Nearest Neighbor Search for Relevance Feedback

Jelena Tešić and B.S. Manjunath
Electrical and Computer Engineering Department
University of California, Santa Barbara, CA 93106-9560
$\{$*jelena, manj*$\}$*@ece.ucsb.edu*

Abstract

We introduce the problem of repetitive nearest neighbor search in relevance feedback and propose an efficient search scheme for high dimensional feature spaces. Relevance feedback learning is a popular scheme used in content based image and video retrieval to support high-level concept queries. This paper addresses those scenarios in which a similarity or distance matrix is updated during each iteration of the relevance feedback search and a new set of nearest neighbors is computed. This repetitive nearest neighbor computation in high dimensional feature spaces is expensive, particularly when the number of items in the data set is large. In this context, we suggest a search algorithm that supports relevance feedback for the general quadratic distance metric. The scheme exploits correlations between two consecutive nearest neighbor sets thus significantly reducing the overall search complexity. Detailed experimental results are provided using 60 dimensional texture feature dataset.

1. Introduction

The basic idea behind this paper is to constrain the search space for the nearest neighbors for the next iteration using the current set of nearest neighbors. This paper expands and generalizes our earlier work on this problem [1] to include similarity matrices that have off diagonal elements and to provides detailed experimental results on real image datasets. We believe that this is the first time that this nearest neighbor update issue has been addressed in the content-based search and retrieval literature.

In content based retrieval, to retrieve images that are similar in texture or color, usually one computes the nearest neighbors of the query feature vector in the corresponding feature space. Given a collection of image or video data, the first step in creating a database is to compute relevant feature descriptors to represent the content. While the low level features are quite effective in similarity retrieval, they do not capture well the high level semantics. In this context, relevance feedback was introduced to facilitate interactive learning for refining the retrieval results [2, 3]. User identifies a set of retrieval examples relevant to the image, and that information is then used to compute a new set of retrievals.

A way to compute the new set of retrievals closer to the user's expectation is to modify the similarity metric used in computing the distances between the query and the database items. The distance between two feature vectors is typically calculated as a quadratic distance of the form $(Q - F)^T W(Q - F)$ where Q is a query vector, F is a database feature vector, and W is a positive semi-definite matrix [4]. During each iteration, the weight matrix is updated based on a user's feedback [5, 6]. Given the updated weight matrix, the next set of nearest neighbors is then computed. There are more recent methods, such as kernel-based ones that appear to be more effective in learning but computationally prohibitive for large datasets in high dimensions.

Nearest neighbor computations over a large number of dataset items is expensive. This is further aggravated by the fact that image descriptors are in very high dimensions, and the need to perform this search repetitively in relevance feedback. This can be a limiting factor on the overall effectiveness of using relevance feedback for similarity retrieval. Recently, there has been much work on indexing structures to support high-dimensional feature spaces. However, as reported in [7], the effectiveness of many of these indexing structures is highly data dependent and, in general, difficult to predict. Often a simple linear scan of all the items in the database is cheaper than using an index based search in high dimensions. An alternative to high-dimensional index structures is a compressed representation of the database items, like vector approximation structure known as VA-File [8].

In the VA-file architecture the feature space is quantized and each feature vector in the database is encoded with its compressed representation. Approximate sequential search over quantized candidates reduces the search complexity by allowing only a fraction of the actual feature vectors to be accessed.

In this paper we address the problem of nearest neighbor search for relevance feedback applications. Our method takes advantage of the correlation between two sets of nearest neighbors from the consecutive iterations in pruning the search space. The details are presented in the following sections.

2. Vector Approximation (VA) Files

The main idea of VA-File is to first construct a compressed representation of the feature vector. The compression of the feature space enables more items to be loaded into the main memory for fast access. For example, the current MPEG-7 standard uses a 62 dimensional texture descriptor and an adaptive color histogram descriptor that can have up to 256 dimensions [9]. In content based retrieval, to retrieve images that are similar in texture or color, usually one computes the nearest neighbors of the query feature vector in the corresponding feature space.

For multimedia databases, compression based methods have certain advantages. First, the construction of the approximation is based on the feature values in each of the dimensions independently. As a result, the compressed domain representation can support query search if the distance is quadratic. Secondly, the construction of approximation can be made adaptive to the dimensionality of data.

