample images used in our case study. (a) The proportions of 1000 iTrees with different δ values. (b) Query image. (c) Relevant image. (d) Irrelevant image.

Table 1
The proportion (%) of iTrees that produce different path lengths for the relevant image (Fig. 2(c)) and the irrelevant image (Fig. 2(d)) out of the number of iTrees that estimate a specified path length for the query (Fig. 2(b)). For this query image, the numbers of iTrees having path lengths $3+c(5)$, $3+c(4)$, $3+c(3)$, $3+c(2)$, 3, 2, 1 are 79, 99, 121, 162, 268, 189, 82, respectively.

Query's path length	Proportion of iTrees with path length						
	$3+c(5)$	$3+c(4)$	$3+c(3)$	$3+c(2)$	3	2	1
(a) Relevant image							
$3+c(5)$	**83.5**	N/A	N/A	N/A	3.8	7.6	5.1
$3+c(4)$	N/A	**77.8**	N/A	3.0	8.1	4.0	7.1
$3+c(3)$	N/A	N/A	**73.6**	8.3	7.4	7.4	3.3
$3+c(2)$	N/A	1.2	5.6	**74.1**	9.3	5.6	4.3
3	1.9	1.5	7.1	6.3	**75.7**	4.5	3.0
2	2.1	3.2	4.2	4.8	7.4	**77.2**	1.1
1	4.9	4.9	3.7	2.4	3.7	2.4	**78.0**
(b) Irrelevant image							
$3+c(5)$	29.1	N/A	N/A	N/A	19.0	15.2	**36.7**
$3+c(4)$	N/A	**33.3**	N/A	7.1	21.2	23.2	15.2
$3+c(3)$	N/A	N/A	**29.8**	8.3	20.7	26.4	14.9
$3+c(2)$	N/A	3.1	6.2	24.7	**32.7**	21.0	12.3
3	1.9	7.8	12.7	17.9	**29.9**	19.0	10.8
2	9.0	11.1	14.8	15.3	**23.8**	18.5	7.4
1	17.1	15.9	11.0	7.3	**29.3**	9.8	9.8

$\mathcal{N}=\emptyset$. Then, $\mathcal{P}$ and $\mathcal{N}$ are enriched with the instances labeled by the user in the relevance feedback process.

If only positive feedbacks are provided, ReFeat puts them in $\mathcal{P}$ and calculates the feature weights in the same way as that for the initial query. Formally, ReFeat defines the weight of feature i due to a positive feedback $\mathbf{z}^+ \in \mathcal{P}$ as:

$$w_i^+(\mathbf{z}^+) = \frac{\ell_i(\mathbf{z}^+)}{c(\psi)} - 1. \tag{4}$$

Then the resultant weight for feature i due to $\mathcal{P}$ is obtained by averaging the weights produced by all the positive instances in $\mathcal{P}$:

$$w_i^+(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{k=1}^{|\mathcal{P}|} w_i^+(\mathbf{z}_k^+). \tag{5}$$

Here $|\cdot|$ denotes the size of a set. Now by replacing $w_i(\mathbf{q})$ with $w_i^+(\mathcal{P})$ in Eq. (3), a new ranking score can be produced for each instance and a refined retrieval result is returned to the user.

When negative feedbacks are also provided in relevance feedback, ReFeat puts them in $\mathcal{N}$ and defines the weight in an opposite way as for the initial query: a high weight is assigned to a feature which signifies that a negative feedback is irrelevant to the profile modeled by the feature; otherwise, a low weight is assigned. To implement this, ReFeat calculates the weight for feature i due to a negative feedback $\mathbf{z}^- \in \mathcal{N}$ as:

$$w_i^-(\mathbf{z}^-) = 1 - \frac{\ell_i(\mathbf{z}^-)}{c(\psi)}. \tag{6}$$

The resultant weight for feature i due to $\mathcal{N}$ is generated by averaging over all negative instances in $\mathcal{N}$:

$$w_i^-(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{s=1}^{|\mathcal{N}|} w_i^-(\mathbf{z}_s^-). \tag{7}$$

Now the final weight for feature i can be obtained by aggregating $w_i^+(\mathcal{P})$ and $w_i^-(\mathcal{N})$. The aggregation can be realized in different ways. Here we use a simple summing method: $w_i(\mathcal{Q}) = w_i^+(\mathcal{P}) + \gamma w_i^-(\mathcal{N})$, where $\gamma \in (0,1]$ is a trade-off parameter accounting for the relative weights of the contributions between positive and negative instances. It is reasonable that positive instances make more contribution to the final ranking than negative ones. Since the farther an instance lies from positive instances, the less likely that it is a relevant one. However, we can not draw an opposite conclusion for negative instances: if an instance lies far from negative instances, it is not necessarily relevant, since it may be far from positive instances too. Similar strategies were employed in previous works (e.g., [9,11]). The empirical study presented in Section 5.2.6 also shows the efficacy of this strategy.

Finally, ReFeat estimates the ranking score for every instance in the database using Eq. (3) (by replacing $w_i(\mathbf{q})$ with $w_i(\mathcal{Q})$), and returns the instances by ranking them in a descending order according to their scores.

4.5. Complexity

We now analyze the time complexity of ReFeat. In the off-line modeling stage, building the iForest model takes $O(t\psi \log \psi)$ and the mapping from $\mathcal{D}$ to $\mathcal{D}'$ costs $O(|\mathcal{D}|t \log \psi)$ [23]. Thus, the total time complexity is $O((|\mathcal{D}|+\psi)t \log \psi)$. In the on-line retrieval stage, the relevance feature mapping for the query costs

Table 2
Time complexities of ReFeat, Euclidean, InstRank [10] and Qsim [11] for on-line retrieval. Here d is the original dimension of the multimedia database $\mathcal{D}$. InstRank and Qsim are methods dealing with relevance feedback only.

Method	With one query	In relevance feedback
ReFeat	$O((\|\mathcal{D}\|+\log\psi)\times t)$	$O((\|\mathcal{D}\|+\|\mathcal{Q}\|)\times t)$
Euclidean	$O(\|\mathcal{D}\|\times d)$	N/A
InstRank	N/A	$O(\|\mathcal{D}\|\times\|\mathcal{Q}\|\times d)$
Qsim	N/A	$O(\|\mathcal{D}\|\times\|\mathcal{Q}\|\times(d+\|\mathcal{P}\|))$

$O(t\log\psi)$, calculating weights takes $O(|\mathcal{Q}|t)$, and producing ranking scores for all instances in the database costs $O(|\mathcal{D}|t)$. Thus, for a query session, ReFeat has a time complexity of $O((|\mathcal{D}|+|\mathcal{Q}|+\log\psi)t)$. It is worth noting that $|\mathcal{Q}|$ is much smaller than $|\mathcal{D}|$, and both t and ψ are fixed at the beginning of the off-line modeling stage which do not change in on-line retrieval. Thus, ReFeat has a linear time complexity with respect to $|\mathcal{D}|$ in both the off-line modeling stage and the on-line retrieval stage, which makes it possible to scale up to large multimedia databases. Table 2 lists the time complexities of ReFeat as well as three other methods for on-line retrieval. It shows that ReFeat has a relatively low time complexity in on-line retrieval although it needs an additional modeling stage. Note that we also compare BALAS and MRBIR in our experiments. Although it is difficult to analyze their complexities, the experimental results show that BALAS and MRBIR usually spend much longer time than ReFeat.

The space requirement of our off-line model is also linear with respect to $|\mathcal{D}|$, since the database $\mathcal{D}'$ costs $O(|\mathcal{D}|t)$ and iForest requires $O((2\psi-1)tb)$ memory space only [23], where b is the memory size taken by a tree node.

