Efficient Manifold Ranking for Image Retrieval

Bin Xu[1], Jiajun Bu[1], Chun Chen[1], Deng Cai[2], Xiaofei He[2], Wei Liu[3], Jiebo Luo[4]

[1]Zhejiang Provincial Key Laboratory of Service Robot
College of Computer Science, Zhejiang University, Hangzhou, China
{xbzju,bjj,chenc,hezhanying}@zju.edu.cn
[2]State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou, China
{dengcai,xiaofeihe}@cad.zju.edu.cn
[3]Columbia University, New York, NY, USA, wliu@ee.columbia.edu
[4]Kodak Research Laboratories, Eastman Kodak Company, Rochester, NY, USA, jiebo.luo@kodak.com

ABSTRACT

Manifold Ranking (MR), a graph-based ranking algorithm, has been widely applied in information retrieval and shown to have excellent performance and feasibility on a variety of data types. 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, both in graph construction and ranking computation stages, which significantly limits its applicability to very large data sets. In this paper, we extend the original manifold ranking algorithm and propose a new framework named Efficient Manifold Ranking (EMR). We aim to address the shortcomings of MR from two perspectives: scalable graph construction and efficient computation. Specifically, we build an anchor graph on the data set instead of the traditional k-nearest neighbor graph, and design a new form of adjacency matrix utilized to speed up the ranking computation. The experimental results on a real world image database demonstrate the effectiveness and efficiency of our proposed method. With a comparable performance to the original manifold ranking, our method significantly reduces the computational time, makes it a promising method to large scale real world retrieval problems.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; H.2.8 [**Database Applications**]: Image databases

General Terms

Algorithm, Performance

Keywords

Efficient manifold ranking, image retrieval, graph-based algorithm, out-of-sample

SIGIR'11 July 24–28, 2011, Beijing, China

1. INTRODUCTION

Traditional image retrieval systems are based on keyword search, such as Google and Yahoo image search. In these systems, user keyword 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 (CBIR) is a considerable choice to overcome these difficulties. CBIR has drawed a great attention in the past two decades [5, 19, 27]. Different from traditional keyword search systems, CBIR systems utilize the low-level features (color, texture, shape, etc.), automatically extracted from images. This is a vital character for efficient management and search in a large image database. But the low-level features used in CBIR systems are often visually characterized, and with no direct connection with semantic concepts of the images. How to narrow the semantic gap has been the main challenge for CBIR.

Many data sets have underlying cluster or manifold structure. Under such circumstances, the assumption of *label consistency* is reasonable [23, 36]. 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. Thus, a good CBIR method should consider low-level features as well as intrinsic structure of the data.

Manifold Ranking (MR) [36,37], a semi-supervised graph-based ranking algorithm, has been widely applied in information retrieval, and shown to have excellent performance and feasibility on a variety of data types, such as the text [28], image [9], and video [34]. The core idea of manifold ranking is to rank the data with respect to the intrinsic structure collectively revealed by a large number of data. By taking the underlying structure into account, manifold ranking assigns each data point a relative ranking score, instead of an absolute pairwise similarity as traditional ways. The score is treated as a distance metric defined on the manifold, which is more meaningful to capturing the semantic relevance degree. He et al. [9] firstly applied manifold ranking 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 data sets – it has expensive computational

cost, both in graph construction and ranking computation stages. Particularly, it is costly to handle an out-of-sample query (a new sample). 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. It's intolerable for a user to wait a long time to get returns.

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. The main contributions of this paper are as follows. (1) Based on the anchor graph construction [18,35], we design a new form of adjacency matrix and give it an intuitive explanation. (2) By the new form, our method EMR achieves a comparable performance to original manifold ranking but significantly reduces the computational time.

The rest of this paper is organized as follows. In section 2, we briefly discuss some related works and in section 3, we review the manifold ranking algorithm and make a detailed analysis. The proposed approach EMR is described in section 4. In section 5, we present the experiment results on a real world image database. Finally we provide an extension analysis in section 6 and conclusions in section 7.

2. RELATED WORK

In this section, we discuss two most relevant topics to our work: ranking model and content-based image retrieval.

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 [22]; or link structure based models, like the famous PageRank [2] and HITS [15]; or cross media models [13]. 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 [7, 26]. However, many conventional models ignore the important issue of efficiency, which is crucial for a real-time systems, such as a web application. In [29], the authors present a unified framework for jointly optimizing effectiveness and efficiency.