2.1. Construction of Approximations

Consider a database $\Phi = \{F_i \mid i \in [1, N]\}$ of N elements, where F_i is an M-dimensional feature vector.

$$F_i = [f_{i1}, f_{i2}, \ldots, f_{iM}]^T \qquad (1)$$

Let Q be the query object from a database Φ, $Q \in \Phi$, $Q = [q_1, q_2, \ldots, q_M]^T$. If W is a symmetric, real, positive definite matrix, define the quadratic distance metric d between a database object F_i and a query Q as :

$$d^2(Q, F_i, W) = (Q - F_i)^T W (Q - F_i) \qquad (2)$$

Each of the feature vector dimensions is partitioned into non overlapping segments. Generally, the number of segments is 2^{B_j}, $j \in [, M]$. B_j is the number of bits allocated to dimension j. Total number of bits allocated for each high-dimensional cell is $\sum_{j=1}^{M} B_j$. Denote the boundary points that determine the 2^{B_j} partitions to be $b_{lj},\ l = 0, 1, 2, \ldots, (2^{B_j} - 1)$.

For a feature vector F_i, its approximation $C(F_i)$ is an index to the cell containing F_i. If F_i is in partition l along the j^{th} dimension, then: $b_{lj} \le f_{ij} \le b_{l+1j}$. Consider Figure 1(a) where $B_1 = B_2 = 2$. The approximation cell for A is $C(A)$. $C(A)$ has the approximation "1100", where "11" and "00" are the indices on dimension d_1 and d_2 respectively. Note that each approximation cell contains a number of feature vectors. For example, $C(D)$ and $C(E)$ have the same approximation "1110".

2.2. Nearest Neighbor (NN) Search

Approximation based nearest neighbor search can be considered as a two phase filtering process [8]. Phase I is an approximation level filtering. In this phase, the set of all vector approximations is scanned sequentially and lower and upper bounds on the distances of each object in the database to the query object are computed. During the scan, a buffer is used to keep track of the K^{th} largest upper bound ρ found from the scanned approximations. If an approximation is encountered such that its lower bound is larger than the K^{th} largest upper bound found so far, the corresponding feature vector can be skipped since at least better candidates exist. Otherwise, the approximation will be selected as a candidate and its upper bound will be used to update the buffer. The resulting set of candidate objects at this stage is $N_1(Q, W)$.

Phase II finds K nearest neighbors from the feature vectors contained by approximations filtered in Phase I. The feature vectors are visited in increasing order of their lower bounds and the exact distances to the query vector are computed. If a lower bound is reached that is larger than the K^{th} actual nearest neighbor distance encountered so far, there is no need to visit the remaining candidates. Finally, the nearest neighbors are found by sorting the distances.

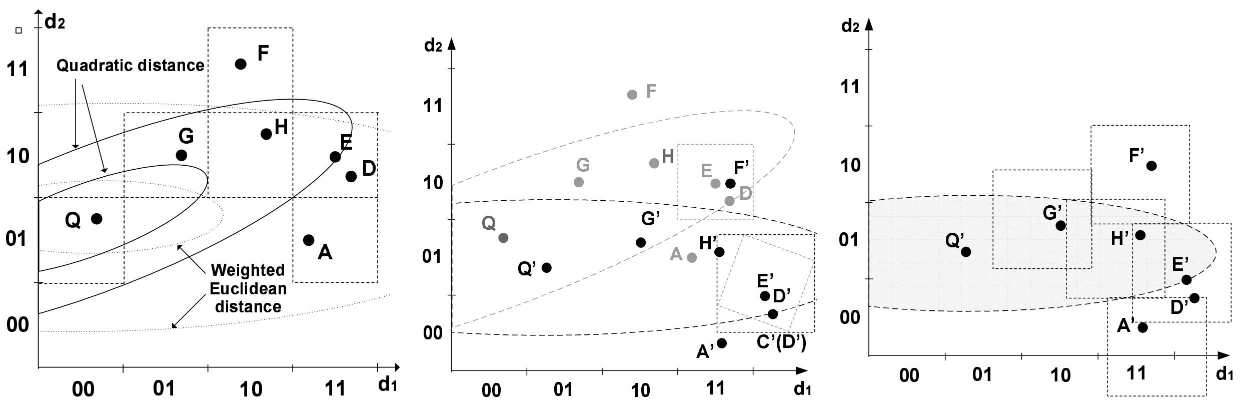
In database searches, the disk/page access is an expensive process. The number of candidates from Phase I filtering determines the cost of disk access/page access. Our focus is on improving Phase I filtering in the presence of relevance feedback.

