Kernel Vector Approximation Files for Relevance Feedback Retrieval in Large Image Databases

Douglas R. Heisterkamp
Computer Science Department
Oklahoma State University
Stillwater, OK 74078
drh@ieee.org

Jing Peng
Electrical Engr. and Computer Science
Tulane University
New Orleans, LA 70118
jp@eecs.tulane.edu

December 1, 2004

Abstract

Many data partitioning index methods perform poorly in high dimensional space and do not support relevance feedback retrieval. The vector approximation file (VA-File) approach overcomes some of the difficulties of high dimensional vector spaces, but cannot be applied to relevance feedback retrieval using kernel distances in the data measurement space. This paper introduces a novel KVA-File (kernel VA-File) that extends VA-File to kernel-based retrieval methods. An efficient approach to approximating vectors in an induced feature space is presented with the corresponding upper and lower distance bounds. Thus an effective indexing method is provided for kernel-based relevance feedback image retrieval methods. Experimental results using large image data sets (approximately 100,000 images with 463 dimensions of measurement) validate the efficacy of our method.

1 Introduction

Content-based image retrieval is an active area of research. Kernel methods have been exploited in classification [16, 11, 12], regression, [5, 20, 24], and information retrieval [5, 20]. Relevance feedback is an approach that attempts to improve image retrieval by learning from the user's opinion of a current set of retrieved images. A variety of retrieval methods have been developed that

exploit relevance feedback to create flexible retrieval metrics, thereby improving accuracy. Kernel methods for image retrieval with relevance feedback have been presented and shown to outperform weighted Euclidean distances [3, 9, 29]. The relevance feedback methods help adapt the distance function to the user's sense of relevance. This yields improved accuracy but the changing distance function causes havoc with index structures and often invalidates them. Thus the improved accuracy is at the expense of reduced efficiency because direct computation of distance between the query and every image in the database is required.

Few systems have been developed that address the apparent flexible metric/indexing dilemma in relevance feedback retrieval. While approximation based index techniques [25, 27] have shown promise in large databases with high dimensions and support relevance feedback retrieval with linear weightings (i.e., database objects still maintain the relative positions along each axis) [10, 15, 17], they are unable to support nonlinear transformations, such as kernel distances [9, 3, 29]. A vector approximation file (VA-File) takes a signature or filter approach to indexing data [25]. By sequentially processing a compressed approximation of the data, VA-File is able to filter most data vectors and need only retrieve a small fraction of the actual data. In [27], retrievals between consecutive iterations of relevance feedback are exploited to improve the filtering and thus reduce the amount of computation required for nearest neighbor search. To be able to conduct the filtering, upper and lower bounds on the distance from the query to the data point are calculated. This calculation from VA-File, however, is invalid for kernel-based approaches.

To support kernel distances, an index structure must be built in kernel induced feature space, resulting in a kernel index. Any metric space based index methods, such as M-Trees [4], can be constructed in the feature space. In [14], M-Trees were used to create a kernel index scheme. For approaches such as 1-SVM [3], that do not modify the kernel, M-Tree can be an effective kernel indexing scheme. But if in the learning process during relevance feedback, the kernel is modified, as in AQK [9], the index structure of an M-Tree can no longer be used. Nor can other tree-based index structures [4, 7, 8, 28] be used.

This paper presents the KVA-File index that allows an efficient approach to approximating vectors in the induced feature space and determining the corresponding upper and lower distance bounds needed for nearest neighbor search. In Section 2, a motivating example of the use of kernel distance and two current kernel-based relevance feedback image retrieval techniques are presented. In Section 3, VA-File is reviewed. The KVA-File approach is presented in Section 4. In Section 5, results from synthetic and real data is presented.

2 Kernel Distance for Relevance Feedback

The kernel trick is a popular method for extending algorithms into a non-linearly transformed space. The space containing the original data is called the *input space*. The kernel method creates a new space called the *feature space* that is a

non-linear transformation of the input space. The kernel trick has been applied to numerous problems [5, 18, 12]. The kernel allows an algorithm to work in a kernel induced feature space. If $\boldsymbol{\phi}(\mathbf{x})$ is a mapping of a point $\mathbf{x}$ in the input space to the feature space then the kernel calculates the dot product in the feature space of the images of two points from input space

$$k(\mathbf{a}, \mathbf{b}) = < \boldsymbol{\phi}(\mathbf{a}), \boldsymbol{\phi}(\mathbf{b}) > .$$