5. Experiments

The performance of ReFeat is evaluated with content-based image and music retrieval tasks on COREL image database (which is used in [24]) and GTZAN music database [25], respectively. The image database consists of 10 000 COREL images that are collected from 100 categories such as car, forest, sunset, tiger, etc.; each category contains 100 images. As in [24], each image is represented by a 67-dimensional feature vector which consists of 32 color features generated by HSV histogram, 24 texture features derived from Gabor wavelet transformation and 11 shape features including invariant moments, center coordinates, area and principal axis orientation. The music database contains 1000 songs which are uniformly distributed over 10 genres including classical, country, disco, hiphop, jazz, rock, blues, reggae, pop, and metal. Each song is a 30-second excerpt which is stored as a 22 050 Hz, 16-bit, mono-audio file. Following the feature extraction steps in [26], we split each song into 3-second segments, where a MFCC [25] feature vector is extracted from each segment and the top 20 MFCC coefficients are kept to represent the segment. The mean and the lower-triangular covariance matrix of MFCC features are calculated and concatenated into a 230-dimensional feature vector to represent the song. Note that there is no feature selection although it may be beneficial. The same features are used by all the compared methods because we are only interested in the relative instead of absolute performance of the methods.

Our experiments study the retrieval performance of ReFeat both with one query and in relevance feedback. The initial queries are chosen as follows: for the image database, we randomly select five images from each category to obtain 500 initial queries; and for the music database, we use every song in the whole database and there are a total of 1000 initial queries. For a query, the images/songs within the same category/genre are regarded as relevant and the rest are irrelevant. We continue to perform five rounds of relevance feedback for each query. In each round, we randomly select two relevant and two irrelevant instances as positive and negative feedbacks, respectively. Note that an instance will not be considered for selection if it has been chosen as a feedback in previous rounds. To simulate different users' behavior, this relevance feedback process is repeated five times, each with a different random series of feedbacks. Finally, we report the average result over multiple runs for the initial query and the subsequent rounds of feedback.

PR-curve is a commonly used performance measure in information retrieval. It depicts the relationship between precision and recall of a retrieval system. In the experiments, we employ PR-curve to evaluate the retrieval performance with one query. However, in relevance feedback, a single PR-curve is not enough to reveal the performance changes with the increasing number of feedbacks. Thus, we use Mean Average Precision (MAP) and Precision at N (P@N) [4]. MAP is the average of precisions computed at the point of each of the relevant instances in the ranked sequence. P@N records the fraction of relevant instances in the top-N ranked instances, and we empirically set $N=50$ in the following experiments. The higher the MAP and P@N values, the better the performance. Notice that previous works (e.g., [10,11]) have included feedback instances in the evaluation of retrieval performance. However, this calculation inflates the performance since the feedbacks are labeled instances that should not be displayed to the user. Thus, we have excluded feedbacks in our performance evaluation.

The efficacy and efficiency of ReFeat are validated in the next subsection, followed by empirical studies showing the effectiveness of the relevance feature mapping, the utility of our ranking scheme, the influence of increasing database dimension, and the effect of different parameter settings in ReFeat. All the experiments are conducted on a Pentium 4 machine with a 1.86 GHz CPU and 2 GB memory.

5.1. *Comparison with existing methods*

In this subsection, we first compare ReFeat with the Euclidean distance based method and a manifold ranking method MRBIR [9] when no relevance feedback is performed. Then with relevance feedbacks, Qsim [11], InstRank [10], MRBIR [9] and BALAS [2] are employed for benchmarking. Here Qsim and InstRank are methods for improving ranking calculation, and BALAS is a Bayesian learning method. Because Qsim and InstRank are proposed to be used only in relevance feedback for improving similarity calculation, we employ Euclidean distance to measure the relevance so that they can deal with query without feedbacks. BALAS also does not mean to work with single query and there is no comparison of BALAS for retrieval with one query. Note that we also include the chance performance of random method (called Random) as a baseline method.

There are three parameters in ReFeat: number of relevance features t, sub-sample size ψ and trade-off parameter γ. ReFeat is not very sensitive to γ when $\gamma\in[0.1,0.4]$, and we set $\gamma=0.25$ for both the image and music databases. The values of t and ψ are problem-dependent. We set $t=1000$, $\psi=8$ for the image database, and $t=1000$, $\psi=4$ for the music database. The effect of the three parameters on the performance of ReFeat is studied in Section 5.2. For MRBIR, we keep the default parameter settings as in [9]: 200 nearest neighbors are used for constructing the weighted graphs; the contribution of negative ranking scores is weighted by 0.25; the trade-off parameter α is set to be 0.99 in the manifold ranking algorithm, which iterates 50 rounds to

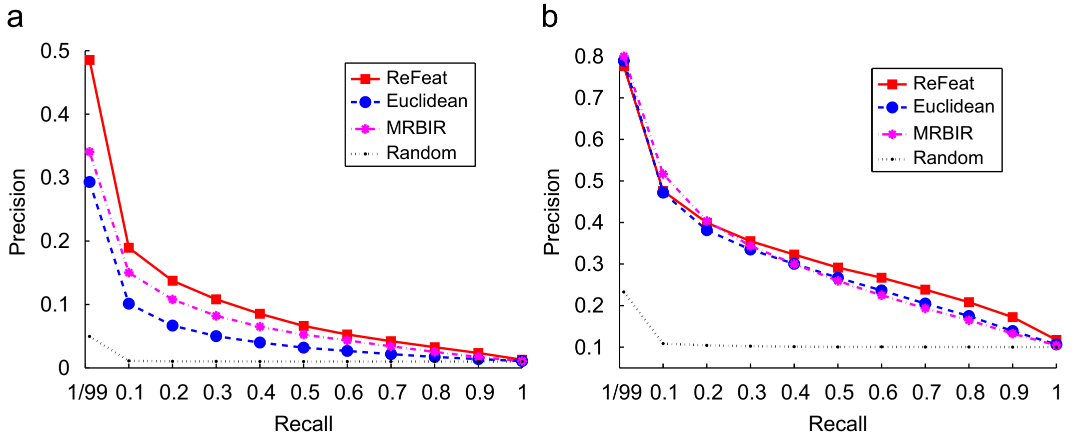

Fig. 3. PR-curves of ReFeat, Euclidean, MRBIR and Random for retrieval with one query. (a) COREL image database. (b) GTZAN music database.

obtain the final results. The only difference lies in the setting of δ_l in computing Laplacian kernels: while [9] empirically sets $\delta_l = 0.05$, we select the best δ_l from $\{0.0125, 0.025, 0.05, 0.1, 0.5, 1\}$ and use 0.05 for the image database and 0.025 for the music database. For BALAS, we generate five random instances to represent each negative feedback (in addition to the feedback instances selected from the database) to enable the estimation of the probability density function. The threshold for determining high trustworthy dimensions is kept to be 0.7 as in [2]. Qsim and InstRank do not have parameters that need to be set.

5.1.1. *Retrieval with one query*

The PR-curves of ReFeat, Euclidean, MRBIR and Random for retrieval with one query are presented in Fig. 3. It shows that on the image database, ReFeat outperforms the other three compared methods, and MRBIR is better than Euclidean; and on the music database, ReFeat is better than Euclidean, MRBIR and Random on most recall values, except that MRBIR achieves the best precision when the recall value ≤ 0.2.

We also provide a detailed comparison in Table 3 to gain further insight into the advantages of ReFeat. For each initial query, we calculate the MAP and P@50 values using every compared method, and present the average results in Table 3. A paired *t*-test at 5% significance level is performed for the MAP (and P@50) series over all queries, and we record the probability of rejecting the hypothesis that ReFeat is significantly better than the compared method. The average results in Table 3 reveal that ReFeat performs better than Euclidean and MRBIR, and the *t*-test results show that the difference is statistically significant. The only exception is that ReFeat achieves no significant result against MRBIR on the music database. These observations reveal the superior performance of ReFeat for retrieval with one query.

5.1.2. *Retrieval in relevance feedback*

Fig. 4 shows the MAP and P@50 results for retrieval in relevance feedback. Note that round 0 presents the retrieval performance with one query only, and Euclidean is used as the base method for Qsim and InstRank.

It is found in Fig. 4 that as the number of feedback rounds increases, the retrieval performance of most methods tends to improve. However, BALAS performs poor on the music database, and we suspect that this might be caused by the violation of feature independent assumption on the music database. Nevertheless, Fig. 4 clearly reflects that ReFeat achieves the best MAP and P@50 no matter how many feedbacks are provided. Since ReFeat has superior performance with both one query and relevance feedbacks, we can conclude that ReFeat is highly effective for CBMIR.