2.1 Graph-based Ranking

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 [2, 15] and social networks research [3, 8, 17]. Generally, a graph 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 in a graph, based on local or global information draw from the graph.

Agarwal [1] proposed to model the data by a weighted graph, and incorporated this graph structure into the ranking function as a regularizer. Guan et al. [8] proposed a graph-based ranking algorithm for interrelated multi-type resources to generate personalized tag recommendation. Liu et al. [17] proposed an automatically tag ranking scheme by performing a random walk over a tag similarity graph. In [3], the authors made the music recommendation by ranking on a unified hypergraph, combining with rich social information and music content. Recently, there have been some papers on speeding up manifold ranking. In [11], the authors partitioned the data into several parts and computed the ranking function by a block-wise way.

2.2 Content-based Image Retrieval

In many cases, we have no more information than the data itself. Without label information, capturing semantic relationship between images is quite difficult. A great amount of researches have been performed for designing more informative low-level features (e.g., SIFT features [20]) to represent images, or better metrics (e.g., DPF [16]) to measure the perceptual similarity, but their performance is restricted by many conditions and is sensitive to the data. Relevance feedback [24] is a powerful tool for interactive CBIR. User's high level perception is captured by dynamically updated weights based on the user's feedback.

Many traditional image retrieval methods focus on the data features too much, and they ignore the underlying structure information, which is of great importance for semantic discovery, especially when the label information is unknown. A good CBIR method should consider the image features as well as the intrinsic structure of the data, in our opinion. Manifold ranking [36,37] ranks data with respect to the intrinsic geometrical structure, which is exactly in line with our consideration.

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 $\chi = \{x_1, x_2, \ldots, x_n\} \subset R^m$ 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 $w_{ij} = \exp[-d^2(x_i, x_j)/2\sigma^2)]$ 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 χ, such as the Euclidean distance. Let $r : \chi \to R$ be a ranking function which assigns to each point x_i a ranking score r_i. Finally, we define an initial vector $y = [y1, \ldots, y_n]^T$, 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

$$O(r) = \frac{1}{2}\left(\sum_{i,j=1}^{n} w_{ij}\left\|\frac{1}{\sqrt{D_{ii}}}r_i - \frac{1}{\sqrt{D_{jj}}}r_j\right\|^2 + \mu\sum_{i=1}^{n}\|r_i - y_i\|^2\right), \tag{1}$$

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 have close ranking scores. The second term is a fitting constraint, which means the ranking result should fit to the initial label assignment. If we have more prior knowledge about the relevance or confidence of each query, we can assign different initial scores to the queries. Minimizing the cost function

$O(r)$, we get the optimal r by the following closed form

$$r^* = (I_n - \alpha S)^{-1} y, \tag{2}$$

where $\alpha = \frac{1}{1+\mu}$, I_n is an identity matrix with $n \times n$, and S is the symmetrical normalization of W, $S = D^{-1/2} W D^{-1/2}$. In large scale problems, we prefer to use the iteration scheme:

$$r(t+1) = \alpha S r(t) + (1-\alpha) y. \tag{3}$$

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 and proved effective for multiple resources, it has its own drawbacks to handle large scale data sets, 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 $O(kn^2)$, 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. In some cases, it is impossible to keep matrix W as large as $n \times n$ in memory, especially for very large data sets or memory-short environment applications. 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 data set.

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 try to address the shortcomings of original manifold ranking from two main perspectives: scalable graph construction and efficient ranking computation. Particularly, our method can handle the out-of-sample retrieval problem, which is crucial for a real world retrieval system.

4.1 Scalable Graph Construction

To handle scalable data sets, we want the graph construction cost to be linear or near linear with the graph size. That means, for each data point, we can't search the whole graph, like kNN strategy does. To achieve this requirement, we construct an anchor graph [18, 35] 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 [33], 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. [18] 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 [18,35]. 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 $f : \chi \to R$ be a real value function which assigns each data point in χ 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 χ and anchors in U. Then we estimate $f(x)$ for each data point as a weighted average of the labels on anchors

$$\widehat{f}(x_i) = \sum_{k=1}^{d} z_{ki} f(u_k), i = 1, \ldots, n, \tag{4}$$

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. We adopt the well known Nadaraya-Watson kernel regression to assign weights smoothly

$$z_{ki} = \frac{K(\frac{|x_i - u_k|}{\lambda})}{\sum_{l=1}^{d} K(\frac{|x_i - u_l|}{\lambda})}, \tag{5}$$

with the *Epanechnikov* quadratic kernel

$$K_\lambda(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \leq 1; \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The smoothing parameter λ 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 λ. For example, it can be a constant selected by cross-validation from a set of training data. In this paper we use a more robust way to get λ, which uses the nearest neighborhood size s to replace λ, that is

$$\lambda(x_i) = |x_i - u_{[s]}|, \tag{7}$$

where $u_{[s]}$ is the sth closest anchor of x_i.