Common kernels are Gaussian,

$$k(\mathbf{a}, \mathbf{b}) = e^{-\frac{||\mathbf{a}-\mathbf{b}||^2}{2\sigma^2}}$$

and polynomial,

$$k(\mathbf{a}, \mathbf{b}) = (1+ < \mathbf{a}, \mathbf{b} >)^d.$$

Distance in the feature space may be calculated by means of the kernel [24, 5]. With $\mathbf{a}$ and $\mathbf{b}$ in the input space, the squared feature space distance is

$$\text{dist}(\mathbf{a}, \mathbf{b})^2 = \|\boldsymbol{\phi}(\mathbf{a}) - \boldsymbol{\phi}(\mathbf{b})\|^2 = k(\mathbf{a}, \mathbf{a}) - 2k(\mathbf{a}, \mathbf{b}) + k(\mathbf{b}, \mathbf{b}). \quad (1)$$

Two known kernel distances in the relevance feedback literature are Adaptive Quasiconformal Kernel (AQK) [9] and One-Class SVM (1-SVM) [3, 21]. Before presenting those approaches, we first present an motivating example for the use of kernel distance.

2.1 Motivating Example

In this subsection, we look at the effect on distance by a simple method of query re-weighting conducted in input space and in feature space. The 2-D data points from a single class are presented in Figure 1(a). A Gaussian kernel is used to induce the feature space. The distance map of all location to the initial query using input space distance (Euclidean distance) is presented in Figure 1(b). The distance map induce in the input space by using the kernel distance in feature space to the initial query is presented in Figure 1(c). The relative ordering of points by distance is the same in Figures 1(b) and 1(c) since a Gaussian kernel is being used, though the rate of change in distance values is different.

An updated query location is created by using the center of the smallest hyper-sphere containing the class data (a circle in 2-D). This corresponds to the one class SVM technique that will be presented in Subsection 2.3. In this example, there is basically no difference between the mean of the class points and the center of this circle. The distance map created by shifting the query to this location in input space is presented in Figure 1(d). The distance map induced in the input space by using kernel distance to new query location that is the center of the hypersphere in feature space is presented in Figure 1(e). The new query location in feature space does not have a corresponding location in input space. Although the same method of query shifting was conducted in both the input and the feature space, the distance map from the kernel-based distance is a better expression of the class distribution.

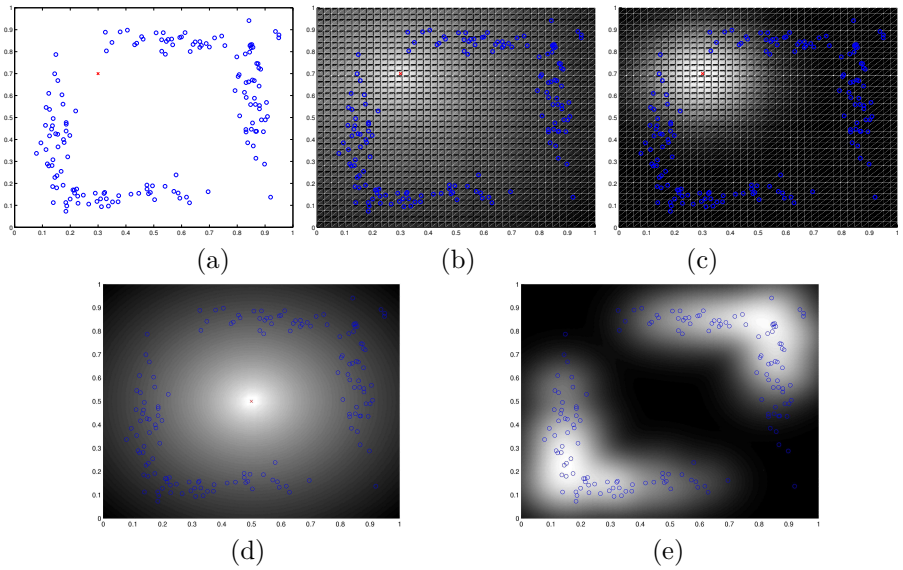


Figure 1: Feature space distance versus input space distance: (a) the original data locations of a class; (b) distance map to query using input space distance; (c) distance map to query using feature space distance; (d) query is moved to the center of the smallest circle contained the class data points and new distance map using input space distance is created; (e) query is moved to the center of the smallest circle containing the class data points in feature space and new distance map using feature space distance is created

A 1-D example is presented in Figure 2 to help explain why this can occur. The initial data is presented in Figure 2(a). The 1-D input space is mapped to a 2-D curve and is presented in Figure 2(b). It is now possible to separate the two classes. The smallest circle that contains one of the classes is presented in 2(c). Note that all points of that class are closer to the center of the circle than any point from the other class. Also note that the circle center does not lie on the 2-D curve and thus does not have a pre-image in the original 1-D input space.