3. Overview of Relevance Feedback

In the context of content-based image retrieval relevance feedback has attracted considerable attention. Given a set of retrievals for an image query, the user may identify some relevant and some non-relevant examples. Based on this examples, the similarity metric is modified to recompute the next set of retrievals. The hope is that this modification to the similarity metric would help provide better matches to a given query and meet the user's expectations.

Let W_t be the weight matrix used in iteration t, and R_t be the set of K nearest neighbors to the query object Q, using (2) to compute the distances. At iteration t, define the k^{th} positive example vector ($k = \overline{1, K'}$) as: $X_k^{(t)} = [x_{k1}^{(t)}, x_{k2}^{(t)}, \ldots, x_{kM}^{(t)}]^T,\ X_k^{(t)} \in R_t$. K' is the number of relevant objects identified by the user. These K' examples are used to modify the weight matrix W_t to W_{t+1}. We consider an optimized learning technique that merges two existing well known updating schemes:

MARS [6] restricts W_t to be a diagonal matrix. The weight matrix is modified using standard deviation σ_m of $x_{km}^{(t)}$ ($k \in [1, K'],\ m = \overline{1, M}$).



(a) VA Construction (b) Rotational Mapping (c) Search Space

Figure 1: a) Construction of 2D VA-file approximations and bound computation for (1) weighted Euclidean and (2)quadratic distance; b) Rotational mapping of feature space: $A \rightarrow A'$:$A' = PA$; c) Illustration of using r^u_{t+1} to limit the search space in Phase I adaptive filtering

The weight matrix is normalized after every update and the result is:

$$(W_{t+1})_m = \frac{(\prod_{i=1}^{M} \sigma_i^2)^{\frac{1}{M}}}{\sigma_m^2}$$

MindReader [5] updates the full weight distance matrix W_t, by minimizing the distances between the query and all positive feedback examples. In this scheme, user picks K' positive examples, and assigns a degree of relevance $\pi_k^{(t)}$ to an k^{th} positive examples $X_k^{(t)}$. Optimal solution for W_t is equivalent to Mahalanobis distance, if we assume the Gaussianity of positive examples:

$$W_{t+1} = \mid C_t \mid^{\frac{1}{M}} (C_t)^{-1} \quad (3)$$

The elements of the covariance matrix C_t are defined as:

$$(C_t)_{ij} = \frac{\sum_{k=1}^{K'} \pi_k^{(t)} (x_{ki}^{(t)} - q_i)(x_{kj}^{(t)} - q_j)}{\sum_{k=1}^{K'} \pi_k^{(t)}} \quad (4)$$

For $K' > M$ matrices C_t and W_t are symmetric, real and positive definite. C_t can be factorized as: $C_t = (P_t')^T \Lambda_t' P_t'$, $P_t'^T P_t' = I, \Lambda_t' = diag(\lambda_1', ..., \lambda_M')$. W_t can be factorized in the same manner, i.e.: $W_t = P_t^T \Lambda_t P_t$, $P_t^T P_t = I, \Lambda_t = diag(\lambda_1, ..., \lambda_M)$. From Eq. (3), it follows that $P_t = P_t'$ and

$$\lambda_i = \frac{(\prod_{i=1}^{M} \lambda_i')^{\frac{1}{M}}}{\lambda_i'}$$

Full matrix update approach better captures the dependencies among feature dimensions, thus reducing redundancies in high dimensional feature space and allowing us to filter out more false candidates. The downside of this full matrix update approach is that inverse covariance matrix $(C_t)^{-1}$ exists only if number of positive examples K' is larger or equal to the number of feature dimension M. If $K' \leq M$, we adopt MARS approach. With this brief introduction to relevance feedback, we can now formulate the nearest neighbor search problem as follows: given R_t, W_t, and K', the weight matrix W_{t+1} is derived from W_t using some update scheme. Compute the next set of K nearest neighbors R_{t+1} using W_{t+1} using minimum number of computations.