Specifically, to build the anchor graph, we connect each data point to its s nearest anchors and then assign weights to each connection by the kernel function. So the construction has a total computational complexity $O(sdn)$, where d is the number of anchors. Thus, the number of anchors determines the efficiency of the anchor graph construction. If $d \ll n$, the construction is very fast.

How can we get the anchors? Active learning [25, 32] or clustering methods are considerable choices. For example, we use k-means algorithm and select the centers as anchors. Some fast k-means algorithms [14] can speed up the computation. Random selection is a competitive method which has extremely low selection cost and acceptable performance. Later in our experiments, we compare the performance of using random anchors, fast k-means anchors and normal k-means anchors.

The main feature, also the main advantage of building an anchor graph is separating the graph construction into

two stages – an off-line anchor selection stage and an on-line graph construction stage. That means, we can adopt any useful method to select or build the anchors with little concern for their time complexity, while the graph construction is always efficient since it has linear complexity to the date size. Note that we don't have to update the anchors frequently, because informative anchors for a large data set 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 $Z \in R^{d\times n}$ can be seen as a d dimensional representation of the data $X \in R^{m\times n}$, 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

$$W = Z^T Z, \tag{8}$$

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 [18].

In large scale cases, it is expensive to keep the matrix W as large as $n \times n$ in memory, while in our construction, we just need to save the $d \times n$ matrix Z. If $d \ll n$, we can dramatically reduce the memory cost.

4.2 Efficient 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 $H = ZD^{-\frac{1}{2}}$, and $S = H^T H$, then the final ranking function r can be directly computed by

$$r^* = (I_n - \alpha H^T H)^{-1} y = (I_n - H^T (HH^T - \frac{1}{\alpha} I_d)^{-1} H) y. \tag{9}$$

Proof. Just check that $(I_n - \alpha H^T H)$ times $(I_n - H^T (HH^T - \frac{1}{\alpha} I_d)^{-1} H)$ gives the identity matrix:

$$\begin{aligned}
&(I_n - \alpha H^T H)(I_n - H^T (HH^T - \tfrac{1}{\alpha} I_d)^{-1} H) \\
&= I_n - \alpha H^T H - (H^T - \alpha H^T H H^T)(HH^T - \tfrac{1}{\alpha} I_d)^{-1} H \\
&= I_n - \alpha H^T H + \alpha H^T (-\tfrac{1}{\alpha} I_d + HH^T)(HH^T - \tfrac{1}{\alpha} I_d)^{-1} H \\
&= I_n - \alpha H^T H + \alpha H^T H \\
&= I_n
\end{aligned}$$

□

By equation (9), 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 (9), D is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$. When $W = Z^T Z$,

$$D_{ii} = \sum_{j=1}^{n} z_i^T z_j = z_i^T v, \tag{10}$$

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

A useful trick for computing r^* in equation (9) 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 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 by kernel regression 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 (9). Images with highest ranking scores are considered as the most relevant and return to the user.

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 a real world image database. All algorithms in our experiments are implemented in MATLAB 9.0 and run on a computer with 2.0 GHZ(×2) CPU, 8GB RAM.

5.1 Experiments Setup

The image database consisting of 7,700 images from COREL image database. COREL is widely used in many CBIR works [12, 19, 31]. All of the images are from 77 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. In Figure 1, we randomly select and show nine image samples from three different categories.

Figure 1: COREL image samples randomly selected from semantic concept beach, bird and elephant.

5.1.1 Evaluation Metric Discussion

There are many measures to evaluate the retrieval results such as *precision*, *recall*, *F measure*, *MAP* and *NDCG* [21]. But for a real CBIR application, especially for a web application, not all the measures are meaningful.

Generally, the image retrieval engine present at most 20 images in a screen without scrolling. 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. Among evaluation measures, MAP has been shown to have especially good discrimination 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 [21]. 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

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk}). \qquad (11)$$

NDCG is designed for situations of non-binary notions of relevance. It is not suitable here.

