EMR: A Scalable Graph-based Ranking Model for Content-based Image Retrieval

Bin Xu, Student Member, IEEE, Jiajun Bu, Member, IEEE, Chun Chen, Member, IEEE, Can Wang, Member, IEEE, Deng Cai, Member, IEEE, and Xiaofei He, Senior Member, IEEE

Abstract—Graph-based ranking models have been widely applied in information retrieval area. In this paper, we focus on a well known graph-based model - the Ranking on Data Manifold model, or Manifold Ranking (MR). Particularly, it has been successfully applied to content-based image retrieval, because of its outstanding ability to discover underlying geometrical structure of the given image database. However, manifold ranking is computationally very expensive, which significantly limits its applicability to large databases especially for the cases that the queries are out of the database (new samples). We propose a novel scalable graph-based ranking model called Efficient Manifold Ranking (EMR), trying to address the shortcomings of MR from two main perspectives: scalable graph construction and efficient ranking computation. Specifically, we build an anchor graph on the database instead of a traditional \( k \)-nearest neighbor graph, and design a new form of adjacency matrix utilized to speed up the ranking. An approximate method is adopted for efficient out-of-sample retrieval. Experimental results on some large scale image databases demonstrate that EMR is a promising method for real world retrieval applications.

Index Terms—Graph-based algorithm, ranking model, image retrieval, out-of-sample

1 INTRODUCTION

Graph-based ranking models have been deeply studied and widely applied in information retrieval area. In this paper, we focus on the problem of applying a novel and efficient graph-based model for content-based image retrieval (CBIR), especially for out-of-sample retrieval on large scale databases.

Traditional image retrieval systems are based on keyword search, such as Google and Yahoo image search. In these systems, a user keyword (query) is matched with the context around an image including the title, manual annotation, web document, etc. These systems don’t utilize information from images. However these systems suffer many problems, such as shortage of the text information and inconsistency of the meaning of the text and image. Content-based image retrieval is a considerable choice to overcome these difficulties. CBIR has drawn a great attention in the past two decades [1]–[3]. Different from traditional keyword search systems, CBIR systems utilize the low-level features, including global features (e.g., color moment, edge histogram, LBP [4]) and local features (e.g., SIFT [5]), automatically extracted from images. A great amount of researches have been performed for designing more informative low-level features to represent images, or better metrics (e.g., DPF [6]) to measure the perceptual similarity, but their performance is restricted by many conditions and is sensitive to the data. Relevance feedback [7] is a useful tool for interactive CBIR. User’s high level perception is captured by dynamically updated weights based on the user’s feedback.

Most traditional methods focus on the data features too much but they ignore the underlying structure information, which is of great importance for semantic discovery, especially when the label information is unknown. Many databases have underlying cluster or manifold structure. Under such circumstances, the assumption of label consistency is reasonable [8], [9]. It means that those nearby data points, or points belong to the same cluster or manifold, are very likely to share the same semantic label. This phenomenon is extremely important to explore the semantic relevance when the label information is unknown. In our opinion, a good CBIR system should consider images’ low-level features as well as the intrinsic structure of the image database.

Manifold Ranking (MR) [9], [10], a famous graph-based ranking model, ranks data samples with respect to the intrinsic geometrical structure collectively revealed by a large number of data. It is exactly in line with our consideration. MR has been widely applied in many applications, and shown to have excellent performance and feasibility on a variety of data types, such as the text [11], image [12], [13], and video[14]. By taking the underlying structure into account, manifold ranking assigns each data sample a relative ranking score, instead of an absolute pairwise similarity as traditional ways. The score is treated as a similarity...
metric defined on the manifold, which is more meaningful to capturing the semantic relevance degree. He et al. [12] firstly applied MR to CBIR, and significantly improved image retrieval performance compared with state-of-the-art algorithms.

However, manifold ranking has its own drawbacks to handle large scale databases – it has expensive computational cost, both in graph construction and ranking computation stages. Particularly, it is unknown how to handle an out-of-sample query (a new sample) efficiently under the existing framework. It is unacceptable to recompute the model for a new query. That means, original manifold ranking is inadequate for a real world CBIR system, in which the user provided query is always an out-of-sample.

In this paper, we extend the original manifold ranking and propose a novel framework named Efficient Manifold Ranking (EMR). We try to address the shortcomings of manifold ranking from two perspectives: the first is scalable graph construction; and the second is efficient computation, especially for out-of-sample retrieval. Specifically, we build an anchor graph on the database instead of the traditional k-nearest neighbor graph, and design a new form of adjacency matrix utilized to speed up the ranking computation. The model has two separate stages: an offline stage for building a new query. With EMR, we can handle a database with 1 million images and do the online retrieval in a short time. To the best of our knowledge, no previous manifold ranking based algorithm has run out-of-sample retrieval on a database in this scale.

A preliminary version of this work previously appeared as [13]. In this paper, the new contributions are as follows:

- We pay more attention to the out-of-sample retrieval (online stage) and propose an efficient approximate method to compute ranking scores for a new query in Section 4.5. As a result, we can run out-of-sample retrieval on a large scale database in a short time.
- We have optimized the EMR code\(^1\) and re-run all the experiments (Section 5). Three new databases including two large scale databases with about 1 million samples are added for testing the efficiency of the proposed model. We offer more detailed analysis for experimental result.
- We formally define the formulation of local weight estimation problem (Section 4.1.1) for building the anchor graph and two different methods are compared to determine which method is better (Section 5.2.2).

The rest of this paper is organized as follows. In Section 2, we briefly discuss some related work and in Section 3, we review the algorithm of MR and make an analysis. The proposed approach EMR is described in Section 4. In Section 5, we present the experiment results on many real world image databases. Finally we provide a conclusions in Section 6.