Table 3
A detailed comparison (average MAP ($\times 10^{-2}$), average P@50 ($\times 10^{-2}$) and *t*-test) of ReFeat against Euclidean and MRBIR for retrieval with one query.

Method	COREL image database		GTZAN music database	
	MAP	P@50	MAP	P@50
ReFeat	**9.11**	**15.64**	**31.06**	37.59
Euclidean	4.76	8.97	28.94	36.18
MRBIR	7.03	11.99	29.27	**37.74**
t-test results of ReFeat against:				
Euclidean	4.6×10^{-28}	1.4×10^{-29}	2.7×10^{-14}	2.0×10^{-4}
MRBIR	2.3×10^{-7}	2.5×10^{-9}	1.9×10^{-10}	0.7199

5.1.3. *Processing time*

The average on-line processing time of all compared methods is tabulated in Table 4 where the shortest time at each round is boldfaced. Note that the processing time for retrieval with one query is reported in round 0, where the time costs of Qsim and InstRank are filled by that of Euclidean.

Table 4 shows that ReFeat has the best efficiency except that it spends a bit more time than Euclidean for retrieval with one query on the image database. This implies that Euclidean prefers low-dimensional databases and ReFeat is more efficient on high-dimensional databases. We have provided a detailed analysis in Section 5.2.3 on how the database dimension influences the retrieving time of the compared methods. Note that ReFeat achieves the shortest and near constant processing time regardless of the feedback round. The time is independent of the number of feedbacks because the time complexity of ReFeat for retrieval in relevance feedback, i.e., $O((|\mathcal{D}|+|\mathcal{Q}|)\times t)$ (as shown in Table 2), is dominated by $O(|\mathcal{D}|\times t)$ as $|\mathcal{Q}| \ll |\mathcal{D}|$. InstRank also has a near constant time cost because the distances calculated in previous feedback rounds are saved for the following rounds. MRBIR has to iteratively update the ranking result with expensive large matrix operations, resulting in the highest on-line retrieval time.

Although ReFeat has an off-line modeling stage, it costs only 2.87 s for the image database containing 10 000 images and 0.33 s for the music database containing 1000 songs, respectively. We believe that it pays to employ such an off-line modeling stage because of the good retrieval performance and quick processing time achieved by ReFeat for on-line retrieval.

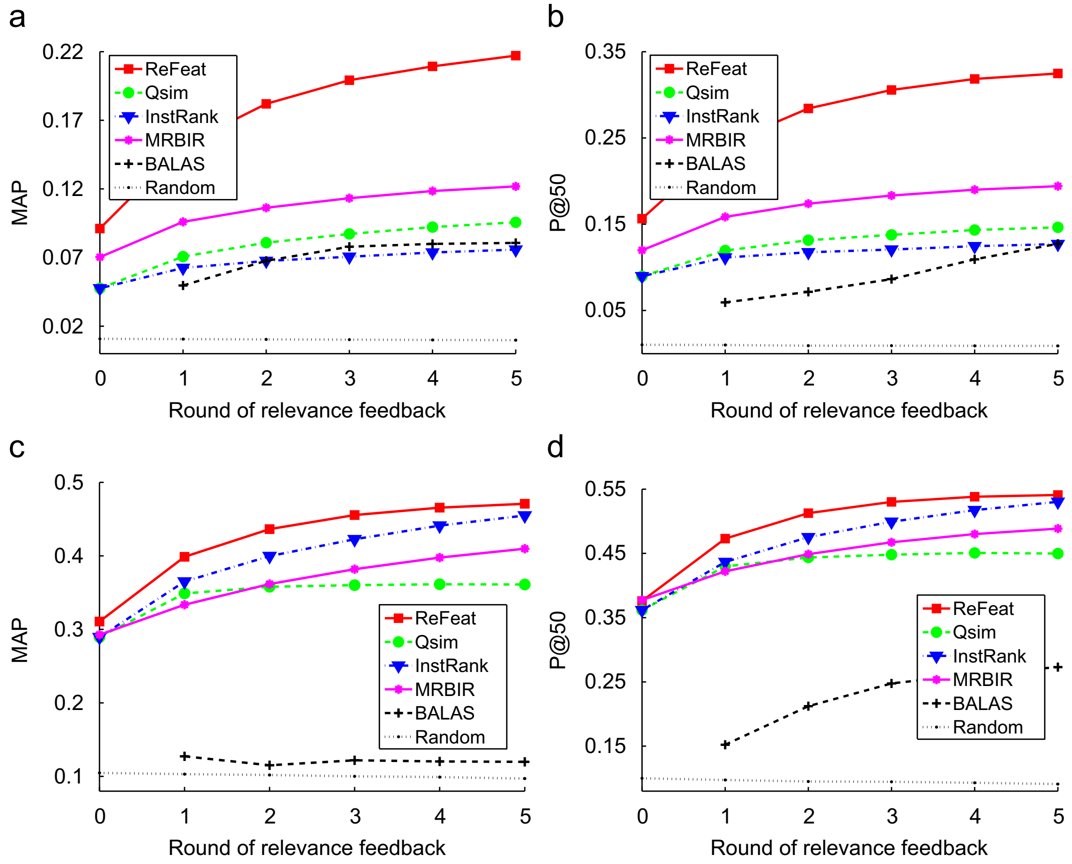

Fig. 4. Average MAP and P@50 values of `ReFeat`, `Qsim`, `InstRank`, `MRBIR`, `BALAS` and `Random` for retrieval in relevance feedback. (a) COREL image database: MAP. (b) COREL image database: P@50. (c) GTZAN music database: MAP. (d) GTZAN music database: P@50.

Table 4
Average on-line processing times (in millisecond) of `ReFeat` (`RF`), `Qsim` (`QS`), `InstRank` (`IR`), `MRBIR` (`MR`) and `BALAS` (`BA`).

Round	RF	QS	IR	MR	BA
(a) COREL image database					
0	27.2	**24.7**	**24.7**	612.9	N/A
1	**23.8**	71.3	32.6	1172.4	262.8
2	**24.0**	146.3	33.4	1172.3	317.5
3	**24.2**	261.9	34.2	1172.3	373.0
4	**24.4**	417.9	34.9	1172.2	437.9
5	**24.5**	615.8	35.5	1172.1	506.0
(b) GTZAN music database					
0	**3.6**	10.8	10.8	168.1	N/A
1	**3.1**	16.6	14.1	279.0	152.2
2	**3.3**	20.9	14.2	279.5	160.2
3	**3.4**	27.4	14.3	279.3	166.7
4	**3.6**	36.7	14.3	278.6	173.2
5	**3.7**	47.8	14.4	280.5	180.3

5.2. *Analysis*

This subsection analyzes some important issues in relation to `ReFeat`. We first empirically validate the effectiveness of the relevance feature mapping and our ranking scheme. Then we show the influence of increasing database dimension on the compared methods. At the end of this subsection, we study the effect of the three parameters in `ReFeat` and give some guidelines for selecting them. Note that the same conclusion can always be made for both MAP and P@50. Thus, we only provide the MAP results hereafter.

5.2.1. *Relevance feature mapping*

Recall that `ReFeat` is a two-stage process, where the first maps database instances to a relevance feature space, and the second ranks the instances in the new space. We conduct experiments to show the effectiveness of our relevance feature mapping in this subsection, and the efficacy of the proposed ranking scheme is validated in the next subsection.

Previous experiments have already shown that `ReFeat` outperforms existing methods which are conducted in the original feature space. Here, we hypothesize that the performance of existing methods can be improved using our relevance features. Thus, we perform three distance based methods, i.e., `Qsim`, `InstRank` and `MRBIR`, in our relevance feature space. The new methods are named `Qsim-RF`, `InstRank-RF` and `MRBIR-RF`, respectively. Table 5 presents the MAP results which are grouped in pairs for ease of comparison. Exactly the same relevance feature mapping is employed for all methods that use it. Note that round 0 gives the results with one query, and the `Euclidean` method performed in the original feature space is used as the base method for `Qsim` and `InstRank`. Similarly, Euclidean distance measured in the relevance feature space is employed by `Qsim-RF` and `InstRank-RF` at round 0.