5.1.2 Baseline Algorithm

To show EMR's effectiveness, several methods are compared: MR, FMR [11] and SVM for image retrieval (baseline) [25]. Given that the goal of our paper is to improve MR, so the main comparative method is the original MR.

FMR 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.

SVM can be easily transformed to a query-based ranking algorithm, and has been successfully applied in image retrieval. With the specified relevant/irrelevant (to the query) information, a maximal margin hyperplane is build to separate the relevant from irrelevant images, and then the most relevant images are the ones farthest from the SVM boundary. We use the well known LIBSVM toolbox [4] and select the RBF kernel. The parameters C and g in LIBSVM are 50 and 0.5 respectively.

5.2 Implementation Issues

Here we discuss some important implementation issues for our experiments: low-level features of images and out-of-sample retrieval.

5.2.1 Low-Level Features of Images

Low-level features representation of images is crucial for CBIR. If the low-level features can accurately measure the semantic distance between images, there is no need to design any new retrieval method. Unfortunately, low-level features always fail to capture the high-level semantic concepts. In our opinion, more than the low-level features, a good CBIR method should take the underlying structure of the data into account. In our experiments, we use a 64-dimensional color histogram and a 64-dimensional Color Texture Moment [12, 31] to represent the images.

5.2.2 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. In [10], 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. Moreover, their method may change the query's initial semantic meaning. In contrast, we efficiently update the graph structure and use the new sample as a query for retrieval. The image database can be dynamically updated.

For one instant retrieval, it is unwise to update the whole graph or rebuild the anchors, especially on a large data set. We can adopt fast updating strategies as follows.

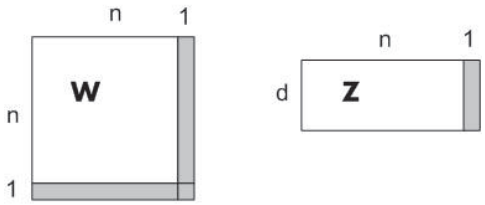


Figure 2: Extend matrix W (MR and FMR) and Z (EMR) in the gray regions for an out-of-sample.

A fast but *inexact* strategy for MR and FMR is leaving the original graph unchanged and adding a new row and a new column to W. But k nearest neighbor relationship is not symmetric, therefore the query's neighbors would lose some local information – its original neighbors' weights decrease relatively and they may have $k+1$ nearest neighbors. While for EMR, each data point is independently computed. We just assign weights between the new query and its nearby anchors. That forms a new column of Z. One point has little effect to the stable anchors in a large data set (e.g., cluster centers). Figure 2 shows the updating strategies.

5.3 Performance

We firstly compare our proposed method EMR with MR, FMR and baseline on COREL database 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. For our method EMR, 1000 k-means anchors are used. Later in the model selection part, we find that using 800 or fewer anchors achieves a close 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.

At the beginning of the retrieval, we don't have the user specified relevant/irrelevant information. To apply SVM, the pseudo relevance feedback [21] strategy, also known as blind relevance feedback is adopted. We rank the images in the database according to their euclidean distances to the query, and then the nearest ten images are considered as relevant and the farthest ten are irrelevant. With such an approach, we run SVM as normal at the beginning, and the user gets images without any additional interaction.

5.3.1 *Average Accuracy*

Each image is used as a query and the retrieval performance is averaged. Figure 3 prints the average precision (at 10 to 80) of each method and Table 1 records the average values of recall, F1 score (at 10 to 80), and the values of MAP with all relevant images. It is easy to find that the performance of MR and EMR are very close, while FMR lose a little precision. All the three algorithms are better than the baseline method in the whole range.

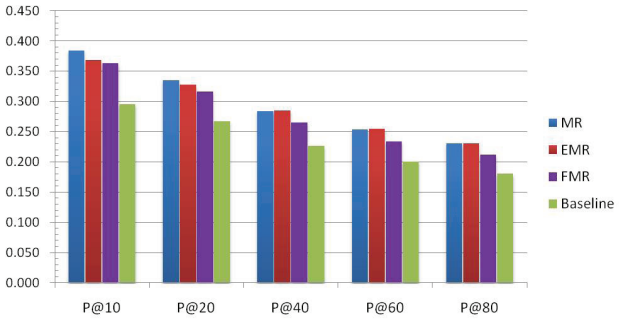


Figure 3: Retrieval precision at top 10 to 80 returns of MR (left), EMR, FMR and Baseline.