4. Bound Computation

Given a query Q and a feature vector F_i, lower and upper bound of distance of $d(Q, F_i, W^t)$ are defined as $L_i(Q, W_t)$ and $U_i(Q, W_t)$ so that the following inequality holds:

$$L_i(Q, W_t) \leq d(Q, F_i, W_t) \leq U_i(Q, W_t) \quad (5)$$

4.1. Weighted Euclidean Distance

The computation of lower and upper bound of distance $d(Q, F_i, \Lambda_t)$ using VA-file index is straightforward for a diagonal Λ_t. If W_t is a diagonal matrix with non-negative entries, $\Lambda_t = diag(\lambda_1, ..., \lambda_M)$, then [8]:

$$L_i^2(Q, W_t) = [l_{i1}, l_{i2}, \ldots, l_{iM}]^T \Lambda_t [l_{i1}, l_{i2} \ldots, l_{iM}] \quad (6)$$

$$U_i^2(Q, W_t) = [u_{i1}, u_{i2} \ldots, u_{iM}]^T \Lambda_t [u_{i1}, u_{i2} \ldots, u_{iM}]$$

where:

$$l_{ij} = \begin{cases} q_j - b_{l+1j} & q_j > b_{l+1j} \\ 0 & q_j \in [b_{lj}, b_{l+1j}] \\ b_{lj} - q_j & q_j < b_{lj} \end{cases}$$

$$u_{ij} = \begin{cases} q_j - b_{lj} & q_j > b_{l+1j} \\ max(q_j - b_{lj},\ b_{l+1j} - q_j) & q_j \in [b_{lj}, b_{l+1j}] \\ b_{l+1j} - q_j & q_j < b_{lj} \end{cases}$$

4.2. General Quadratic Distance

For the case of general quadratic distance metric, nearest neighbor query becomes an ellipsoid query.

Locus of all points F_i having a distance $d(Q, F_i, W_t) \leq \epsilon$ is an ellipsoid centered around query point Q. It is difficult to determine whether a general ellipsoid intersects a cell in the original feature space, see Figure 1(a) for illustration of lower and upper bounds in case of weighted Euclidean distance and the quadratic distance measure. For quadratic metric, exact distance computation between the query object Q and a rectangle $C(F_i)$ requires numerically extensive quadratic programming approach. That would undermine the advantages of using an indexing structure.

Conservative bounds on rectangular approximations introduced in [4, 10] allow us to avoid the exact distance computation for query object Q and every approximation cell $C(F_i)$. However, for restrictive bounds, the distance computation stays quadratic with number of dimensions M. In [10], spatial transformation of the feature vector space significantly reduces CPU cost. We adopt similar approach, further reducing computational cost and improving efficiency by using the fact that it makes no difference whether to determine Euclidean or weighted Euclidean distance.

Assuming that distance matrix W_t is real, symmetric, and positive definite, we can factorize W_t as $W_t = P_t^T \Lambda_t P_t$, $P_t * P_t^T = I$, and the distance is:

$$d^2(Q, F_i, W_t) = (P_t(Q - F_i))^T \Lambda_t (P_t(Q - F_i)) \quad (7)$$

Define a rotational mapping of a point D as $D \rightarrow D'$, where $D' = P_t D$. All quadratic distances in the original space transform to weighted Euclidean distances in the mapped space. Cell $C(D)$ that approximates feature point D is transformed into a hyper parallelogram $C(D')$ in the mapped space, as illustrated in Figure 1(b). The parallelogram $C(D')$ can be approximated with bounding hyper rectangular cell $C'(D')$. The approximation $C(D)$ only specifies the bounding rectangle position in the mapped space. The size of relative bounding rectangle depends only on the cell size in the original space, and the rotation matrix P. Relative bounding rectangle in the mapped space can be computed before Phase I filtering.

Note that the W_t update is computed before approximation level filtering. Weight matrix in the mapped space is Λ_t, and the quadratic distance becomes a weighted Euclidean distance. Lower and upper bounds, $L_i(Q, W_t)$ and $U_i(Q, W_t)$, respectively, are approximated in the mapped space with $L_i(Q', \Lambda_t)$ and $U_i(Q', \Lambda_t)$, using weighted Euclidean matrix Λ_t, as in (6). Also, $L_i(Q', \Lambda_t) \leq L_i(Q, W_t)$, and $U_i(Q, W_t) \leq U_i(Q', \Lambda_t)$, and therefore:

$$L_i(Q', \Lambda_t) \leq d(Q, F_i, W_t) \leq U_i(Q', \Lambda_t) \quad (8)$$

5. Adaptive Nearest Neighbor Search

Let $R_{t-1} = \{F_k^{(t-1)}\}$ be the set of K nearest neighbors of query Q at iteration $t-1$ under weight matrix W_{t-1}, and $r_t(Q)$ be the maximum distance between Q and the items in R_t. Define $r_t^u(Q)$ as $r_t^u(Q) = max\{d(Q, F_k^{(t-1)}, W_t)\}, k \in [1, K]$.