2.2 Adaptive Quasiconformal Kernel

Adaptive quasiconformal kernel (AQK) distance [9, 16] combines the kernel distance (1) with a quasiconformal mapping [1],

$$\tilde{k}\left(\mathbf{a},\mathbf{b}\right) = c(\mathbf{a})c(\mathbf{b})k\left(\mathbf{a},\mathbf{b}\right), \tag{2}$$

to create a new kernel $\tilde{k}$ and hence a new kernel distance:

$$\begin{aligned} \mathrm{dist}(\mathbf{a},\mathbf{b})^2 &= \tilde{k}\left(\mathbf{a},\mathbf{a}\right) - 2\tilde{k}\left(\mathbf{a},\mathbf{b}\right) + \tilde{k}\left(\mathbf{b},\mathbf{b}\right) \\ &= c(\mathbf{a})^2 k\left(\mathbf{a},\mathbf{a}\right) - 2c(\mathbf{a})c(\mathbf{b})k\left(\mathbf{a},\mathbf{b}\right) + c(\mathbf{b})^2 k\left(\mathbf{b},\mathbf{b}\right) \end{aligned} \tag{3}$$

where $c(\mathbf{a})$ is a positive real valued function of $\mathbf{a}$ in the input space. One can select $c(\mathbf{a})$ from relevance feedback to expand the spatial resolution around irrelevant samples and contract the spatial resolution around relevant samples. That is, distance to irrelevant samples from the query is increased and distance to relevant samples are decreased. One such candidate is

$$c(\mathbf{a}) = \frac{\Pr(\mathbf{a}|I)}{\Pr(\mathbf{a}|R)}$$

where $\Pr(\mathbf{x}|I)$ ($\Pr(\mathbf{x}|R)$) denotes the density of irrelevant (relevant) retrievals. The resulting distance (when Gaussian kernels are applied) is related to the weighted Chi-squared distance [9].

2.3 One-Class SVM

One-class SVM (1-SVM) kernel distance is the distance from a sample to the center of the smallest hypersphere that includes most of the relevant retrievals from the previous iterations [3, 21]. After finding the center

$$\mathbf{c} = \sum_i \gamma_i \boldsymbol{\phi}(\mathbf{x}_i), \quad \text{where} \quad \gamma_i \geq 0 \quad \text{and} \quad \sum_i \gamma_i = 1,$$

the one-class SVM kernel distance of vector $\mathbf{z}$ to $\mathbf{c}$ is

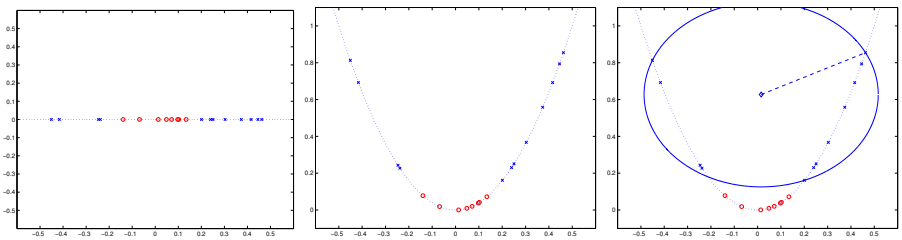
$$\text{dist}(\mathbf{z}, \mathbf{c})^2 = k(\mathbf{z}, \mathbf{z}) - 2\sum_i \gamma_i k(\mathbf{x}_i, \mathbf{z}) + \sum_{i,j} \gamma_i \gamma_j k(\mathbf{x}_i, \mathbf{x}_j). \tag{4}$$

The γ_i's are the Lagrangian multipliers from the solution of the quadratic programming problem

$$\min_{R,\xi,\mathbf{c}} R^2 + \frac{1}{vl}\sum_i \xi_i \quad \text{subject to} \quad \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0$$

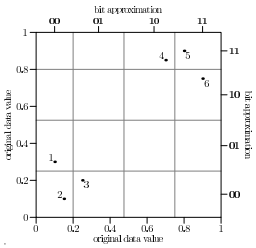
3 Vector Approximation File (VA-File)

The vector-approximation file (VA-File) uses a signature file as a filter [25, 26]. The signature file is a compressed approximation of the original data file. Each



(a) (b) (c)

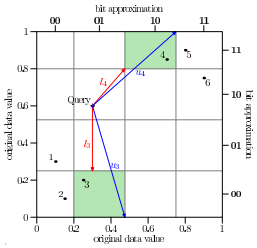
Figure 2: Example 1D to 2D mapping



(a)