Table 1: The average values of Recall, F1 on top 10 to 80, and MAP.

	MR	EMR	FMR	SVM
R@10	0.039	0.037	0.037	0.030
F1@10	0.071	0.068	0.067	0.054
R@20	0.068	0.066	0.064	0.054
F1@20	0.113	0.110	0.106	0.090
R@40	0.115	0.115	0.107	0.092
F1@40	0.163	0.164	0.153	0.130
R@60	0.154	0.155	0.142	0.122
F1@60	0.191	0.193	0.177	0.152
R@80	0.186	0.187	0.172	0.146
F1@80	0.206	0.207	0.190	0.161
MAP	0.190	0.191	0.174	0.143

The anchor points are computed off-line and do not affect the current on-line retrieval system. To speed up the selection, we can use fast k-means strategy or just select random anchors. The fast k-means strategy we used is very simple and effective: setting the number of maximum iterations of k-means to 10 (default 100). Experiments results in Figure 4 show that the performance of fast k-means or random anchors is close to normal k-means anchors. Using random anchors dramatically decreases the anchor points selection cost when the data size is extremely large. Thus, cost and performance are trade-offs in many situations.

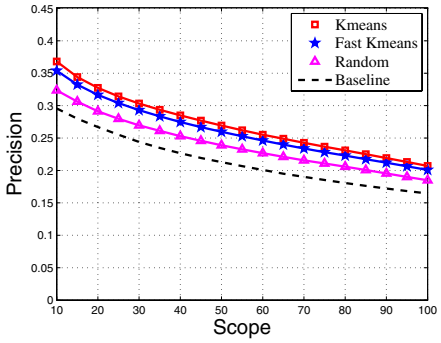


Figure 4: Performance of EMR with random anchors vs. k-means (fast and normal) anchors. Note that using random anchors achieves a close performance to k-means anchors.

To see the performance distribution in the whole data set more concretely, we plot the retrieval precision at top 10 returns for all 77 categories in Figure 5 without relevance feedback. 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 category, and on some categories, EMR is better than MR. But for FMR, the distribution is totally different.

5.3.2 *Response Time for Out-of-Sample*

In this section, we compare the response time of MR, FMR and EMR when handling a new sample. We use a serial number of COREL samples, from 1,500 to 7,500, to build the graph and test 100 out-of-samples. The response time (in seconds) for each sample is averaged and showed in Table 2. To test EMR's efficiency in very large data sets,

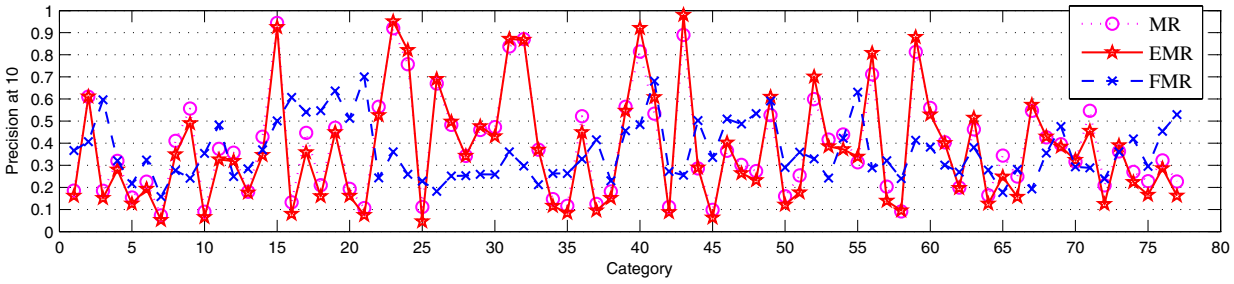


Figure 5: Precision at the top 10 returns of the three algorithms on each category.

we use 50,000 to 200,000 synthetic data in 128 dimensions. Since EMR has a closed form solution, so the response time is reliable and comparable – no matter the data is real or synthetic.

Table 2: Response time in seconds, MR (closed form in MR_c and iteration scheme in MR_{it}), FMR, and EMR (1000 anchors in EMR_{1k} and 800 anchors in EMR_{800}) with different graph size. The last three lines use synthetic data.