1. http://eagle.zju.edu.cn/~binxu/

## 2 RELATED WORK

The problem of ranking has recently gained great attentions in both information retrieval and machine learning areas. Conventional ranking models can be content based models, like the Vector Space Model, BM25, and the language modeling [15]; or link structure based models, like the famous PageRank [16] and HITS [17]; or cross media models [18]. Another important category is the learning to rank model, which aims to optimize a ranking function that incorporates relevance features and avoids tuning a large number of parameters empirically [19], [20]. However, many conventional models ignore the important issue of efficiency, which is crucial for a real-time systems, such as a web application. In [21], the authors present a unified framework for jointly optimizing effectiveness and efficiency.

In this paper, we focus on a particular kind of ranking model – graph-based ranking. It has been successfully applied in link-structure analysis of the web [16], [17], [22]–[24], social networks research [25]–[27] and multimedia data analysis [28]. Generally, a graph [29] can be denoted as \( G = (V, E, W) \), where \( V \) is a set of vertices in which each vertex represents a data point, \( E \subseteq V \times V \) is a set of edges connecting related vertices, and \( W \) is a adjacency matrix recording the pairwise weights between vertices. The object of a graph-based ranking model is to decide the importance of a vertex, based on local or global information draw from the graph.

Agarwal [30] proposed to model the data by a weighted graph, and incorporated this graph structure into the ranking function as a regularizer. Guan et al. [26] proposed a graph-based ranking algorithm for interrelated multi-type resources to generate personalized tag recommendation. Liu et al. [25] proposed an automatically tag ranking scheme by performing a random walk over a tag similarity graph. In [27], the authors made the music recommendation by ranking on a unified hypergraph, combining with rich social information and music content. Hypergraph is a new graph-based model and has been studied in many works [31]. Recently, there have been some papers on speeding up manifold ranking. In [32], the authors partitioned the data into several parts and computed the ranking function by a block-wise way.

## 3 MANIFOLD RANKING REVIEW

In this section, we briefly review the manifold ranking algorithm and make a detailed analysis about its drawbacks. We start form the description of notations.

### 3.1 Notations and Formulations

Given a set of data \( \chi = \{x_1, x_2, \ldots, x_d\} \subset R^n \) and build a graph on the data (e.g., kNN graph). \( W \in R^{n \times n} \) denotes the adjacency matrix with element \( w_{ij} \) saving the weight of the edge between point \( i \) and \( j \). Normally the weight can be defined by the heat kernel \( w_{ij} = \exp \left( -d^2(x_i, x_j)/2\sigma^2 \right) \) if there is an edge linking \( x_i \) and \( x_j \), otherwise \( w_{ij} = 0 \). Function \( d(x_i, x_j) \) is a distance metric of \( x_i \) and \( x_j \) defined on \( \chi \), such as the Euclidean distance. Let \( r: \chi \rightarrow R \) be a ranking function which assigns to each point \( x_i \) a ranking score \( r_i \). Finally, we define an initial vector \( y = [y_1, \ldots, y_n]^T \), in which \( y_i = 1 \) if \( x_i \) is a query and \( y_i = 0 \) otherwise.
The cost function associated with $r$ is defined to be
\[
O(r) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left( \frac{1}{\sqrt{D_{ii}}} r_i - \frac{1}{\sqrt{D_{jj}}} r_j \right)^2 + \mu \sum_{i=1}^{n} \| r_i - y_i \|^2 ,
\]
(1)
where $\mu > 0$ is the regularization parameter and $D$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$.

The first term in the cost function is a smoothness constraint, which makes the nearby points in the space having close ranking scores. The second term is a fitting constraint, which makes the nearby points in the space having close ranking scores. The second term is a fitting constraint, which means the ranking result should fit to the initial label assignment. With more prior knowledge about the relevance or confidence of each query, we can assign different initial scores to the queries. Minimizing the cost function respect to $r$ results into the following closed form solution
\[
r^* = (I_n - \alpha S)^{-1} y ,
\]
(2)
where $\alpha = \frac{1}{1+\mu}$, $I_n$ is an identity matrix with $n \times n$, and $S$ is the symmetrical normalization of $W$, $S = D^{-1/2}WD^{-1/2}$. In large scale problems, we prefer to use the iteration scheme:
\[
r(t+1) = \alpha S r(t) + (1 - \alpha) y .
\]
(3)

During each iteration, each point receives information from its neighbors (first term), and retains its initial information (second term). The iteration process is repeated until convergence. When manifold ranking is applied to retrieval (such as image retrieval), after specifying a query by the user, we can use the closed form or iteration scheme to compute the ranking score of each point. The ranking score can be viewed as a metric of the manifold distance which is more meaningful to measure the semantic relevance.

### 3.2 Analysis

Although manifold ranking has been widely used in many applications, it has its own drawbacks to handle large scale databases, which significantly limits its applicability.

The first is its graph construction method. The kNN graph is quite appropriate for manifold ranking because of its good ability to capture local structure of the data. But the construction cost for kNN graph is $O(n^2 \log k)$, which is expensive in large scale situations. Moreover, manifold ranking, as well as many other graph-based algorithms directly use the adjacency matrix $W$ in their computation. The storage cost of a sparse $W$ is $O(kn)$. Thus, we need to find a way to build a graph in both low construction cost and small storage space, as well as good ability to capture underlying structure of the given database.

The second, manifold ranking has very expensive computational cost because of the matrix inversion operation in equation (2). This has been the main bottleneck to apply manifold ranking in large scale applications. Although we can use the iteration algorithm in equation (3), it is still inefficient in large scale cases and may arrive at a local convergence. Thus, original manifold ranking is inadequate for a real-time retrieval system.

### 4 Efficient Manifold Ranking

We address the shortcomings of original MR from two perspectives: scalable graph construction and efficient ranking computation. Particularly, our method can handle the out-of-sample retrieval, which is important for a real-time retrieval system.

#### 4.1 Scalable Graph Construction

To handle large databases, we want the graph construction cost to be sub-linear with the graph size. That means, for each data point, we can’t search the whole database, as kNN strategy does. To achieve this requirement, we construct an anchor graph [33], [34] and propose a new design of adjacency matrix $W$.