	Vectors	VA-File
1	(0.1,0.3)	00 01
2	(0.15,0.1)	00 00
3	(0.25,0.2)	01 00
4	(0.7,0.85)	10 11
5	(0.8,0.9)	11 11
6	(0.9,0.75)	11 10

(b)



(c)

Figure 3: VA-File approximations

data vector from the original data is stored in the approximation file as the bit encoding of the hypercube in which it lies. For example, in Figure 3, two bits per dimension were used to approximate the data vectors. Vector 3, $[0.25, 0.2]$, is represented by the bit pattern 0100, meaning bin 01 of first dimension and bin 00 of second dimension. Typically, the compressed file is 10% to 15% of the size of the original data file.

The distance of a point to an approximated vector is bounded by the maximum and minimum distances to the hypercube that the bit encoding of the approximation represents. The maximum and minimum distances of a point to the hypercube provides an upper and lower bounds on the distance between the query location and the original data point. For example, in Figure 3, the distance from the query to vector 3 is within the lower bound of l_3 and the upper bound of u_3. Similarly, l_4 and u_4 are the lower and upper bounds of the distance of vector 4 from the query.

When using a VA-File, a K nearest neighbor search is a two phase processes. In phase one, a filtering of the possible K-NN is done by a sequential scan of the approximation file. The filtering process creates a candidate list. As each approximated vector is processed, if its lower bound is less than the current Kth closest upper bound then it is added as a candidate for phase two. In phase two, the candidates are visited in ascending order of lower bound until the lower bound of the next candidate is greater than the actual distance to the current Kth nearest neighbor.

In [22, 27], retrievals between consecutive iterations with relevance feedback are exploited to improve the initial bounds used in the filtering process that creates the candidate list. This results in greatly reducing the size of the candidate list.

4 Kernel Vector Approximation File

While the existing input space techniques, such as VA-File, can handle linear transformations of the input space, they are no longer valid for the non-linear transformation induced by a kernel. The goal of this section is to develop a feature space approximation that allows an efficient calculation of upper and

lower distance bounds of kernel distances.

To create an approximation, we use a reduced set of orthonormal basis vectors in the feature space. The feature space location of a data point is projected into the reduced space and used as the approximation. In addition, the magnitude of the error (the remaining component orthogonal to basis) is also used as part of the approximation.

Both Kernel-PCA [19] and a Gram-Schmidt approach [6, 2] have been used to find an orthonormal basis of the feature space. The eigenvectors from Kernel-PCA that correspond to the largest eigenvectors would yield the best approximation of the locations in feature space. The problem with using Kernel-PCA is that the representation of the eigenvectors is not compact (it can be on the order of the number of vectors in the data set). For the work in this paper, we used a Gram-Schmidt approach similar to the efficient algorithm in [2] to find an orthonormal basis in the feature space. One of the sparse approaches for Kernel-PCA [23] may also be useful.

Given a reduced set of basis vectors, $\boldsymbol{\nu}_t$, a point in feature can be represented by a linear combination of basis vectors and a component, $\boldsymbol{\phi}^{\perp}(\mathbf{x})$ that is orthogonal to the basis. For example, the feature space points $\mathcal{P}$ and $\mathcal{Q}$ corresponding to the input space points $\mathbf{x}_p$ and $\mathbf{x}_q$ can be represent by

$$\mathbf{P} = \boldsymbol{\phi}(\mathbf{x}_p) = \boldsymbol{\phi}^{\perp}(\mathbf{x}_p) + \sum_{t=0}^{d-1} \alpha_t \boldsymbol{\nu}_t \tag{5}$$

$$\mathbf{Q} = \boldsymbol{\phi}(\mathbf{x}_q) = \boldsymbol{\phi}^{\perp}(\mathbf{x}_q) + \sum_{t=0}^{d-1} \beta_t \boldsymbol{\nu}_t \tag{6}$$

where d is the number of basis vectors. The squared distance in feature space between points $\mathcal{P}$ and $\mathcal{Q}$ using this representation is

$$\begin{aligned} \mathrm{dist}(Q,P)^2 = {} & \boldsymbol{\phi}^{\perp}(\mathbf{x}_q)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_q) + \boldsymbol{\phi}^{\perp}(\mathbf{x}_p)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_p) \\ & -2\boldsymbol{\phi}^{\perp}(\mathbf{x}_q)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_p) + \sum_{t=0}^{d-1} (\alpha_t - \beta_t)^2. \end{aligned} \tag{7}$$

Thus we only need the coefficients of the point's representation in terms of the basis vectors and the remaining orthogonal vectors to exactly determine the distance between the two points.