As shown in Table 5, with the help of the relevance feature mapping, `Qsim-RF`, `InstRank-RF` and `MRBIR-RF` significantly outperform their original versions, i.e., `Qsim`, `InstRank` and `MRBIR`, respectively. There are two exceptions on the music database: the first is `InstRank-RF` which performs worse than `InstRank`, and the second is for retrieval with one query, `Euclidean` performs slightly better in the original space. Nevertheless, these observations show that our relevance feature space is more suitable for retrieval than the original space, and thus, we can conclude that the power of ReFeat is largely derived from the relevance feature mapping.

We also report the on-line processing time in Table 6. The time costs of Qsim-RF and InstRank-RF are expected to be longer than each of the original versions because the dimensionality of the relevance feature space is significantly higher than that of the original space. It is interesting to note that MRBIR-RF spends less time than MRBIR in most cases. This indicates that it is easier to find the underlying manifold in our relevance feature space, as compared to that in the original space.

Despite these improvements, ReFeat is still significantly better than the other three methods applied in the relevance feature space (except that MRBIR-RF achieves the best performance for retrieval with one query on the image database). The processing time reported in Table 6 also shows that ReFeat has the best efficiency among these methods. These results validate the efficacy and efficiency of our proposed ranking scheme. We will provide a more detailed analysis on our ranking scheme in the next subsection.

Table 5
Average MAP values ($\times 10^{-2}$) of ReFeat (RF), Qsim-RF (QS-RF), Qsim (QS), InstRank-RF (IR-RF), InstRank (IR), MRBIR-RF (MR-RF) and MRBIR (MR). The figures boldfaced are the best performance on each feedback round while the underlined indicate the better performance in each grouped pair.

Round	RF	QS-RF	QS	IR-RF	IR	MR-RF	MR
(a) COREL image database							
0	9.11	8.87	4.76	8.87	4.76	**10.88**	7.03
1	**15.17**	14.83	7.07	10.56	6.24	14.52	9.60
2	**18.20**	17.51	8.08	11.81	6.76	16.01	10.63
3	**19.92**	19.17	8.72	12.85	7.06	17.05	11.32
4	**20.93**	20.17	9.22	13.49	7.37	17.68	11.84
5	**21.71**	20.98	9.57	14.07	7.58	18.11	12.18
(b) GTZAN music database							
0	**31.07**	28.73	28.94	28.73	28.94	29.54	29.27
1	**39.87**	35.14	34.89	32.70	36.50	34.15	33.36
2	**43.64**	37.06	35.80	36.06	39.97	37.01	36.17
3	**45.56**	38.17	36.02	38.64	42.26	39.06	38.19
4	**46.56**	38.78	36.14	40.52	44.11	40.58	39.76
5	**47.09**	39.12	36.10	41.92	45.49	41.76	40.97

Table 6
Average on-line processing time (in millisecond) of ReFeat (RF), Qsim-RF (QS-RF), Qsim (QS), InstRank-RF (IR-RF), InstRank (IR), MRBIR-RF (MR-RF) and MRBIR (MR). The figures boldfaced are the smallest time on each feedback round while the underlined indicate the smaller time in each grouped pair.

Round	RF	QS-RF	QS	IR-RF	IR	MR-RF	MR
(a) COREL image database							
0	27.2	345.5	**24.7**	345.5	**24.7**	955.5	612.9
1	**23.8**	461.9	71.3	421.6	32.6	1117.4	1172.4
2	**24.0**	540.1	146.3	422.4	33.4	1117.5	1172.3
3	**24.2**	660.7	261.9	423.1	34.2	1117.5	1172.3
4	**24.4**	823.1	417.9	423.8	34.9	1117.5	1172.2
5	**24.5**	1030.2	615.8	424.5	35.5	1117.4	1172.1
(b) GTZAN music database							
0	**3.6**	34.2	10.8	34.2	10.8	96.6	168.1
1	**3.1**	45.6	16.6	42.2	14.1	114.8	279.0
2	**3.3**	51.0	20.9	42.3	14.2	114.9	279.5
3	**3.4**	59.7	27.4	42.4	14.3	114.9	279.3
4	**3.6**	71.2	36.7	42.5	14.3	114.8	278.6
5	**3.7**	86.4	47.8	42.5	14.4	114.8	280.5

5.2.2. *The ranking scheme*

This subsection analyzes our ranking scheme. ReFeat incorporates the query information into the feature weights. Here, we employ the same weights in the existing methods to improve their performance. Based on InstRank-RF, we design a new method called InstRank-WRF which uses weighted Euclidean distance instead of Euclidean distance in InstRank-RF. The weights for the relevance features are calculated in exactly the same way as that in ReFeat. InstRank-WRF is compared with ReFeat and InstRank-RF in Fig. 5 and Table 7. It is shown that InstRank-WRF outperforms InstRank-RF in most cases except for retrieval with one query on the image database. These observations show that the feature weights are not only useful in our ranking scheme, but also in existing distance-based ranking schemes. We also provide the retrieval performance of InstRank in Fig. 5. Recall that InstRank performs better than InstRank-RF on the music database. However, with the feature weights, InstRank-WRF is now better than InstRank.

Overall, Fig. 5 and Table 7 reveal that ReFeat is superior to InstRank-WRF in terms of both retrieval performance and

Table 7
Average on-line processing time (in millisecond) of ReFeat (RF), InstRank-WRF (IR-WRF) and InstRank-RF (IR-RF).

Round	COREL image database			GTZAN music database		
	RF	IR-WRF	IR-RF	RF	IR-WRF	IR-RF
0	**27.2**	731.6	345.5	**3.6**	98.5	34.2
1	**23.8**	1773.0	421.6	**3.1**	232.4	42.2
2	**24.0**	1881.2	422.4	**3.3**	237.9	42.3
3	**24.2**	1899.9	423.1	**3.4**	241.1	42.4
4	**24.4**	1920.9	423.8	**3.6**	244.1	42.5
5	**24.5**	1940.8	424.5	**3.7**	247.5	42.5

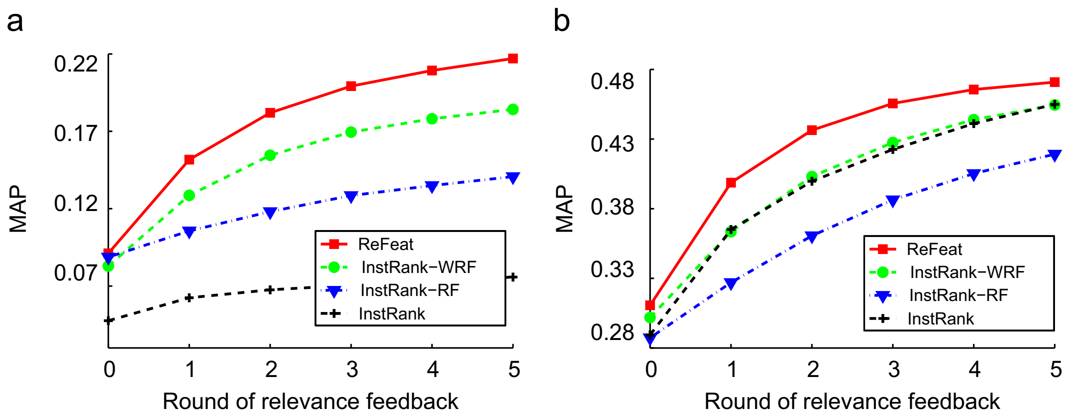

Fig. 5. Average MAP values of ReFeat, InstRank-WRF, InstRank-RF and InstRank. (a) COREL image database. (b) GTZAN music database.

processing time. This indicates that there is no need to calculate the costly distance in the relevance feature space; instead, a good ranking can be efficiently produced by simply averaging the weighted relevance feature values.