The definitions of anchor points and anchor graph have appeared in some other works. For instance, in [35], the authors proposed that each data point on the manifold can be locally approximated by a linear combination of its nearby anchor points, and the linear weights become its local coordinate coding. Liu et al. [33] designed the adjacency matrix in a probabilistic measure and used it for scalable semi-supervised learning. This work inspires us much.

#### 4.1.1 Anchor Graph Construction

Now we introduce how to use anchor graph to model the data [33], [34]. Suppose we have a data set $\chi = \{x_1, \ldots, x_n\} \subset R^m$ with $n$ samples in $m$ dimensions, and $U = \{u_1, \ldots, u_d\} \subset R^m$ denotes a set of anchors sharing the same space with the data set. Let $f: \chi \rightarrow R$ be a real value function which assigns each data point in $\chi$ a semantic label. We aim to find a weight matrix $Z \in R^{d \times n}$ that measures the potential relationships between data points in $\chi$ and anchors in $U$. Then we estimate $f(x)$ for each data point as a weighted average of the labels on anchors
\[
\hat{f}(x_i) = \sum_{k=1}^{d} z_{ki} f(u_k), i = 1, \ldots, n,
\]
(4)
with constraints $\sum_{k=1}^{d} z_{ki} = 1$ and $z_{ki} \geq 0$. Element $z_{ki}$ represents the weight between data point $x_i$ and anchor $u_k$. The key point of the anchor graph construction is how to compute the weight vector $z_i$ for each data point $x_i$. Two issues need to be considered: (1) the quality of the weight vector and (2) the cost of the computation.

Similar to the idea of LLE [8], a straightforward way to measure the local weight is to optimize the following convex problem:

\[
\min_{z_i} \epsilon(z_i) = \frac{1}{2} \| x_i - \sum_{s=1}^{N(x_i)} u_{s \in N(x_i)} z_{is} \|^2 \\
\text{s.t.} \quad \sum_{s} z_{is} = 1, \quad z_{is} \geq 0,
\]
(5)

where $N(x_i)$ is the index set of $x_i$’s nearest anchors. We call the above problem as the local weight estimation problem. A standard quadratic programming (QP) can solve this problem, but QP is very computational expensive. A projected gradient based algorithm was proposed in [33] to compute weight matrix and in our previous work [13], a kernel regression method was adopted. In this paper, we compare these two different methods to find the weight vector $z_i$. Both of them are much faster than QP.
(1) Solving by Projected Gradient
The first method is the projected gradient method, which has been used in the work of [33]. The updating rule in this method is expressed as the following iterative formula [33]:
$$ z_i^{(t+1)} = \Pi_s(z_i^{(t)} - \eta_t \nabla E(z_i^{(t)})), $$
where $\eta_t$ denotes the step size of time $t$, $\nabla E(z)$ denotes the gradient of $E$ at $z$, and $\Pi_s(z)$ denotes the simplex projection operator on any $z \in \mathbb{R}^s$. Detailed algorithm can be found in Algorithm 1 of [33].

(2) Solving by Kernel Regression
We adopt the Nadaraya-Watson kernel regression to assign weights smoothly [13]
$$ z_{ki} = \frac{K\left(\frac{|x_i - u_k|}{\lambda}\right)}{\sum_{i=1}^{d} K\left(\frac{|x_i - u_k|}{\lambda}\right)}, $$
with the Epanechnikov quadratic kernel
$$ K_i(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \leq 1; \\ 0 & \text{otherwise.} \end{cases} $$
The smoothing parameter $\lambda$ determines the size of the local region in which anchors can affect the target point. It is reasonable to consider that one data point has the same semantic label with its nearby anchors in a high probability. There are many ways to determine the parameter $\lambda$. For example, it can be a constant selected by cross-validation from a set of training data. In this paper we use a more example, it can be a constant selected by cross-validation $\lambda$. There are many ways to determine the parameter $\lambda$. For example, it can be a constant selected by cross-validation from a set of training data. In this paper we use a more example, it can be a constant selected by cross-validation $\lambda$.

The above two methods.
Specifically, to build the anchor graph, we connect each sample to its $s$ nearest anchors and then assign the weights. So the construction has a total complexity $O(n d \log s)$, where $d$ is the number of anchors and $s$ is very small. Thus, the number of anchors determines the efficiency of the anchor graph construction. If $d \ll n$, the construction is linear to the database.

How can we get the anchors? Active learning [36], [37] or clustering methods are considerable choices. In this paper, we use k-means algorithm and select the centers as anchors. Some fast k-means algorithms [38] can speed up the computation. Random selection is a competitive method which has extremely low selection cost and acceptable performance.

The main feature, also the main advantage of building an anchor graph is separating the graph construction into two parts – anchor selection and graph construction. Each data sample is independent to the other samples but related to the anchors only. The construction is always efficient since it has linear complexity to the date size. Note that we don’t have to update the anchors frequently, as informative anchors for a large database are relatively stable (e.g., the cluster centers), even if a few new samples are added.

4.1.2 Design of Adjacency Matrix
We present a new approach to design the adjacency matrix $W$ and make an intuitive explanation for it. The weight matrix $Z \in \mathbb{R}^{d \times n}$ can be seen as a $d$ dimensional representation of the data $X \in \mathbb{R}^{n \times n}$, $d$ is the number of anchor points. That is to say, data points can be represented in the new space, no matter what the original features are. This is a big advantage to handle some high dimensional data. Then, with the inner product as the metric to measure the adjacent weight between data points, we design the adjacency matrix to be a low-rank form [33], [39]
$$ W = Z^T Z, $$
which means that if two data points are correlative ($W_{ij} > 0$), they share at least one common anchor point, otherwise $W_{ij} = 0$. By sharing the same anchors, data points have similar semantic concepts in a high probability as our consideration. Thus, our design is helpful to explore the semantic relationships in the data.