An incremental method to find an orthogonal basis and create data approximations is presented in Figure 4. The algorithm makes multiple passes through the data file. Each pass updates the approximation for each data vector with respect to the next basis vector. The running time of the algorithm is dominated by the line calculating α_{t+1} and thus yields a time complexity of $\mathcal{O}\left(k^2 n\right)$ where k is number of basis vectors and n is the number of data vectors. The storage complexity of the algorithm is $\mathcal{O}\left(n + k^2\right)$ where the data and approximation vectors are on the order of n and the basis vectors are represented in the lower triangle of a $k \times k$ matrix. Since the algorithm sequentially processes both the

Select first basis vector corresponding to data vector $\mathbf{x}_s$.
Set $t = 0$
for each data vector $\mathbf{x}_p$ **do**
  $\alpha_t = \frac{k(\mathbf{x}_p, \mathbf{x}_s)}{\sqrt{k(s,s)}}$
  $\hat{\mathbf{G}}_t(p,p) = k(p,p) - \alpha_t^2$
end-for
while need more basis vectors **do**
  Select $\mathbf{x}_q$ such that $\hat{\mathbf{G}}_t(q,q)$ is maximum for next basis vector.
  Let β_i's and $\hat{\mathbf{G}}_t(q,q)$ be the current approximation of $\boldsymbol{\phi}(\mathbf{x}_a)$.
  for each data vector $\mathbf{x}_p$ **do**
    $\alpha_{t+1} = \frac{k(\mathbf{x}_p, \mathbf{x}_q) - \sum_{i=0}^{t} \alpha_i \beta_i}{\sqrt{\hat{\mathbf{G}}_t(q,q)}}$
    $\hat{\mathbf{G}}_{t+1}(p,p) = \hat{\mathbf{G}}_t(p,p) - \alpha_{t+1}^2$
  end-for
  Record $\mathbf{x}_q$, $\hat{\mathbf{G}}_t(q,q)$, and β_i's for later use.
  increment t
end-while

Figure 4: Incremental Orthogonalization Algorithm

data vectors and the approximation vectors, only an individual block of each file needs to be in memory at any one point in time.

In addition to finding the basis, this algorithm stores the coefficients of a point's representation in the basis and the magnitude of the orthogonal vector into an approximation file. That is, it determines the coefficients, α_i and β_i, and the magnitudes, $\boldsymbol{\phi}^{\perp}(\mathbf{x}_p)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_p) = \hat{\mathbf{G}}_d(p,p)$ and $\boldsymbol{\phi}^{\perp}(\mathbf{x}_q)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_q) = \hat{\mathbf{G}}_d(q,q)$, needed in the distance equation (7). The only remaining unknown term in (7) is $\boldsymbol{\phi}^{\perp}(\mathbf{x}_q)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_p)$. But this is also equal to

$$\sqrt{\boldsymbol{\phi}^{\perp}(\mathbf{x}_q)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_q)} \sqrt{\boldsymbol{\phi}^{\perp}(\mathbf{x}_p)^T \boldsymbol{\phi}^{\perp}(\mathbf{x}_p)} \cos\theta$$

where θ is the angle between the two vectors. In terms of Algorithm 4, this is expressed as $\sqrt{\hat{\mathbf{G}}_d(q,q)} \sqrt{\hat{\mathbf{G}}_d(p,p)} \cos\theta$. The angle is not represented in the approximation but the extremes can be used to generate bounds on the distance. Thus the upper distance bounds, $\mathcal{U}(Q,P)$, and the lower distance bounds, $\mathcal{L}(Q,P)$, is

$$\begin{aligned} \mathcal{U}(Q,P) &= \Big(\hat{\mathbf{G}}_d(q,q) + \hat{\mathbf{G}}_d(p,p) + 2\sqrt{\hat{\mathbf{G}}_d(q,q)} \sqrt{\hat{\mathbf{G}}_d(p,p)} \\ &\quad + \sum_{t=0}^{d-1} (\alpha_t - \beta_t)^2 \Big)^{\frac{1}{2}} \qquad (8) \\ \mathcal{L}(Q,P) &= \Big(\hat{\mathbf{G}}_d(q,q) + \hat{\mathbf{G}}_d(p,p) - 2\sqrt{\hat{\mathbf{G}}_d(q,q)} \sqrt{\hat{\mathbf{G}}_d(p,p)} \\ &\quad + \sum_{t=0}^{d-1} (\alpha_t - \beta_t)^2 \Big)^{\frac{1}{2}} \qquad (9) \end{aligned}$$

The ranges for each α_i and for $\hat{\mathbf{G}}_n(p,p)$ can be partitioned into bins represented by a bit encoding. The upper and lowers bounds are then adjusted in a similar fashion to that of VA-File. Note that once the coefficients and magnitudes (i.e., the α's and $\hat{\mathbf{G}}_d(p,p)$) are stored into the approximation file, all later upper and lower distance bound calculations do not require a kernel evaluation.