Claim: When W_{t-1} is updated to W_t, we can establish an upper bound on $r_t(Q)$ as:

$$r_t(Q) \leq r_t^u(Q)$$

This claim states that the maximum of the distances between the query Q and objects in R_t computed using W_t, can not be larger than the maximum distance between the query Q and the objects in R_{t-1} computed using W_t. This is intuitively clear, since R_t is the set of K nearest neighbors from Φ to Q, under W_t.

Phase I filtering determines a subset of approximations from which the K nearest neighbors can be retrieved, as described in Section 2. Let $N_1^{opt}(Q, W_t)$ be the minimal set of approximations that contain K nearest neighbors. The best case scenario for Phase I filtering is to identify exactly this subset $N_1(Q, W_t) = N_1^{opt}(Q, W_t)$. For approximation $C(F_i)$ to be a qualified one in $N_1^{opt}(Q, W_t)$, its lower bound $L_i(Q, W_t)$ must satisfy:

$$L_i(Q, W_t) < r_t^u(Q) \quad (9)$$

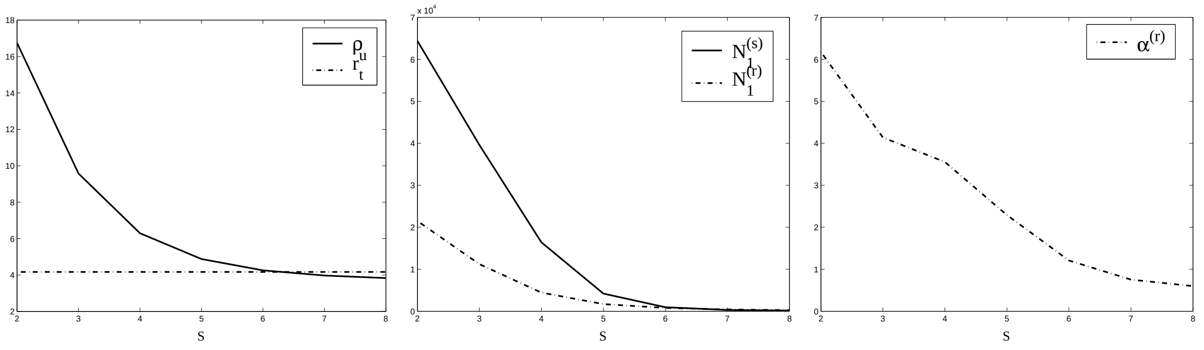
Let ρ be the K^{th} largest upper bound encountered so far during a sequential scan of approximations. In the standard approaches [8], the approximation $C(F_i)$ is included in $N_1(Q, W_t)$ only if $L_i(Q, W_t) < \rho$, and the ρ is updated if $U_i(Q, W_t) < \rho$. Only the K^{th} smallest upper bound from the scanned approximations is available and used for filtering.

The best filtering bound in Phase I is $min(r_t^u(Q), \rho)$. The relationship $r_t^u(Q, W_t) \leq \rho$ is satisfied for the larger part of database scanning. In general, fewer candidates need to be examined in Phase I filtering if we use $r_t^u(Q)$ as a filtering bound.

5.1. An adaptive K-NN search algorithm

For $t = 1$, the K-NN approximation filtering phase reduces to standard Phase I filtering. Data filtering step results in a set of K nearest neighbors R_t. Then, the algorithm starts a new iteration and assigns $t + 1$ to t. User identifies positive examples in R_{t-1}. Based on that, W_{t-1} is updated to W_t, and P_t and Λ_t are computed in the process. $r_t^u(Q)$ is computed before Adaptive Phase I filtering. For $t > 1$, the adaptive Phase I filtering is:

```
Adaptive Phase I:
Compute $\rho = r_t^u(Q) = max\{d(Q, F_i^{t-1}, W_t)\}$;
for $i \in [0, N-1]$ do
    Compute $L_i(Q, W_t)$ as presented in Section 4.2
    if $L_i(Q, W_t) \leq \rho$ then
        Insert $C(F_i)$ into $N_1(Q, W_t)$;
    end if
end for
```



(a) Phase I Filter Bounds (b) Average Number of Phase 1 Candidates (c) Effectiveness measure α