This formula naturally preserves some good properties of $W$: sparseness and nonnegativeness. The highly sparse matrix $Z$ makes $W$ sparse, which is consistent with the observation that most of the points in a graph have only a small amount of edges with other points. The nonnegative property makes the adjacent weight more meaningful: in real world data, the relationship between two items is always positive or zero, but not negative. Moreover, nonnegative $W$ guarantees the positive semidefinite property of the graph Laplacian in many graph-based algorithms [33].

4.2 Efficient Ranking Computation
After graph construction, the main computational cost for manifold ranking is the matrix inversion in equation (2), whose complexity is $O(n^3)$. So the data size $n$ can not be too large. Although we can use the iteration algorithm, it is still inefficient for large scale cases.

One may argue that the matrix inversion can be done off-line, then it is not a problem for on-line search. However, off-line calculation can only handle the case when the query is already in the graph (an in-sample). If the query is not in the graph (an out-of-sample), for exact graph structure, we have to update the whole graph to add the new query and compute the matrix inversion in equation (2) again. Thus, the off-line computation doesn’t work for an out-of-sample query. Actually, for a real CBIR system, user’s query is always an out-of-sample.

With the form of $W = Z^T Z$, we can rewrite the equation (2), the main step of manifold ranking, by Woodbury formula as follows. Let $H = Z D^{-\frac{1}{2}}$, and $S = H^T H$, then the final ranking function $r$ can be directly computed by
$$ r^* = (I_n - \alpha H^T H)^{-1} y = \left( I_n - H^T \left( HH^T - \frac{1}{\alpha} I_d \right)^{-1} H \right) y. $$

By equation (11), the inversion part (taking the most computational cost) changes from a $n \times n$ matrix to a $d \times d$ matrix. If $d \ll n$, this change can significantly speed up the calculation of manifold ranking. Thus, applying our proposed method to a real-time retrieval system is viable, which is a big shortage for original manifold ranking.

During the computation process, we never use the adjacency matrix $W$. So we don’t save the matrix $W$ in memory,
but save matrix $Z$ instead. In equation (11), $D$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$. When $W = Z^T Z$,

$$D_{ii} = \sum_{j=1}^{n} z_i^T z_j = z_i^T v,$$  \hspace{1cm} (12)

where $z_i$ is the $i$th column of $Z$ and $v = \sum_{j=1}^{n} z_i$. Thus we get the matrix $D$ without using $W$.

A useful trick for computing $r^*$ in equation (11) is running it from right to left. So every time we multiply a matrix by a vector, avoiding the matrix - matrix multiplication. As a result, to compute the ranking function, EMR has a complexity $O(dn + d^3)$.

### 4.3 Complexity Analysis

In this subsection, we make a comprehensive complexity analysis of MR and EMR, including the computation cost and storage cost. As we have mentioned, both MR and EMR have two stages: the graph construction stage and the ranking computation stage.

For the model of MR:

- MR builds a kNN graph, i.e., for each data sample, we need to calculate the relationships to its $k$-nearest neighbors. So the computation cost is $O(n^2 \log k)$. At the same time, we save the adjacency matrix $W \in R^{n \times n}$ with a storage cost $O(kn)$ since $W$ is sparse.

- In the ranking computation stage, the main step is to compute the matrix inversion in $Z$, which is approximately $O(n^3)$.

For the model of EMR:

- EMR builds an anchor graph, i.e., for each data sample, we calculate the relationships to its $s$-nearest anchors. The computation cost is $O(nd \log s)$. We use $k$-means to select the anchors, we need a cost of $O(Tdn)$, where $T$ is the iteration number. But this selection step can be done off-line and unnecessarily updated frequently. At the same time, we save the sparse matrix $\tilde{Z} \in R^{d \times n}$ with a storage cost $O(sn)$.

- In the ranking computation stage, the main step is Eq.(11), which has a computational complexity of $O(dn + d^3)$.

As a result, EMR has a computational cost of $O(dn) + O(d^3)$ (ignoring $s$, $T$) and a storage cost $O(sn)$, while MR has a computational cost of $O(n^2) + O(n^3)$ and a storage cost $O(kn)$. Obviously, when $d \ll n$, EMR has a much lower cost than MR in computation.

### 4.4 EMR for Content-Based Image Retrieval

In this part, we make a brief summary of EMR applied to pure content-based image retrieval. To add more information, we just extend the data features.

First of all, we extract the low-level features of images in the database, and use them as coordinates of data points in the graph. We will further discuss the low-level features in Section 5. Secondly, we select representative points as anchors and construct the weight matrix $Z$ with a small neighborhood size $s$. Anchors are selected off-line and does not affect the on-line process. For a stable data set, we don’t frequently update the anchors. At last, after the user specifying or uploading an image as a query, we get or extract its low-level features, update the weight matrix $Z$, and directly compute the ranking scores by equation (11). Images with highest ranking scores are considered as the most relevant and return to the user.

### 4.5 Out-of-Sample Retrieval

For in-sample data retrieval, we can construct the graph and compute the matrix inversion part of equation (2) offline. But for out-of-sample data, the situation is totally different. A big limitation of MR is that, it is hard to handle the new sample query. A fast strategy for MR is leaving the original graph unchanged and adding a new row and a new column to $W$ (left picture of Fig. 1). Although the new $W$ is efficiently to compute, it is not helpful for the ranking process (Eq.(2)). Computing Eq.(2) for each new query in the online stage is unacceptable due to its high computational cost.

In [40], the authors solve the out-of-sample problem by finding the nearest neighbors of the query and using the neighbors as query points. They don’t add the query into the graph, therefore their database is static. However, their method may change the query’s initial semantic meaning, and for a large database, the linear search for nearest neighbors is also costly.

In contrast, our model EMR can efficiently handle the new sample as a query for retrieval. In this subsection, we describe the light-weight computation of EMR for a new sample query. We want to emphasize that this is a big improvement over our previous conference version of this work, which makes EMR scalable for large-scale image databases (e.g., 1 million samples). We show the algorithm as follows.