Size	MR_c	MR_{it}	FMR	EMR_{1k}	EMR_{800}
1500	1.283	1.685	0.821	0.462	0.264
2500	5.416	5.463	3.014	0.517	0.308
3500	14.233	12.437	7.507	0.576	0.358
4500	29.712	23.355	15.169	0.638	0.402
5500	53.854	39.559	26.866	0.710	0.447
6500	85.319	59.296	42.944	0.769	0.495
7500	129.715	85.916	65.124	**0.833**	**0.542**
50k	-	-	-	**3.869**	**2.700**
100k	-	-	-	**7.375**	**5.201**
200k	-	-	-	**14.534**	**10.183**

When the data size is large, it is extremely costly to save matrix $W \in R^{n \times n}$ in memory for MR and FMR. Even if we could save W in memory, the computational time is too long. But for EMR, we just save the smaller matrix $Z \in R^{d \times n}$, and the time is acceptable.

With different graph size, we fix the number of anchors to 1000 and 800 for convenience. Actually we could use fewer anchors to achieve a good performance in many cases. For MR, we use both the closed form and the iteration scheme. We record the response time of MR by closed form in MR_c and by iteration scheme in MR_{it}. The stop condition for MR_{it} is defined as $\|r^{t+1} - r^t\| < \epsilon$, where t is the iteration step and ϵ is a threshold. We use a very loose threshold 10^{-4}. With smaller threshold, MR_{it} needs more time to converge. We find that, the computational time of MR or FMR increases very fast, while our method increases slowly. Note that the time of extracting low-level features from the query image is not included in the response time, since it is not the focus of our ranking method.

5.3.3 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. For each single value, EMR is run independently 10 times and the results are averaged.

There are three parameters in our method EMR: s, α, and N. Parameter s is the neighborhood size in the anchor graph. Small value of s makes the weight matrix Z very sparse. Parameter α is the tradeoff parameter in EMR and MR. Parameter N is the number of anchor points. For convenience, the parameter α is fixed at 0.99, consistent with the experiments performed in [9, 36, 37]

Figure 6 shows the performance of EMR (Precision at 10) 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 $s = 5$.

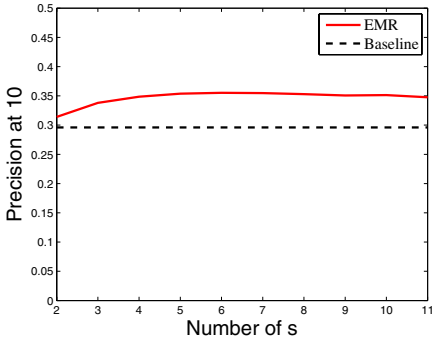
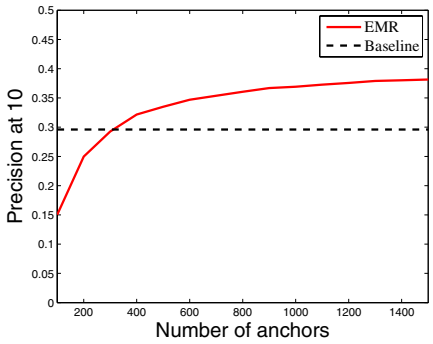
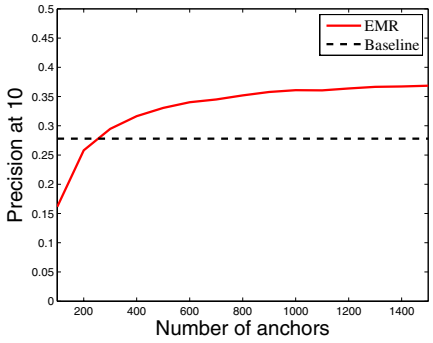


Figure 6: Retrieval precision versus different values of parameter s. The dotted line represents the baseline.

Figure 7(a) shows the performance of EMR (Precision at 10) versus different number of anchors in the whole data set. From the figure, we find that the performance increases very slowly when the number of anchors is larger than 400 (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 and out-of-sample retrieval cost will be further reduced. To further investigate how many anchors are required, we evaluate EMR on half of the data set (77 categories, 50 images per category, 3850 images in total) in Figure 7(b). The number of categories is the same and the data size is reduced by half. However



(a) 7700 images



(b) 3850 images

Figure 7: Retrieval precision versus different number of anchor points in (a) 7700 images and (b) 3850 images. The dotted lines represent baseline performance in each case.

the result is similar – the performance increases very slowly when the number of anchors is larger than 400. This demonstrates that the number of required anchors is affected by the underlying structure, but not easily proportional to the size of the data set.