5.2.3. Increasing dimensionality

Recall that every image in the COREL image database is represented by a 67-dimensional feature vector containing shape, texture and color features. Here we denote the database as $\text{COREL}_{[67]}$, and construct three other databases: (i) $\text{COREL}_{[11]}$ employs 11 shape features only; (ii) $\text{COREL}_{[35]}$ uses 35 features, consisting of 11 shape and 24 texture features; and (iii) $\text{COREL}_{[200]}$ is a 200-dimensional database, created by adding 133 random features to $\text{COREL}_{[67]}$ (each random feature is generated from a uniform distribution). Similarly, we denote the original GTZAN music database as $\text{GTZAN}_{[230]}$, and construct three other databases: (i) $\text{GTZAN}_{[20]}$ uses the first 20 features of $\text{GTZAN}_{[230]}$; (ii) $\text{GTZAN}_{[100]}$ employs the first 100 feature of $\text{GTZAN}_{[230]}$; and (iii) $\text{GTZAN}_{[400]}$ is created by adding 170 random features to $\text{GTZAN}_{[230]}$. All the methods are evaluated on the eight databases, and the retrieval results with one query and in feedback round 5 are shown in Fig. 6 and Table 8.

Fig. 6 shows that ReFeat outperforms the other methods regardless of how many features are used to describe the database. The only exception is $\text{GTZAN}_{[20]}$, on which Euclidean is slightly better than ReFeat for retrieval with one query. These results validate the efficacy of ReFeat when dealing with different dimensional databases.

Note that on the databases $\text{COREL}_{[200]}$ and $\text{GTZAN}_{[400]}$ with randomly generated features, ReFeat outperforms the other methods with the lowest performance degradation, when compared with that on the original databases $\text{COREL}_{[67]}$ and $\text{GTZAN}_{[230]}$, respectively. For example, at feedback round 5 of image retrieval, the MAP value of ReFeat degrades by 50.4%, which is much better than 93.7% for Qsim, 93.7% for InstRank, 94.7% for MRBIR and 92.4% for BALAS; at feedback round 5 of music retrieval, ReFeat only degrades by 11.3%, as compared to 56.7% for Qsim, 66.1% for InstRank, 84.5% for MRBIR and 51.3% for BALAS. These results show that ReFeat has a good tolerance to randomly generated or irrelevant features.

Moreover, it is interesting to note that for our music retrieval problem, every method (except MRBIR) achieves the best MAP on $\text{GTZAN}_{[100]}$ out of the four databases including the original one, $\text{GTZAN}_{[230]}$. This observation indicates that the music retrieval

Table 8
Average on-line processing time (in millisecond) of the methods tested on the image and music databases with different dimensions. The method names are abbreviated as ReFeat (RF), Euclidean (EU), MRBIR (MR), Qsim (QS), InstRank (IR) and BALAS (BA).

Database	One query			Round 5				
	RF	EU	MR	RF	QS	IR	MR	BA
(a) COREL image database								
$\text{COREL}_{[11]}$	27.2	**4.3**	591.6	24.5	590.9	**10.6**	1172.1	347.1
$\text{COREL}_{[35]}$	27.2	**13.7**	599.6	24.5	602.8	**22.5**	1172.1	415.7
$\text{COREL}_{[67]}$	27.2	**24.7**	612.9	**24.5**	615.8	35.5	1172.1	506.0
$\text{COREL}_{[200]}$	**27.2**	69.5	645.2	**24.5**	670.0	89.7	1172.1	874.2
(b) GTZAN music database								
$\text{GTZAN}_{[20]}$	3.7	**0.4**	144.7	3.7	35.0	**1.6**	280.5	24.8
$\text{GTZAN}_{[100]}$	3.7	**3.6**	147.1	**3.7**	37.9	4.5	280.5	50.0
$\text{GTZAN}_{[230]}$	**3.6**	10.8	168.1	**3.7**	47.8	14.4	280.5	180.3
$\text{GTZAN}_{[400]}$	**3.7**	13.8	159.0	**3.7**	50.6	17.2	280.5	148.5

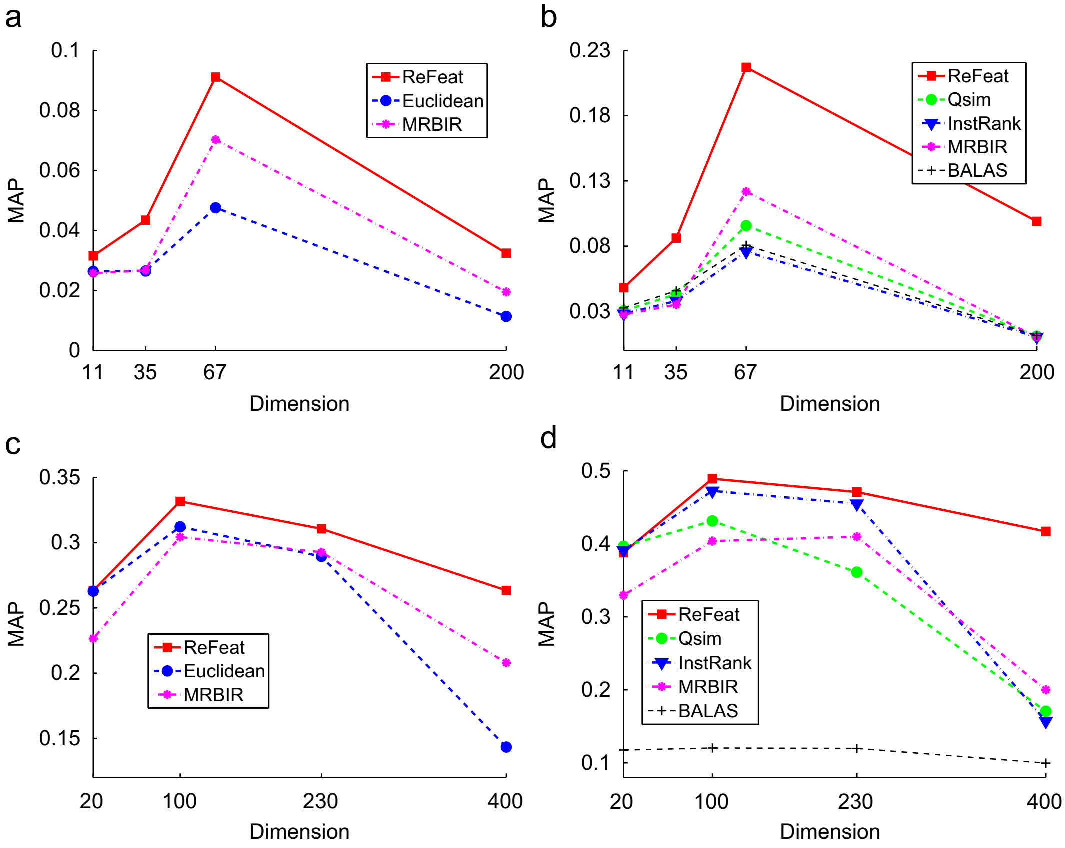

Fig. 6. Average MAP values of the compared methods evaluated on the image and music databases with different dimensions. (a) COREL image database: one query. (b) COREL image database: round 5. (c) GTZAN music database: one query. (d) GTZAN music database: round 5.

performance might be further improved using some proper feature selection scheme.

The processing time reported in Table 8 shows that `Euclidean` and `InstRank` spend the shortest time in low-dimensional cases, but their processing time increases linearly with respect to the database dimension. `Qsim`, `MRBIR` and `BALAS` spend much more time than `ReFeat`. `ReFeat` achieves constant time with respect to the database dimension, either dealing with one query or handling feedbacks. This enables our framework to scale up to high-dimensional databases without increasing the processing time.

5.2.4. *Using different numbers of relevance features*

We study the effect of the number of relevance features, i.e., t, in this subsection. Fig. 7 shows the MAP values of `ReFeat` as t varies from 10, 50, 100, 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, to 10 000. Here we set $\psi = 8$ and $\gamma = 0.25$ by default.