For one instant retrieval, it is unwise to update the whole graph or rebuild the anchors, especially on a large database. We believe one point has little effect to the stable anchors in a large data set (e.g., cluster centers). For EMR, each data point ($z_i$) is independently computed, so we assign weights between the new query and its nearby anchors, forming a new column of $Z$ (right picture of Fig. 1).

We use $z_i$ to denote the new column. Then, $D_i = z_i^T v$ and $h_i = z_i D_i^{-\frac{1}{2}}$, where $h_i$ is the new column of $H$. As we have described, the main step of EMR is Eq.(11). Our goal is to further speedup the computation of Eq.(11) for a new query. Let

$$C = \left( HH^T - \frac{1}{\alpha} I_d \right)^{-1} = \left( \sum_{i=1}^{n} h_i h_i^T - \frac{1}{\alpha} I_d \right)^{-1},$$

\hspace{1cm} (13)
and the new $C'$ with adding the column $h_l$ is

$$
C' = \left( \sum_{i=1}^{n} h_i h_i^T + h_l h_l^T - \frac{1}{\alpha} I_d \right)^{-1} \approx C
$$

when $n$ is large and $h_l$ is highly sparse. We can see the matrix $C$ as the inverse of a covariance matrix. The above equation says that one single point would not affect the covariance matrix of a large database. That is to say, the computation of $C$ can be done in the off-line stage.

The initial query vector $y_l$ is

$$
y_l = \begin{bmatrix} 0_n \\ 1 \end{bmatrix},
$$

where $0_n$ is a $n$-length zero vector. We can rewrite Eq.(11) with the new query as

$$
r^{(n+1) \times 1} = (I_{n+1} - [H^T C \ h_l^T C \ | H \ h_l]) \begin{bmatrix} 0_n \\ 1 \end{bmatrix}.
$$

Our focus is the top $n$ elements of $r$, which is equal to

$$
r^{n \times 1} = -H^T C h_l = E h_l.
$$

The matrix $E^{n \times d} = -H^T C$ can be computed offline, i.e., in the online stage, we need to compute a multiplication of a $n \times d$ matrix and a $d \times 1$ vector only. As $h_l$ is sparse (e.g., $s$ non-zero elements), the essential computation is to select $s$ columns of $E$ according to $h_l$ and do a weighted summation. As a result, we need to do $sn$ scalar multiplications and $(s - 1)n$ scalar additions to get the ranking score ($r^{n \times 1}$) for each database sample; while for linear scan using Euclidean distance, we need to do $mn$ scalar subtractions, $mn$ scalar multiplications and $(m - 1)n$ scalar additions. As $s \ll m$, our model EMR is much faster than linear scan using Euclidean distance in the online stage.

5 Experimental Study

In this section, we show several experimental results and comparisons to evaluate the effectiveness and efficiency of our proposed method EMR on four real world databases: two middle size databases COREL (5,000 images) and MNIST (70,000 images), and two large size databases SIFT1M (1 million sif descriptors) and ImageNet (1.2 million images). We use COREL and MNIST to compare the ranking performance and use SIFT1M and ImageNet to show the efficiency of EMR for out-of-sample retrieval. Our experiments are implemented in MATLAB and run on a computer with 2.0 GHz(×2) CPU, 64GB RAM.

5.1 Experiments Setup

The COREL image data set is a subset of COREL image database consisting of 5,000 images. COREL is widely used in many CBIR works [2], [41], [42]. All of the images are from 50 different categories, with 100 images per category. Images in the same category belong to the same semantic concept, such as beach, bird, elephant and so on. That is to say, images from the same category are judged relevant and otherwise irrelevant. We use each image as a query for testing the in-sample retrieval performance. In Fig. 2, we randomly select and show nine image samples from three different categories. In our experiments, we extract four kinds of effective features for COREL database, including Grid Color Moment, edge histogram, Gabor Wavelets Texture, Local Binary Pattern and GIST feature. As a result, a 809-dimensional vector is used for each image [43].

The MNIST database² of handwritten digits has a set of 70,000 examples. The images were centered in a 28 × 28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28 × 28 field. We use the first 60,000 images as database images and the rest 10,000 images as queries for testing the out-of-sample retrieval performance. The normalized gray-scale values for each pixel are used as image features.

The SIFT1M database contains one million SIFT features and each feature is represented by a 128-dimensional vector.

The ImageNet is an image database organized according to the WordNet nouns hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images³. We downloaded about 1.2 million images’ BoW representations. A visual vocabulary of 1,000 visual words is adopted, i.e., each image is represented by a 1,000-length vector. Due to the complex structure of the database and high diversity of images in each node, as well as the low quality of simple BoW representation, the retrieval task is very hard.

We use SIFT1M and ImageNet databases to evaluate the efficiency of EMR on large and high dimensional data. We randomly select 1,000 images as out-of-sample test queries for each. Some basic statistics of the four databases are listed in Table 1. For COREL, MNIST and SIFT1M databases, the data samples have dense features, while for ImageNet database, the data samples have sparse features.

5.1.1 Evaluation Metric Discussion

There are many measures to evaluate the retrieval results such as precision, recall, F measure, MAP and NDCG [44].


<table>
<thead>
<tr>
<th>Table 1: Statistics of the Four Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td># of images</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td># of categories</td>
</tr>
<tr>
<td># of features</td>
</tr>
</tbody>
</table>

Fig. 2. COREL image samples randomly selected from semantic concept balloon, beach, and butterfly.
They are very useful for a real CBIR application, especially for a web application in which only the top returned images can attract user interests. Generally, the image retrieval results are displayed screen by screen. Too many images in a screen will confuse the user and drop the experience evidently. Images in the top pages attract the most interests and attentions from the user. So the precision at K metric is significant to evaluate the image retrieval performance.