5.3.4 *Performance with Relevance Feedback*

Relevance Feedback [24] 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 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 two rounds, since the users have no patience to do more. Images have been labeled in the first round are excluded in the second round. The retrieval performance are plotted in Figure 8. To avoid changing the original query's semantic meaning, the initial scores of the query and the feedbacks are 10 and ± 1 respectively.

By relevance feedback, original MR receives higher improvement than any other methods. But, relevance feedback is designed for real interactive systems in which the data size is always large. Original MR is too slow for these systems. See Table 2, when the data size is 7500, for one out-of-sample retrieval, MR_c needs more than 100 seconds. The meaning of applying relevance feedback to original MR is limited. Our method is a little worse than MR, but it is much faster than MR and better than the rest two. We leave the question of how to make further efforts to EMR's performance by relevance feedback in our future works. In this paper, our focus is to speed up manifold ranking, but not the relevance feedback.

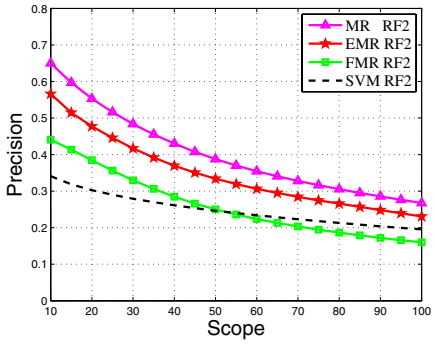


Figure 8: Average precision-scope curves for each methods after two relevance feedbacks.

5.3.5 *Case Study*

We show two real image retrieval cases in this part to see the performance of EMR intuitively. The first case in Figure 9 succeeds to retrieve relevant images by EMR without relevance feedback. The second case in Figure 10(a) is failed to capture the query's concept at first, but improved greatly after two feedback rounds in Figure 10(b).

An interesting thing is that the distance metric defined by EMR (we name it manifold distance) is very different with traditional metrics (e.g., Euclidean distance) used in many other retrieval methods. Euclidean distance only considers the data similarity, but manifold distance tries to capture the semantic relevance by the underlying structure of the data set. For example, the first retrieved image in the first case is the 34th image retrieved by Euclidean distance.

In the second case, the query is confused with many other concepts, including the oil painting and the elephant. The similar color and texture features make images in those concepts receiving higher ranking scores. At the same time, this case indicates that EMR is confused by the low-level features and misses the dominant concept *antelope* of the query, when there is no manual help. Fortunately, EMR is easy to integrate the label information – after two relevance feedbacks, the concept is captured successfully by EMR.

6. EXTENSION

Many real world applications have highly complex graph structure, like a social network or a web link graph. In such circumstances, the graph is existing and has very large size. If we have rich information about each node, we use EMR for efficient retrieval. But if we have no more information except the graph, how can we accelerate manifold ranking?

Figure 9: A successful content-based image retrieval case by EMR without relevance feedback. The first image with red box is the query, and the rest are the top 10 returns.

(a) Top 10 returns without relevance feedback. The image with green box is the only one correct.

(b) Top 10 returns after two feedback rounds. All images belong to the same category.

Figure 10: A failed case by EMR without relevance feedback, but improved greatly by two feedback rounds. The first image with red box is the query.

Adjacency matrix W is symmetric. Its SVD decomposition is $W = U\Sigma U^T$, where $U \in R^{n\times k}$ and $\Sigma \in R^{k\times k}$. Let $H = D^{-\frac{1}{2}}U$, we can rewrite equation (2) to be

$$r^* = (I_n - H(H^TH - \frac{1}{\alpha}\Sigma^{-1})^{-1}H^T)y, \qquad (12)$$

which is very efficient if $k \ll n$. But computing SVD of matrix W requires $O(n^3)$ operations, which is the same with the original matrix inversion. If we can fast decompose W, we save the computational time. Fortunately, W is highly sparse and may have a very low rank. Thus, some fast low-rank matrix approximation methods may help. We can arrange matrix W to be

$$W = \begin{pmatrix} G & W_{21}^T \\ W_{21} & W_{22} \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} G \\ W_{21} \end{pmatrix}$$