Figure 2: Weighted Euclidean distance: a) Phase I selectivity bound ρ for standard filtering and r_t^u for adaptive one; b) Average number of Phase I candidates for standard $N_1^{(s)}$ and adaptive approach $N_1^{(r)}$; c) Effectiveness measure $\alpha(r)$ for Phase I adaptive method

There are two essential differences between the the existing approaches and the proposed approach: (1) Only the absolute position of the approximation rectangle is computed during the Phase I filtering. We can compute the grid mapping in advance. Lower bound $L_i(Q, W_t)$ computation during Phase I is linear with number of dimensions for every VA-file index. (2) The proposed constraint in approximation filtering is based on relevance feedback results, and it gives us a smaller set of false candidates.

A simple example is illustrated in Figure 1(c). User identified points E and G as the ones similar to the query point Q. Based on that, the weight matrix was updated from W_1 to W_2. Under new distance metric, $r_2^u(Q) = d(Q, E, W_2)$ and the search space is restricted to the shaded area in Figure 1(c).

6. Experiments

In this section, we demonstrate the effectiveness of the proposed approach over a dataset of $N = 90774$ texture feature vectors, $M = 60$ dimensions each. Experiments are carried out for different resolutions S used for constructing standard VA, $S = 2\bar{,}8$. 2^S bits are assigned to each of the M uniformly partitioned feature dimensions. We compare the standard VA approach to K-NN search to our adaptive for different S, in relevance feedback presence.

Queries Q_i are selected from the dataset to cover both dense cluster representatives and outliers in the feature space. For each query Q_i, $K = 70$ nearest neighbors are retrieved during each iteration. The feedback from the user is based on texture relevance only. For a specific query, the user selects $K' = 65$ relevant retrievals to update the distance measure. The distance metric is updated before every iteration (Section 3).

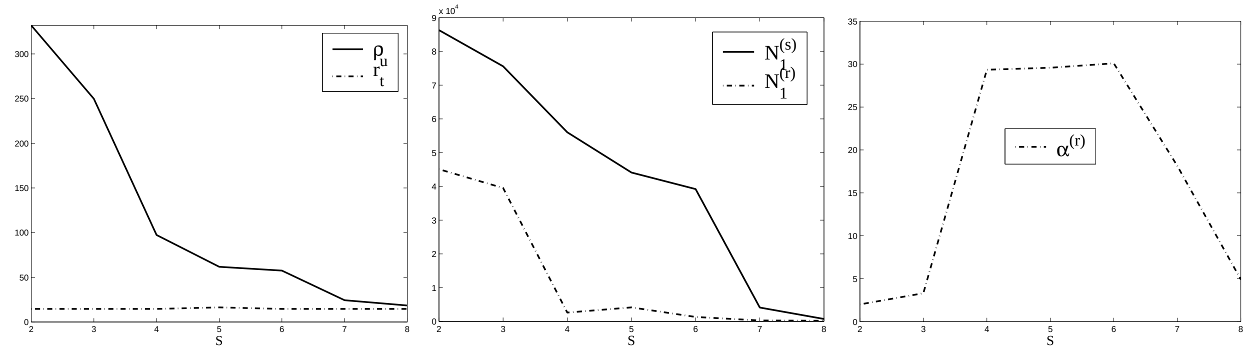
The approximations are constructed using the standard VA index, as described in Section 2.1. We choose to assign 2^S bits to all dimensions. Each of the feature dimensions is uniformly partitioned, at different resolutions. Experiments are carried out for different numbers of bits assigned to every dimension, $S = 2\bar{,}7$. Larger value corresponds to the approximation constructed at a finer resolution. We compare the standard VA approach to computing the K nearest neighbors to our proposed method for different S, in relevance feedback presence. Computation of upper bounds for the standard approach adds marginal complexity, since $U_i(Q, W_t)$ is found in a mapped space. Filter bound ρ rapidly increases when the resolution is smaller, due to the larger sizes of the hyper rectangles used in the corresponding approximations. A larger difference between r_t^u and ρ should impose a significant improvement for the proposed method. Thus, in the presence of relevance feedback we can either save some memory for approximation storage or reduce the number of disc access for the same resolution. Figures 2(a) and 3(a) show that the difference between $r_t^u(Q)$ and ρ increases for lower values of S, for weighted Euclidean distance and quadratic distance, respectively.