Fig. 7 shows that the retrieval performance of `ReFeat` rapidly increases with the increase of t when t is relatively small. Even with a sufficiently large t, the performance still appears to rise without overfitting. These observations show the possibility of improving the performance of `ReFeat` by adding more relevance features. However, when setting t, the trade-off between performance and processing time should be considered.

5.2.5. *Using different sub-sample sizes*

From the analysis provided in Section 4.3, we know that iForests built with different sub-sample sizes generate different sets of topologically distinct iTrees, thus producing different sets of distinct path lengths. We suspect that the "diversity" of the path lengths has a critical impact on the performance of `ReFeat`, because a system with diverse path lengths tends to provide a full range of relevancy to improve ranking results. Therefore, to gain an insight into the setting of sub-sample size ψ, we use Shannon index [27] to measure the diversities of iForests built with different ψ values. Shannon index is a statistic for measuring the biodiversity of an ecosystem. The index increases when the ecosystem has additional unique species or a greater species evenness. A bigger Shannon index indicates a larger diversity. In this subsection, each instance (e.g., an image or a song) is treated as an ecosystem. The instance may have different relevance feature values on different iTrees, and each possible feature value is considered as a species in the ecosystem. We count the numbers of the species and measure the instance diversity by Shannon index. The final diversity of the iTrees is estimated by averaging the Shannon indices over all instances. Formally, the diversity $D(\psi)$ of the iTrees built with sub-sample size ψ is calculated by:

$$D(\psi) = -\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{L}_\psi|}\left(\frac{n_j(\mathbf{x}_i)}{t}\ln\frac{n_j(\mathbf{x}_i)}{t}\right) - \frac{|\mathcal{L}_\psi|-1}{2t}, \qquad (8)$$

where $\mathcal{L}_\psi = \{\ell_1, \ell_2, \ldots, \ell_k\}$ is the set of all possible relevance feature values measured by the iTrees, $\mathbf{x}_i \in \mathcal{D}$ is an instance in the database, $n_j(\mathbf{x}_i)$ returns the number of iTrees in which $\mathbf{x}_i$ has feature value ℓ_j, t is the total number of iTrees, and $(|\mathcal{L}_\psi|-1)/2t$ is a correction factor.

We set $\psi = 2^2, 2^3, \ldots, 2^{12}$ for the image database, and $\psi = 2^2, 2^3, \ldots, 2^9$ for the music database. The resultant Shannon indices are plotted in Fig. 8, which shows that the diversity increases as ψ increases from 4 and reaches the peak at $\psi = 64$ on both the image and music databases. It is also interesting to note that the diversity decreases as ψ goes beyond 64, even though the number of possible feature values (i.e., possible species) increases. The MAP values of `ReFeat` are also shown in Fig. 8. Since the best performance of `ReFeat` is obtained with $\psi = 8$ for the image database and $\psi = 4$ for the music database, and there is no benefit to use a large ψ (i.e., $\psi > 64$), we suspect that the optimal setting for any task is somewhere between the smallest ψ $(=4)$ and the diversity peak. This can be used as an empirical guideline for setting the sub-sample size.

5.2.6. *Using different γ values*

We also study how the trade-off parameter γ affects the performance of `ReFeat` in relevance feedback. We test it by varying γ from 0 to 1 with step 0.1, and the resultant MAP values are shown in Fig. 9. It shows that `ReFeat` achieves relative good performance when $\gamma \in [0.1, 0.4]$ on both the image and music databases. These observations verify our statement in Section 4.4 that positive instances should contribute more than negative ones.

6. Discussion

This section discusses three related issues. We first provide some necessary characteristics of a ranking model to be applied in the `ReFeat` framework. Then, we detail the difference between our ranking scheme and that measured by distance and similarity. Finally, our ranking score calculation is compared with the one used in iForest for anomaly detection.

For a successful application in the proposed framework, the necessary characteristics of alternative ranking models are (i) each individual model provides a ranking of instances through some

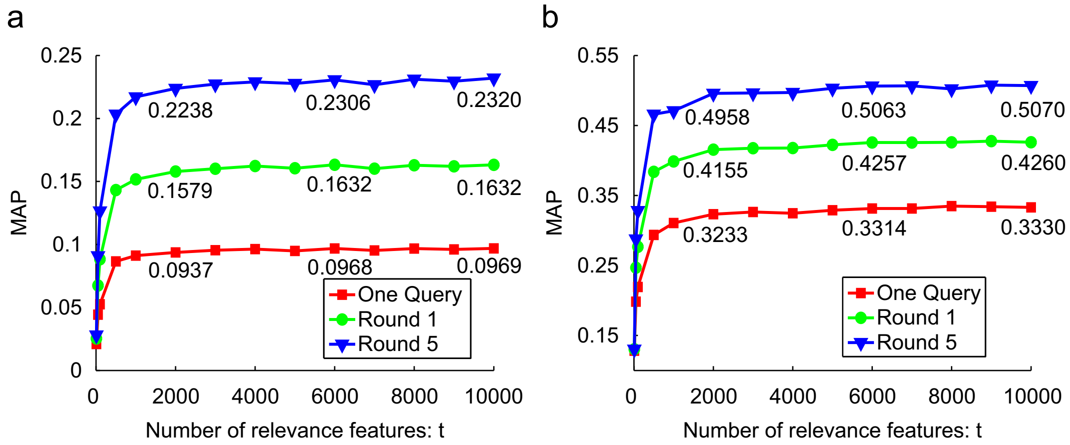

Fig. 7. Average MAP values of `ReFeat` with one query and in feedback rounds 1 and 5 using different number of relevance features. (a) COREL image database. (b) GTZAN music database.

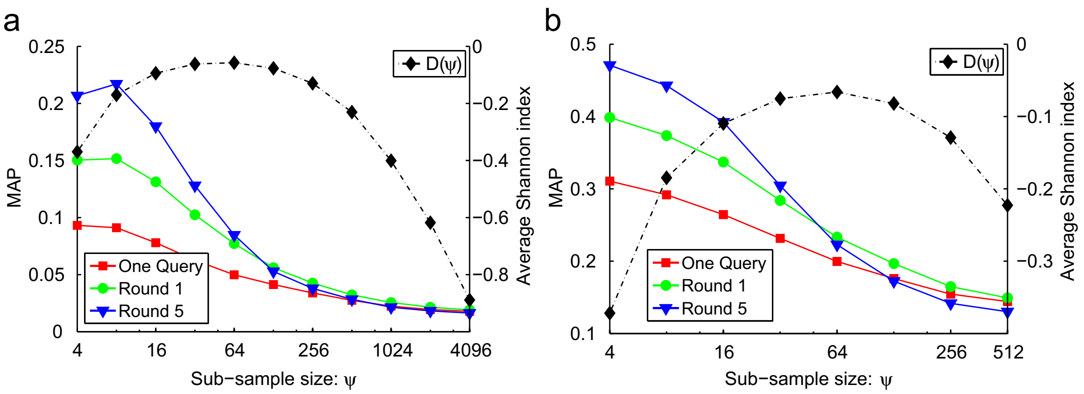

Fig. 8. Average MAP values of ReFeat with one query and in feedback rounds 1 and 5 using different sub-samples sizes. $D(\psi)$ is the average Shannon index calculated for the iTrees built with sub-sample size ψ. (a) COREL image database. (b) GTZAN music database.