where G is a $l \times l$ symmetric matrix from W, $l \ll n$. The Nyström method [30] computes SVD of G ($G = U_G\Sigma_G U_G^T$) and approximates singular values and singular vectors of W by $\Sigma = (\frac{n}{l})\Sigma_G$ and $U = \sqrt{\frac{l}{n}}CU_G\Sigma_G^{-1}$. When k singular vectors are used, the cost of this algorithm is $O(l^3 + nlk)$. Another alternative method is the Column-sampling method [6]. It approximates the spectral decomposition of W by using the SVD of C directly ($C = U_C\Sigma_C V_C^T$). That is, $\Sigma = \sqrt{\frac{n}{l}}\Sigma_C$ and $U = U_C$. Its cost is $O(nl^2)$. These two methods both can be used to speed up the computation.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we propose the Efficient Manifold Ranking algorithm which extends the original manifold ranking to handle scalable data sets. We apply EMR to a content-based image retrieval application based on a real world image database. 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 real world image retrieval systems – it significantly reduces the computational time, as well as the storage space. Actually, our new design of the adjacency matrix can be used to many other graph-based algorithms, and EMR is also feasible to other types of information resources.

In our future work, we will test more visual features and evaluate our method on other databases. Moreover, many social network sites like Flickr allow users to annotate images with free tags, which significantly help us to understand semantic concepts of images. Thus, for images having tag information, we can combine the image features and tag information to improve retrieval performance. Another extension of our work is the distributed computation, an enterprise solution for web scale retrieval systems.

8. ACKNOWLEDGEMENT

This work is supported by Program for New Century Excellent Talents in University (NCET-09-0685).

9. REFERENCES

[1] S. Agarwal. Ranking on graph data. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 25–32, 2006.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[3] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the 18th annual ACM international conference on Multimedia*, pages 391–400, 2010.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

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

[6] A. Frieze, R. Kannan, and S. Vempala. Fast Monte Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[7] W. Gao, P. Cai, K. Wong, and A. Zhou. Learning to rank only using training data from related domain. In

Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pages 162–169. ACM, 2010.

[8] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 540–547, 2009.

[9] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 9–16, 2004.

[10] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Generalized manifold-ranking-based image retrieval. *IEEE Transactions on Image Processing*, 15(10):3170–3177, 2006.

[11] R. He, Y. Zhu, and W. Zhan. Fast Manifold-Ranking for Content-Based Image Retrieval. In *ISECS International Colloquium on Computing, Communication, Control, and Management*.

[12] X. He, D. Cai, and J. Han. Learning a maximum margin subspace for image retrieval. *IEEE Transactions on Knowledge and Data Engineering*, pages 189–201, 2007.

[13] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126, 2003.

[14] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[15] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[16] B. Li, E. Chang, and C. Wu. DPF-a perceptual distance function for image retrieval. In *International Conference on Image Processing*, volume 2, pages 597–600, 2002.

[17] D. Liu, X. Hua, L. Yang, M. Wang, and H. Zhang. Tag ranking. In *Proceedings of the 18th international conference on World wide web*, pages 351–360, 2009.

[18] W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 679–686, 2010.

[19] Y. Liu, D. Zhang, G. Lu, and W. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[20] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, page 1150, 1999.

[21] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[22] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

[23] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

[24] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644–655, 2002.

[25] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the 9th ACM international conference on Multimedia*, pages 107–118, 2001.

[26] M. Tsai, T. Liu, T. Qin, H. Chen, and W. Ma. FRank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390. ACM, 2007.

[27] R. Veltkamp and M. Tanase. Content-Based Image Retrieval Systems: A Survey. 2002.

[28] X. Wan, J. Yang, and J. Xiao. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2903–2908, 2007.

[29] L. Wang, J. Lin, and D. Metzler. Learning to efficiently rank. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145, 2010.

[30] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2001.

[31] H. Yu, M. Li, H. Zhang, and J. Feng. Color texture moments for content-based image retrieval. In *International Conference on Image Processing*, volume 3, pages 929–932, 2002.

[32] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd international conference on Machine learning*, pages 1081–1088, 2006.

[33] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *In Advances in Neural Information Processing Systems*, 2009.

[34] X. Yuan, X. Hua, M. Wang, and X. Wu. Manifold-ranking based video concept detection on large database and feature pool. In *Proceedings of the 14th annual ACM International Conference on Multimedia*, pages 623–626, 2006.

[35] K. Zhang, J. Kwok, and B. Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240, 2009.

[36] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 595–602, 2004.

[37] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems*, page 169, 2004.