For a given resolution S, and the query vector Q_i, number of candidates from Phase I filtering is noted as $N_1^{(s)}(Q_i)$ for standard approach and $N_1^{(r)}(Q_i)$ for adaptive approach. Define an average number of Phase I candidates over the example queries as $N_1^{(s)}$ for standard approach and $N_1^{(r)}$ for adaptive approach. Also, define an effectiveness measure of the proposed method:

$$\alpha^{(r)} = \frac{1}{I} \sum_{i=1}^{I} \frac{N_1^{(s)}(Q_i)}{N_1^{(r)}(Q_i)};$$

We average over $I = 20$ query vectors.

The number of candidates resulting from the standard and adaptive Phase I filtering, under weighted Euclidean distance metric, is given in Figure 2(b) and the average gain α of the proposed method is given in Figure 2(c). For quadratic distance, we update the weight matrix as defined in (3), for $K' = 65$ relevant examples.



(a) Phase I Filter Bounds (b) Average Number of Phase I Candidates (c) Effectiveness measure α

Figure 3: Quadratic distance metric: a) Phase I selectivity bound ρ for standard filtering and r_t^u for adaptive one; b) Average number of cells accessed in Phase I for standard $N_1^{(s)}$ and adaptive approach $N_1^{(r)}$; c) Effectiveness measure $\alpha^{(r)}$ for Phase I adaptive method

Proposed method improves the efficiency of Phase I for quadratic distance metric, Figure 3(b). The effectiveness $\alpha^{(r)}$ is significant, specially for coarser approximations, see Figure 3(c).

This is a real dataset, and α is not monotone over S, since the results are strongly correlated with distribution of the feature points in 60-dimensional space. We can conclude that the minimum gain of the proposed adaptive filtering is still significant at every resolution, for both weighted Euclidean and quadratic distance matrix.

7. Summary

We presented a framework that supports efficient retrieval of relevance feedback results, even when the similarity metric is quadratic. Based on user's input, the weight matrix of feature space is modified in every iteration. A new set of nearest neighbors is computed. The cost of nearest neighbor computation in each iteration is quadratic with number of dimensions and linear with number of items. The proposed scheme allows us to use rectangular approximations for nearest neighbor search under quadratic distance metric and exploits correlations between two consecutive nearest neighbor sets. We believe that this is the first time that this issue of nearest neighbor update using VA-file indexing and relevance feedback metric update has been addressed in content-based search and retrieval research. Proposed approach significantly reduces the overall search complexity.

8. Acknowledgments

This research was supported in part by The Institute of Scientific Computing Research (ISCR) award under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No.W-7405-ENG-48, ONR# N00014-01-1-0391, NSF Instrumentation #EIA-9986057, and NSF Infrastructure #EIA-0080134.

References

[1] Peng Wu and B.S. Manjunath, "Adaptive nearest neighbor search for relevance feedback in large image datasets," in *Proc. ACM Multimedia (MM)*, October 2001, pp. 87–98.

[2] A. L. Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez, "A framework for learning query concepts in image classification," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, November 1999, vol. 1, pp. 423–431.

[3] J. J. Rocchio, "Relevance feedback in information retrieval," in *The SMART Retrieval System– Experiments in Automatic Document Processing*, Englewood Cliffs, NJ, 1971, pp. 313–323.

[4] M. Ankerst, B. Braunmüller, H.-P. Kriegel, and T. Seidl, "Improving adaptable similarity query processing by using approximations," in *Proc. Very Large Data Bases (VLDB)*, August 1998, pp. 206–217.

[5] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Query databases through multiple examples," in *Proc. Very Large Databases (VLDB)*, August 1997, pp. 218–227.

[6] Y. Rui and T. Huang, "Optimizing learning in image retrieval," in *Proc IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2000, vol. 1, pp. 236–243.

[7] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Computing Surveys*, vol. 33, no. 3, pp. 322–373, September 2001.

[8] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proc. Very Large Data Bases, (VLDB)*, August 1998, pp. 194–205.

[9] B.S.Manjunath, P. Salembier, and T. Sikora, Eds., *Introduction to MPEG7: Multimedia Content Description Interface*, John Wiley & Sons Ltd., 2002.

[10] Y. Sakurai, M. Yoshikawa, R. Kataoka, and S. Uemura, "Similarity search for adaptive ellipsoid queries using spatial transformation," in *Proc. Very Large Databases (VLDB)*, September 2001, pp. 231–240.