4.1 Adapting the distance bounds due to relevance feedback

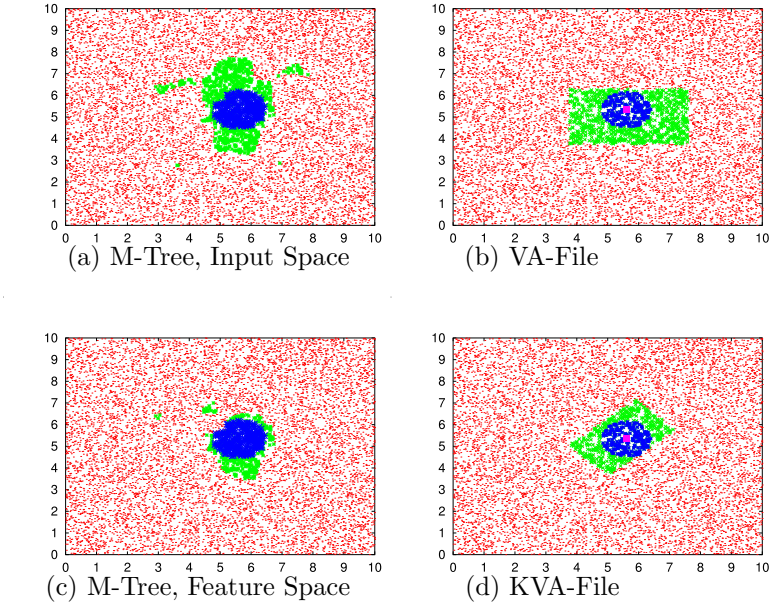
In One-Class SVM (see Section 2.3) a query location is presented as a linear combination of feature space points: $\mathbf{Q} = \sum_i \gamma_i \boldsymbol{\phi}(\mathbf{x}_i)$. The same process as the previous section can be used to generate the upper and lower bounds:

$$\begin{aligned}
\mathcal{U}(Q,P) &= \Big(\sum_i \sum_j \gamma_i \gamma_j k(\mathbf{x}_i, \mathbf{x}_j) + 2\sqrt{\hat{\mathbf{G}}_n(p,p)} \sum_i \gamma_i \sqrt{\hat{\mathbf{G}}_n(i,i)} \\
&\quad -2 \sum_{t=0}^{n-1} \sum_i \alpha_t \gamma_i \beta_{i,t} + \hat{\mathbf{G}}_n(p,p) + \sum_{t=0}^{n-1} \alpha_t^2 \Big)^{\frac{1}{2}} \\
\mathcal{L}(Q,P) &= \Big(\sum_i \sum_j \gamma_i \gamma_j k(\mathbf{x}_i, \mathbf{x}_j) - 2\sqrt{\hat{\mathbf{G}}_n(p,p)} \sum_i \gamma_i \sqrt{\hat{\mathbf{G}}_n(i,i)} \\
&\quad -2 \sum_{t=0}^{n-1} \sum_i \alpha_t \gamma_i \beta_{i,t} + \hat{\mathbf{G}}_n(p,p) + \sum_{t=0}^{n-1} \alpha_t^2 \Big)^{\frac{1}{2}}
\end{aligned}$$

The Adaptive quasi-conformal kernel, [9], modifies the distance calculation by creating a new kernel, $\tilde{k}$, and resulting distance as equation (3). After substituting the new kernel into distance equation (7) and re-arraigning, the the upper and lower bounds on the distance become