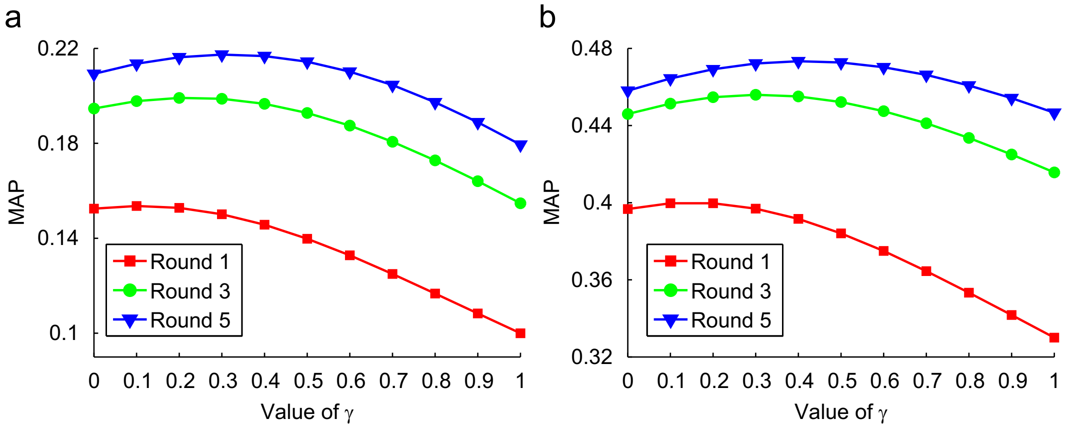

Fig. 9. Average MAP values of ReFeat in feedback rounds 1, 3 and 5 with different γ values. (a) COREL image database. (b) GTZAN music database.

profile underlying the database and (ii) each model is generated efficiently so that the multiple models, representing multiple profiles of the database, can be generated very quickly to form the relevance feature space. We show that iTrees work well in our framework. Whether there are other ranking models which satisfy the characteristics is an open question.

Next we analyze the difference between our ranking scheme and that measured by distance and similarity. Let $d(\mathbf{a},\mathbf{b})$ and $s(\mathbf{a},\mathbf{b})$ denote the distance value and similarity value, respectively, between two instances **a** and **b**. Then a distance metric and its inversely related similarity measure are required to obey the following four axioms for all instances **a**, **b** and **c** [28]: (i) equal self-similarity: $d(\mathbf{a},\mathbf{a}) = d(\mathbf{b},\mathbf{b})$ and $s(\mathbf{a},\mathbf{a}) = s(\mathbf{b},\mathbf{b})$; (ii) minimality: $d(\mathbf{a},\mathbf{b}) > d(\mathbf{a},\mathbf{a})$ and $s(\mathbf{a},\mathbf{b}) < s(\mathbf{a},\mathbf{a})$; (iii) symmetry: $d(\mathbf{a},\mathbf{b}) = d(\mathbf{b},\mathbf{a})$ and $s(\mathbf{a},\mathbf{b}) = s(\mathbf{b},\mathbf{a})$; (iv) triangle inequality: $d(\mathbf{a},\mathbf{b}) + d(\mathbf{b},\mathbf{c}) > d(\mathbf{a},\mathbf{c})$, and if **a** and **b** are similar and **b** and **c** are similar, then **a** and **c** must also be similar.

The score calculated by Eq. (3) does not satisfy any of the above axioms. For example, symmetry does not hold in our calculation since $Score(\mathbf{a}|\mathbf{b}) - Score(\mathbf{b}|\mathbf{a}) = \sum_{i=1}^{t} (\ell_i(\mathbf{b}) - \ell_i(\mathbf{a}))$, which is not 0 in most cases. The violation of the axioms provides our ranking scheme more flexibility when ranking instances with respect to a query. In fact, questions have been raised about the practical validity of each of these axioms [28]. To the best of our knowledge, there is no other CBMIR ranking scheme that violates all the axioms.

In the anomaly detection setting [23], instances are anomalies if they are irrelevant to the various profiles modeled by different iTrees, i.e., if they have short average path lengths in an iForest model. Thus, the anomaly scoring formulation given in [23] can be rewritten as $Score_{AD}(\mathbf{x}) = \frac{1}{t}\sum_{i=1}^{t} \ell_i(\mathbf{x})$, where high scores indicate normal points, and low scores indicate anomalies. The above anomaly scoring formulation is only different from Eq. (3) by one term, which is the feature weight $w_i(\mathbf{q})$. We show that, under CBMIR, this term effectively modifies the ranking scheme from providing an ordering from normal points to anomalies under anomaly detection, to providing an ordering from instances most relevant to those most irrelevant with respect to the query **q**.

7. Conclusions

This paper proposes a novel ranking framework for CBMIR with relevance feature mapping derived from an ensemble of ranking models. We employ an ensemble of iTrees to map instances from the original feature space to the proposed new relevance feature space. We show that the new relevance feature space has richer information than the original one for ranking database instances with respect to a given query as well as subsequent feedbacks. We also show that the relevance feature space accounts for the significant performance improvement of several existing methods when compared to the same methods applied in the original feature space. Moreover, our experiments validate the utility of our relevance feature weighting, on which the proposed new ranking scheme is based. The new scheme performs better than the four existing methods when they are evaluated in the same footing, in terms of both retrieval performance and time complexity.

The proposed framework has the following unique characteristics: (i) it utilizes no distance measure and has linear time and space complexities with respect to the database size when building its model and mapping the database off-line; (ii) it has constant on-line retrieval time, irrespective of the number of relevance feedback rounds; (iii) it can deal with high-dimensional databases with constant time complexity, once the number of relevance features is fixed; and (iv) it has a good tolerance to irrelevant features.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under No. 61070097 and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20100131110021.

We are grateful for the anonymous reviewers for their suggestions and comments, which have led to significant improvements of this paper. Zhi-Hua Zhou had given Guang-Tong a strong foundation in CBMIR when Guang-Tong visited Nanjing University for six months prior to this project. We would also like to thank Zhouyu Fu for his many helpful discussions and technical assistance with regard to the music data set.

Appendix A. Isolation forest

This section briefly introduces the methodology of iForest [23], which employs a two-stage process to detect anomalies. We provide some insights on how each iTree measures the relevance of instances with respect to a profile underlying the data. It helps to understand the relevance feature space.

In the first stage, iForest builds a collection of iTrees using fixed-sized random sub-samples of a data set. Each iTree is constructed by recursively random-partitioning the sub-sample along axis-parallel coordinates until every instance is isolated from the rest of instances or a specified height limit is reached. The algorithmic details are given by Algorithms 1 and 2. Note that an iTree models a profile of the given random sub-sample, and different iTrees describe different profiles due to the randomness incurred in both the sub-sampling process and the tree building process.

Algorithm 1. *iForest*($\mathcal{D}$,t,ψ).

```
input : $\mathcal{D}$ - input data, $t$ - number of iTrees, $\psi$ - sub-sample size
output: a set of $t$ iTrees
set height limit $h = \lceil \log_2(\psi) \rceil$;
for $i=1$ to $t$ do
    $D \leftarrow sample(\mathcal{D},\psi)$; // randomly sample $\psi$ instances from $\mathcal{D}$
    $T_i \leftarrow iTree(D,0,h)$;
end
```

Algorithm 2. *iTree*(D,e,h).

```
input : $D$ - input data, $e$ - current tree height, $h$ - height limit
output: an iTree
if $e \geq h$ or $|D| \leq 1$ then
    return $exNode\{Size \leftarrow |D|\}$; // an external node
else
    randomly select an attribute $a$ from the data $D$;
    randomly select a split point $p$ from max and min values of attribute $a$ in $D$;
    $D_l \leftarrow filter(D, a < p)$; // instances in $D$ which have values less than $p$ on attribute $a$
    $D_r \leftarrow filter(D, a \geq p)$; // instances in $D$ which have values greater than or equal to $p$ on attribute $a$
    return $inNode\{SplitAtt \leftarrow a, SplitValue \leftarrow p,$
        $Left \leftarrow iTree(D_l, e+1, h), Right \leftarrow iTree(D_r, e+1, h)\}$;
        // an internal node
end
```

In the second stage, iForest calculates an anomaly score for each test instance based on its average path length over all iTrees. A path length is estimated by counting the number of edges from the root node to the external node as an instance travels through the iTree. If the instance falls into an external node with $Size > 1$, the returned path length is adjusted by adding $c(Size)$, which is defined in Eq. (2) and accounts for the average path length of an unbuilt subtree beyond the height limit. This process is given by Algorithm 3.