MAP (Mean Average Precision) provides a single-figure measure of quality across recall levels. MAP has been shown to have especially good discriminative power and stability. For a single query, Average Precision is the average of the precision value obtained for the set of top k items existing after each relevant item is retrieved, and this value is then averaged over all queries [44]. That is, if the set of relevant items for a query \( q_j \in Q \) is \( \{d_1, \ldots, d_{m_j}\} \) and \( R_{jk} \) is the set of ranked retrieval results from the top result until you get to item \( d_k \), then

\[
\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{m_j}{m} \sum_{k=1}^{|m_j|} \text{Precision}(R_{jk}).
\]

NDCG is a wildly used metric to evaluate a ranked list [44]. NDCG@K is defined as:

\[
\text{NDCG@K} = \frac{1}{\text{IDCG}} \times \sum_{j=1}^K 2^{r_j-1} \frac{1}{\log_2 (j+1)},
\]

where \( r_j \) is 1 if the item at position \( j \) is a relevant item and 0 otherwise. IDCG is chosen so that the perfect ranking has a NDCG value 1.

### 5.2 Experiments on COREL Database

The goal of EMR is to improve the speed of manifold ranking with acceptable ranking accuracy loss. We first compare our model EMR with the original manifold ranking (MR) and fast manifold ranking (FMR) [32]) algorithm on COREL database. As both MR and FMR are designed for in-sample image retrieval, we use each image as a query and evaluate in-sample retrieval performance. More comparison to ranking with SVM can be found in our previous conference version [13]. In this paper, we pay more attention on the trade-off of accuracy and speed for EMR respect to MR, so we ignore the other methods.

We first compare the methods without relevance feedback. Relevance feedback asks users to label some retrieved samples, making the retrieval procedure inconvenient. So if possible, we prefer an algorithm having good performance without relevance feedback. In Section 5.2.4, we evaluate the performance of the methods after one round of relevance feedback. MR-like algorithms can handle the relevance feedback very efficiently - revising the initial score vector \( y \).

#### 5.2.1 Baseline Algorithm

**Eud:** the baseline method using Euclidean distance for ranking.

**MR:** the original manifold ranking algorithm, the most important comparison method. Our goal is to improve the speed of manifold ranking with acceptable ranking accuracy loss.

**FMR:** fast manifold ranking [32] firstly partitions the data into several parts (clustering) and computes the matrix inversion by a block-wise way. It uses the SVD technique which is time consuming. So its computational bottleneck is transformed to SVD. When SVD is accurately solved, FMR equals MR. But FMR uses the approximate solution to speed up the computation. We use 10 clusters and calculate the approximation of SVD with 10 singular values. Higher accuracy requires much more computational time.

#### 5.2.2 Comparisons of Two Weight Estimation Methods for EMR

Before the main experiment of comparing our algorithm EMR to some other models, we use a single experiment to decide which weight estimation method described in Section 4.1.1 should be adopted. We records the average retrieval precision (each image is used as a query) and the computational time (seconds) of EMR with the two weight estimation methods in Table 2.

From the table, we see that the two methods have very close retrieval results. However, the projected gradient is much slower than kernel regression. In the rest of our experiments, we use the kernel regression method to estimate the local weight (computing \( Z \)).

#### 5.2.3 Performance

An important issue needs to be emphasized: although we have the image labels (categories), we don’t use them in our algorithm, since in real world applications, labeling is very expensive. The label information can only be used to evaluation and relevance feedback.

Each image is used as a query and the retrieval performance is averaged. Fig. 3 prints the average precision (at 20 to 80) of each method and Table 3 records the average values of recall, F1 score, NDCG and MAP (MAP is evaluated only for the top-100 returns). For our method EMR, 1000 anchors are used. Later in the model selection part, we find that using 500 anchors achieves a close performance. It is easy to find that the performance of MR and EMR are very close, while FMR lose a little precision due to its approximation by SVD. As EMR’s goal is to improve the speed of manifold ranking with acceptable ranking accuracy loss, the performance results are not to show which method is better but to show the ranking performance of EMR is close to MR on COREL.

We also record the offline building time for MR, FMR and EMR in Table 3. For in-sample retrieval, all the three
methods have the same steps and cost, so we ignore it on COREL. We find that for a database with 5,000 images, all three methods have acceptable building time, and EMR is the most efficient. However, according to the analysis in Section 4.3, MR’s computational cost is cubic to the database size while EMR is linear to the database size. The result can be found in our experiments on MNIST database.

The anchor points are computed off-line and do not affect the current on-line retrieval system. In the work of [13], we have tested different strategies for anchor points selection, including normal k-means, fast k-means and random anchors. The conclusion is that the cost and performance are trade-offs in many situations.

To see the performance distribution in the whole data set more concretely, we plot the retrieval precision at top 10 returns for all 50 categories in Fig. 4. As can be seen, the performance of each algorithm varies with different categories. We find that EMR is fairly close to MR in almost every categories, but for FMR, the distribution is totally different.

5.2.4 Performance with Relevance Feedback
Relevance Feedback [7] is a powerful interactive technique used to improve the performance of image retrieval systems. With user provided relevant/irrelevant information on the retrieved images, The system can capture the semantic concept of the query more correctly and gradually improve the retrieval precision.

Applying relevance feedback to EMR (as well as MR and FMR) is extremely simple. We update the initial vector $y$ and recompute the ranking scores. We use an automatic labeling strategy to simulate relevance feedback: for each query, the top 20 returns’ ground truth labels (relevant or irrelevant to the query) are used as relevance feedbacks. It is performed for one round, since the users have no patience to do more. The retrieval performance are plotted in Fig. 5. By relevance feedback, MR, FMR and EMR get higher retrieval precision but still remain close to each other.

5.2.5 Model Selection
Model selection plays a key role to many machine learning methods. In some cases, the performance of an algorithm may drastically vary by different choices of the parameters, thus we have to estimate the quality of the parameters. In this subsection, we evaluate the performance of our method EMR with different values of the parameters.

There are three parameters in our method EMR: $s$, $\alpha$, and $d$. Parameter $s$ is the neighborhood size in the anchor graph. Small value of $s$ makes the weight matrix $Z$ very sparse. Parameter $\alpha$ is the tradeoff parameter in EMR and MR. Parameter $d$ is the number of anchor points. For
convenience, the parameter $\alpha$ is fixed at 0.99, consistent with the experiments performed in [9], [10], [12].