$$\begin{aligned}
\mathcal{U}(Q,P) &= \big(c(\mathbf{x}_q)^2 \hat{\mathbf{G}}_d(q,q) + c(\mathbf{x}_p)^2 \hat{\mathbf{G}}_d(p,p) \\
&\quad + 2c(\mathbf{x}_q) c(\mathbf{x}_p) \sqrt{\hat{\mathbf{G}}_d(q,q)} \sqrt{\hat{\mathbf{G}}_d(p,p)} \\
&\quad + \sum_{t=0}^{d-1} (c(\mathbf{x}_p)\alpha_t - c(\mathbf{x}_q)\beta_t)^2 \big)^{\frac{1}{2}} \\
\mathcal{L}(Q,P) &= \big(c(\mathbf{x}_q)^2 \hat{\mathbf{G}}_d(q,q) + c(\mathbf{x}_p)^2 \hat{\mathbf{G}}_d(p,p) \\
&\quad - 2c(\mathbf{x}_q) c(\mathbf{x}_p) \sqrt{\hat{\mathbf{G}}_d(q,q)} \sqrt{\hat{\mathbf{G}}_d(p,p)} \\
&\quad + \sum_{t=0}^{d-1} (c(\mathbf{x}_p)\alpha_t - c(\mathbf{x}_q)\beta_t)^2 \big)^{\frac{1}{2}}.
\end{aligned}$$

For a query $\mathbf{x}_q$, we will have the actual data so we can calculate the exact $c(\mathbf{x}_q)$. But when processing the approximation file, we only have the approximations of the feature space point $\mathcal{P}$ and thus need to use a bound on the value of $c(\mathbf{x}_p)$. When using AQK, c is determined from Gaussians of known points so



(a) M-Tree, Input Space (b) VA-File

(c) M-Tree, Feature Space (d) KVA-File

Figure 5: Data points (green color) examined by the index to retrieve two hundred nearest neighbors (blue color) to a query at (5.5,5.5).

it is a straightforward, though tedious, process to determine the bounds on c at an approximated location.

5 Experimental Results

For illustration purposes, we present a simple two dimensional example in Figure 5. Ten thousand points were uniformly randomly generated over a fixed range in the input space. The points were non-linearly transformed into the feature space by the kernel method. A M-Tree [4] and a VA-File index were built in the input space and a M-Tree [14] and a KVA-File index were built in the feature space. A query to find the 200 nearest neighbors to a location was performed using each index. A 3-bit approximate was used for both VA-File and KVA-File. The KVA-File used two basis dimensions.

Comparing the number of candidates for which a distance calculation is performed in conducting the nearest neighbor search is informative. The M-Tree input space index searched 833 candidates to determine the 200 nearest neighbors. The VA-File input space index searched 933 candidates to determine the 200 nearest neighbors. The M-Tree feature space kernel index searched 454 candidates to determine the same 200 nearest neighbors. The KVA-File feature space index searched 548 candidates to determine the 200 nearest neighbors.

The feature space was induced by a Gaussian kernel and thus the relative

distance ranking of the data with respect to the query location is the same in feature space as the input space. Utilizing the index in feature space yielded approximately 40% reduction of computational load (455 vs. 833 and 548 vs. 933 distance calculations) to complete the same task. While this is simple 2D example, we also noticed a similar effect when processing real data. The tree approaches outperforms the corresponding vector approximation approaches in this example due to the low dimensionality of the data.

An explanation of a factor contributing to the better performance of the kernel indexes can be seen as follows. The exponential function used in the Gaussian kernel makes a change in the relative distance between two points by expanding the distance resolution of close items and contracting the distance resolution of far items. This can create a beneficial effect on nearest neighbor search. By expanding the distance resolution of close records, it becomes easier to differentiate between the k and the $k+1$ neighbor. The lost of distance resolution of far records has the effect of pushing far location even farther away. Both effects contribute to an increase in the effectiveness of the bounds used for early pruning in the search algorithms.

In the following, we use block I/O to compare nearest neighbor search using the kernel indexing methods of (1) M-Tree in feature space and (2) KVA-File with different kernel-based relevance feedback approaches (1-SVM and AQK) on the following two real data sets.

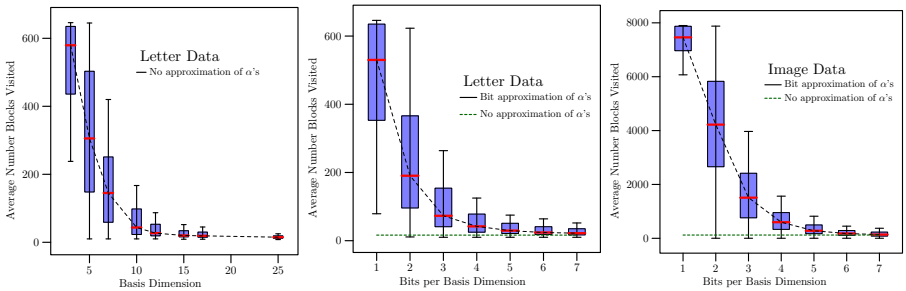
LIRD: The **Letter Image Recognition Data** (LIRD) data set is taken from [13]. This data set consists of a large number (20,000) of black-and-white rectangular letter images as one of the 26 upper-case letters in the English alphabet. Sample images are shown in Figure 6. The characters are based on 20 Roman alphabet fonts. They represent five different stroke styles and six different letter styles. Each letter is randomly distorted through a quadratic transformation to produce a set of 20,000 unique letter images that are then converted into 16 primitive numerical features. Basically, these numerical features are statistical moments and edge counts. The LIRD data used block size of 1988 bytes (31 records per block).