Algorithm 3. *PathLength*($\mathbf{x}$,T,e).

```
input : $\mathbf{x}$ - an instance, $T$ - an iTree, $e$ - current path length (to be initialized to 0 when first called)
output: the path length of $\mathbf{x}$
if $T$ is an external node then
    return $e + c(T.Size)$; // $c(\cdot)$ is defined in Eq. (2)
end
$a \leftarrow T.SplitAtt$, $p \leftarrow T.SplitValue$;
if $\mathbf{x}_a < p$ then
    return $PathLength(\mathbf{x}, T.Left, e+1)$;
else
    return $PathLength(\mathbf{x}, T.Right, e+1)$;
end
```

Here, a short path length means that we can easily isolate the instance from the majority of instances by a few random partitions. Thus, instances having short path lengths always differ from the majorities on some characteristics. Note that an iTree describes a data profile from a given sub-sample. Therefore, instances having short path length have different data characteristics to the majorities which have long path lengths. Thus, the path length stipulated by an iTree actually measures the relevance of an instance with respect to the profile modeled by this iTree: a short (long) path length indicates that the instance is irrelevant (relevant) to the profile. For anomaly detection, instances identified to be irrelevant to the various profiles modeled by a number of iTrees are deemed to be anomalies, and instances relevant to the various profiles are normal points.

References

[1] M.S. Lew, N. Sebe, C. Djeraba, R. Jain, Content-based multimedia information retrieval: state of the art and challenges, ACM Transactions on Multimedia Computing, Communications, and Applications 2 (1) (2006) 1–19.

[2] R. Zhang, Z.M. Zhang, BALAS: empirical Bayesian learning in the relevance feedback for image retrieval, Image and Vision Computing 24 (3) (2006) 211–223.

[3] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12) (2000) 1349–1380.

[4] R. Datta, D. Joshi, J. Li, J.Z. Wang, Image retrieval: ideas, influences and trends of the new age, ACM Computing Surveys 40 (2) (2008) 1–60 (Article 5).

[5] R. Typke, F. Wiering, R.C. Veltkamp, A survey of music information retrieval systems, in: Proceedings of the Sixth International Conference on Music Information Retrieval, London, UK, 2005, pp. 153–160.

[6] C. Weihs, U. Ligges, F. Mörchen, D. Müllensiefen, Classification in music research, Advances in Data Analysis and Classification 1 (3) (2007) 255–291.

[7] Y. Rui, T.S. Huang, M. Ortega, S. Mehrotra, Relevance feedback: a power tool for interactive content-based image retrieval, IEEE Transactions on Circuits and Systems for Video Technology 8 (5) (1998) 644–655.

[8] X.S. Zhou, T.S. Huang, Relevance feedback in image retrieval: a comprehensive review, Multimedia Systems 8 (6) (2003) 536–544.

[9] J. He, M. Li, H. Zhang, H. Tong, C. Zhang, Manifold-ranking based image retrieval, in: Proceedings of the Twentieth ACM International Conference on Multimedia, New York, 2004, pp. 9–16.

[10] G. Giacinto, F. Roli, Instance-based relevance feedback for image retrieval, in: Advances in Neural Information Processing Systems, vol. 17, Vancouver, Canada, 2005, pp. 489–496.

[11] Z.-H. Zhou, H.-B. Dai, Query-sensitive similarity measure for content-based image retrieval, in: Proceedings of the Sixth IEEE International Conference on Data Mining, Hong Kong, China, 2006, pp. 1211–1215.

[12] D. Zhou, J. Weston, A. Gretton, O. Bousquet, B. Schölkopf, Ranking on data manifolds, in: Advances in Neural Information Processing Systems, vol. 16, Vancouver, Canada, 2003, pp. 169–176.

[13] A. Frome, Y. Singer, F. Sha, J. Malik, Learning globally consistent local distance functions for shape-based image retrieval and classification, in: Proceedings of the Eleventh International Conference on Computer Vision, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[14] J.-E. Lee, R. Jin, A. K. Jain, Rank-based distance metric learning: an application to image retrieval, in: Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, pp. 1–8.

[15] G. Wu, E. Y. Chang, N. Panda, Formulating context-dependent similarity functions, in: Proceedings of the Thirteenth ACM International Conference on Multimedia, Singapore, 2005, pp. 725–734.

[16] X. He, W.-Y. Ma, H. Zhang, Learning an image manifold for retrieval, in: Proceedings of the Twentieth ACM International Conference on Multimedia, New York, 2004, pp. 17–23.

[17] Y.-Y. Lin, T.-L. Liu, H.-T. Chen, Semantic manifold learning for image retrieval, in: Proceedings of the Thirteenth ACM International Conference on Multimedia, Singapore, 2005, pp. 249–258.

[18] Y. Rui, T.S. Huang, S. Mehrotra, Content-based image retrieval with relevance feedback in MARS, in: Proceedings the 1997 International Conference on Image Processing, Washington, DC, 1997, pp. 815–818.

[19] N. Panda, E.Y. Chang, Efficient top-k hyperplane query processing for multimedia information retrieval, in: Proceedings of the Fourteenth ACM International Conference on Multimedia, Santa Barbara, CA, 2006, pp. 317–326.

[20] J.A. Aslam, M.H. Montague, Models for metasearch, in: Proceedings of the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, LA, 2001, pp. 275–284.

[21] N. Rasiwasia, P.J. Moreno, N. Vasconcelos, Bridging the gap: query by semantic example, IEEE Transactions on Multimedia 9 (5) (2007) 923–938.

[22] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison Wesley Longman, Boston, MA, 1999.

[23] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 413–422, software download at ⟨http://sourceforge.net/projects/iforest/⟩.

[24] Z.-H. Zhou, K.-J. Chen, H.-B. Dai, Enhancing relevance feedback in image retrieval using unlabeled data, ACM Transactions on Information Systems 24 (2) (2006) 219–244.

[25] G. Tzanetakis, P.R. Cook, Musical genre classification of audio signals, IEEE Transactions on Speech and Audio Processing 10 (5) (2002) 293–302.

[26] M.I. Mandel, D. Ellis, Song-level features and support vector machines for music classification, in: Proceedings of the Sixth International Conference on Music Information Retrieval, London, UK, 2005, pp. 594–599.

[27] C.J. Krebs, Ecological Methodology, HarperCollins, New York, 1989.

[28] S. Santini, R. Jain, Similarity measures, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (9) (1999) 871–883.

Guang-Tong Zhou received his B.Sc. and M.Sc. degrees from Shandong University in 2007 and 2010, respectively. He is currently a Ph.D. candidate at School of Computing Science, Simon Fraser University. His research interests include data mining, machine learning and their applications to content-based image retrieval, fingerprint recognition and social network analysis.

Kai Ming Ting received his Ph.D. from the University of Sydney, Australia. Later, he worked at the University of Waikato, New Zealand and Deakin University, Australia. He joined Monash University since 2001 and currently serves as the Associate Dean Research Training in Faculty of Information Technology and an Associate Professor in Gippsland School of Information Technology at Monash University. He had previously held visiting positions at Osaka University, Japan, Nanjing University, China, and Chinese University of Hong Kong.

His current research interests are in the areas of mass estimation and mass-based approaches, ensemble approaches, data stream data mining, and swarm intelligence. He is an associate editor for Journal of Data Mining and Knowledge Discovery. He had co-chaired the Pacific-Asia Conference on Knowledge Discovery and Data Mining 2008 and will co-chaired the Pacific Rim International Conference on Artificial Intelligence 2012. He had served as a member of program committees for a number of international conferences including ACM SIGKDD, IEEE ICDM, and ICML. His research projects are supported by grants from Australian Research Council, US Air Force of Scientific Research (AFOSR/AOARD), and Australian Institute of Sport.

Fei Tony Liu received his Ph.D. in 2011 from Monash University. During his post-graduate studies, he was awarded with the Best Paper and Best Student Paper Awards in the Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006) and the Runner-Up Best Theoretical/Algorithms Paper Award in the IEEE International Conference on Data Mining (ICDM 2008). His research interests include ensemble learning, outlier detection, and predictive classification.

Yilong Yin is the Director of MLA Group and a Professor of Shandong University. He received his Ph.D. degree in 2000 from Jilin University. From 2000 to 2002, he worked as a post-doctoral fellow in the Department of Electronic Science and Engineering, Nanjing University. His research interests include machine learning, data mining, and biometrics.