Fig. 6 shows the performance of EMR (Precision at 60) by k-means anchors at different values of $s$. We find that the performance of EMR is not sensitive to the selection of $s$ when $s > 3$. With small $s$, we can guarantee the matrix $Z$ highly sparse, which is helpful to efficient computation. In our experiments, we just select $s = 5$.

Fig. 7 shows the performance of EMR versus different number of anchors in the whole data set. We find that the performance increases very slowly when the number of anchors is larger than 500 (approximately). In previous experiments, we fix the number of anchors to 1000. Actually, a smaller number of anchors, like 800 or 600 anchors, can achieve a close performance. With fewer anchors, the graph construction cost will be further reduced. But as the size of COREL is not large, the saving is not important.

### 5.3 Experiments on MNIST Database

We also investigate the performance of our method EMR on the MNIST database. The samples are all gray digit images in the size of $28 \times 28$. We just use the gray values on each pixel to represent the images, i.e., for each sample, we use a 784-dimensional vector to represent it. The database was separated into 60,000 training data and 10,000 testing data, and the goal is to evaluate the performance on the testing data. Note that although it is called ‘training data’, a retrieval system never uses the given labels. All the ranking models use the training data itself to build their models and rank the samples according to the queries. Similar idea can be found in many unsupervised hashing algorithms [45], [46] for approximate and fast nearest neighbor search.

With MNIST database, we want to evaluate the efficiency and effectiveness of the model EMR. As we have mentioned, MR’s cost is cubic to the database size, while EMR is much faster. We record the training time (building the model offline) of MR, FMR and EMR (1k anchors) in Table 4 with the database size increasing step by step. The required time for MR and FMR increases very fast and for the last two sizes, their procedures are out of memory due to inverse operation. The algorithm MR with the solution of Eq.(2) is hard to handle the size of MNIST. FMR performs even worse than MR as it clusters the samples and computes a large SVD – it seems that FMR is only useful for small-size database. However, EMR is much faster in this test. The time cost scales linearly – 6 seconds for 10,000 samples and 35 seconds for 60,000 samples. We use k-means algorithm with maximum 5 iterations to generate the anchor points. We find that running k-means with 5 iterations is good enough for anchor point selection.

<table>
<thead>
<tr>
<th># of Samples</th>
<th>MR</th>
<th>FMR</th>
<th>EMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>24.532</td>
<td>46.218</td>
<td>6.328</td>
</tr>
<tr>
<td>20,000</td>
<td>143.312</td>
<td>307.547</td>
<td>13.110</td>
</tr>
<tr>
<td>30,000</td>
<td>422.344</td>
<td>967.484</td>
<td>20.531</td>
</tr>
<tr>
<td>40,000</td>
<td>916.813</td>
<td>2322.625</td>
<td>28.469</td>
</tr>
<tr>
<td>50,000</td>
<td>–</td>
<td>–</td>
<td>29.343</td>
</tr>
<tr>
<td>60,000</td>
<td>–</td>
<td>–</td>
<td>34.500</td>
</tr>
</tbody>
</table>
5.3.1 Out-of-Sample Retrieval Test

In this section, we evaluate the response time of EMR when handling an out-of-sample (a new sample). As MR (as well as FMR)'s framework is hard to handle the out-of-sample query and is too costly for training the model on the size of MNIST (Table 4), from now on, we don’t use MR and FMR as comparisons, but some other ranking score (similarity or distance) generating methods should be compared. We use the following two methods as baseline methods:

**Eud**: linear scan by Euclidean distance. This maybe the most simple but meaningful baseline to compare the out-of-sample retrieval performance. Many previous fast nearest neighbor search algorithms or hashing-based algorithms were proposed to accelerate the linear scan speed with some accuracy loss than Euclidean distance. Their goal is different with ranking – the ranking model assigns each sample a score but not only the neighbors.

**LSH**: locality sensitive hashing [45], a famous hashing code generating method. We use LSH to generate binary codes for the images for both training and testing samples and then calculate the hamming distance of a query to all database samples as ranking metric. We use 128 bits and 256 bits as the code length of LSH.

In Fig. 8(a), we draw the MAP (top 200) values for all the testing data of our model EMR with different numbers of anchor points. The performance of Eud and LSH are showed by three horizontal lines. We can see that, when more than 400 anchors are used, EMR outperforms Euclidean distance metric significantly. LSH is worse than Eud due to its binary representation. We also record EMR’s offline training time and online retrieval time in Fig. 8(b) and Fig. 8(c). The computational time for both offline and online increases linearly to the number of anchors.

Then, in Table 5, we record the computational time (in seconds) and out-of-sample retrieval performance of EMR (1000 anchors), Eud and LSH with 128 and 256 code length. The best performance of each line is in bold font. EMR and LSH-128 have close online retrieval time, which is greater than linear scan Eud – about 30 times faster. LSH has very small training cost as its hashing functions are randomly selected, while EMR needs more time to build the model. With more offline building cost, EMR receives higher retrieval performance in metric of precision, NDCG at 100 and MAP. The offline cost is valuable. The number with * means it is significant higher than Eud at the 0.001 significance level.

5.3.2 Case Study

Fig. 9 is an out-of-sample retrieval case with Fig. 9(a) using Euclidean distance to measure the similarity and Fig. 9(b) using EMR with 400 anchors and Fig. 9(c) with 600 anchors. Since the database structure is simple, we just need to use a small number of anchors to build our anchor graph. When we use 400 anchors, we have received a good result (Fig. 9(b)). Then, when we use more anchors, we can get a better result. It is not hard to see that, the results of Fig. 9(b) and (c) are all correct, but the quality of Fig. 9(c) is a little better – the digits are more similar with the query.

5.4 Experiments on Large Scale Databases

In our consideration, the issue of performance should include both efficiency and effectiveness. Since our method is designed to speedup the model ‘manifold ranking’, the efficiency is the main point of this paper. The first several experiments are used to show that our model is much faster than MR in both offline training and online retrieval processes, with only a small accuracy loss. The original MR model can not be directly applied to a large data set, e.g., a data set with 1 million samples. Thus, to show the performance of our method for large data sets, we compare many state-of-the-art hash-based fast nearest neighbor search algorithms (our ranking model can naturally do the

<table>
<thead>
<tr>
<th>TABLE 5</th>
<th>Out-of-Sample Retrieval Time (s) and Retrieval Performance Comparisons of EMR (1k Anchors), Eud and LSH with 128 and 256 Code Length on MNIST Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eud</td>
<td>LSH-128</td>
</tr>
<tr>
<td>Offline training time</td>
<td>0.723</td>
</tr>
<tr>
<td>Online retrieval time</td>
<td>0.887</td>
</tr>
<tr>
<td>@100</td>
<td>0.899</td>
</tr>
<tr>
<td>MAP</td>
<td>0.903</td>
</tr>
</tbody>
</table>

The best performance is in bold font. The number with * means it is significant higher than Eud at the 0.001 significance level.
work of nearest neighbor search) on SIFT1M and ImageNet databases.

For these two sets, there is no exact labels, so we follow the criterion used in many previous fast nearest neighbor search work [46]: the groundtruth neighbors are obtained by brute force search. We use the top-1 percent nearest neighbors as groundtruth. We record the computational time (offline training and online retrieval) and ranking performance in Tables 6 and 7. The offline time is for training and the online time is for a query retrieval (averaged). We randomly select 1,000 images from the database as out-of-sample queries and evaluate the performance.

For comparison, some state-of-the-art hashing methods including LSH, Spectral Hashing [46] and Spherical Hashing (a very recent proposed method [47]) are used. For EMR, we select 10% of the database samples to run k-means algorithm with maximum 5 iterations, which is very fast. In the online stage, the hamming distances between the query sample and the database samples are calculated for LSH, Spectral hashing and Spherical Hashing and then the distances are sorted. While for our method, we directly compute the scores via Eq.(17) and sort them. If we adopt any filtering strategy to reduce the number of candidate samples, the computational cost for each method would be reduced equally. So we only compare the largest computational cost (brute force search). We adopt 64-bit binary codes for SIFT1M and 128-bit for ImageNet for all the hash methods.

From Tables 6 and 7, we find that EMR has a comparable online query cost, and a high nearest neighbor search accuracy, especially on the high dimensional data set ImageNet, showing its good performance.

5.5 Algorithm Analysis

From the comprehensive experimental results above, we get a conclusion that our algorithm EMR is effective and efficient. It is appropriate for CBIR since it is friendly to new queries. A core point of the algorithm is the anchor points selection. Two issues should be further discussed: the quality and the number of anchors. Obviously, our goal is to select less anchors with higher quality. We discuss them as follows:

- How to select good anchor points? This is an open question. In our method, we use k-means clustering centers as anchors. So any faster or better clustering methods do help to the selection. There is a tradeoff between the selection speed and precision. However, the k-means centers are not perfect – some clusters are very close while some clusters are very small. There is still much space for improvement.

- How many anchor points we need? There is no standard answer but our experiments provide some clues: SIFT1M and ImageNet databases are larger than COREL, but they need similar number of anchors to receive acceptable results, i.e., the required number of anchors is not proportional to the database size. This is important, otherwise EMR is less useful. The number of anchors is determined by the intrinsic cluster structure.

6 CONCLUSION

In this paper, we propose the Efficient Manifold Ranking algorithm which extends the original manifold ranking to
handle large scale databases. EMR tries to address the shortcomings of original manifold ranking from two perspectives: the first is scalable graph construction; and the second is efficient computation, especially for out-of-sample retrieval. Experimental results demonstrate that EMR is feasible to large scale image retrieval systems – it significantly reduces the computational time.

ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under Grant 61125203, 91120302, 61173186, 61222207, and 61173185, and in part by the National Basic Research Program of China (973 Program) under Grant 2012CB316400, Fundamental Research Funds for the Central Universities, Program for New Century Excellent Talents in University (NCET-09-0685), Zhejiang Provincial Natural Science Foundation under Grant Y1101043 and Foundation of Zhejiang Provincial Educational Department under Grant Y201018240.

REFERENCES


Bin Xu received the B.S. degree in computer science from Zhejiang University, China, in 2009. He is currently a candidate for the Ph.D. degree in computer science at Zhejiang University. His current research interests include information retrieval, recommender system, and machine learning. He is a student member of the IEEE.

Jiajun Bu received the B.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1995 and 2000, respectively. He is a Professor in the College of Computer Science, Zhejiang University. His current research interests include embedded system, data mining, information retrieval, and mobile database. He is a member of the IEEE.

Chun Chen received the B.S. degree in mathematics from Xiamen University, China, in 1981, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1984 and 1990, respectively. He is a Professor in the College of Computer Science, Zhejiang University. His current research interests include information retrieval, data mining, computer vision, computer graphics, and embedded technology. He is a member of the IEEE.

Can Wang received the B.S. degree in economics, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1995, 2003, and 2009, respectively. He is currently a Faculty Member in the College of Computer Science at Zhejiang University. His current research interests include information retrieval, data mining, and machine learning. He is a member of the IEEE.

Deng Cai is a Professor in the State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, China. He received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, USA in 2009. Before that, he received the bachelor’s and master’s degrees from Tsinghua University, China, in 2000 and 2003, respectively, both in automation. His current research interests include machine learning, data mining and information retrieval. He is a member of the IEEE.

Xiaofei He received the B.S. degree in computer science from Zhejiang University, China, in 2000 and the Ph.D. degree in computer science from the University of Chicago, IL, USA, in 2005. He is a Professor in the State Key Lab of CAD&CG at Zhejiang University. Prior to joining Zhejiang University in 2007, he was a Research Scientist at Yahoo! Research Labs, Burbank, CA, USA. His current research interests include machine learning, information retrieval, and computer vision. He is a senior member of the IEEE.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.