Figure 6: Sample letter images.

Image Data: The Hemera Photo-Object image data set consists of 94800 images that are very heterogeneous with annotated ground truth. Sample images are shown in Figure 7. To represent the images, a color histogram is created for 14 regions. Each histogram has 11 bins (or zones) and were extracted from three scales of each image. This results in 462 features for each image. The Image Data used a block size of 22180 bytes (12 records per block).

Two hundred random samples were selected from each data set to used as

Figure 7: Sample Hemera Photo-Object images.



(a) (b) (c)

Figure 8: In (a) the number of basis dimensions were varied with no compression of the α values. In (b) twenty-five basis vectors were used on the LIRD data and the number of bits per basis dimension used in the compression of the α values were varied. In (c) one hundred basis vectors were used on the Image data and the number of bits per basis dimension used in the compression of the α values were varied.

query locations. A Gaussian kernel was used to create the initial feature space. The results of varying the basis dimensionality and the bits per dimension for the initial retrieval of the 10 nearest neighbors using KVA-File are presented in Figure 8. As can be seen from the graphs, increasing the number of basis vectors decrease the average number of data blocks read by creating a better representation in feature space and thus a tighter upper and lower distance bounds. But it also increases the size of the approximation file. Decreasing the number of bits to represent a coefficient of a basis vector in an approximation increases the average number of data blocks read. But it also decreases the size of the approximation file. For examples, using seven bits per α (basis coefficient) and one hundred basis vector for the Image data results in an approximation file 4.8% the size of the original data file and the ten nearest neighbor search processed, on average, 2.2% of the original data file. Using four bits per α and twenty-five basis vector for the LIRD data set results in an approximation file 20.4% the size of the original data file and the ten nearest neighbor search processed, on average, 6.4% of the original data file.

The response to multiple iterations of relevance feedback is presented in Figure 9. Both an M-Tree and an KVA-File were created for the Image data set. Each retrieval returned a set of 20 images for the next iteration of relevance

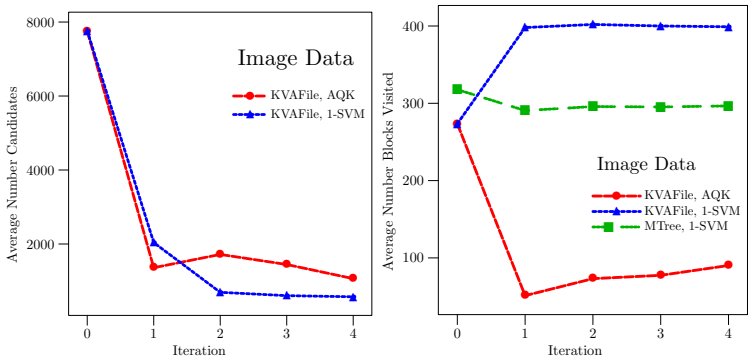


Figure 9: A KVA-File was built for the Image data using seventy-five basis vectors and eight bits per basis dimension. Part (a) contains the average number of candidates at the end of the first phase of KVA-File over the iterations of relevance feedback using 1-SVM and AQK. Part (b) contains the average number of blocks visited during the second phase KVA-File over the iterations of relevance feedback using 1-SVM and AQK.

feedback. Both AQK and 1-SVM were evaluated for KVA-File. Only 1-SVM was evaluated for M-Tree since AQK changes the distance metric between iterations and thus invalidates the M-Tree index structure. The KVA-File used 75 basis dimensions with eight bits per dimension. Thus the approximation file was 4% of the original data file. Both 1-SVM and AQK have a dramatic decrease in the average number of phase I candidates (7761 to 2039 and 7761 to 1368, respectfully) but 1-SVM increase block I/O and AQK decreased block I/O over feedback iterations. Thus just as in the input space approach [27], reusing the information from a previous iteration to set an initial bounds for phase one filtering does greatly reduce the size of the candidate list.

In terms of computational cost, M-Tree averages about 18000 distance calculations per iteration. Each distance calculation may involve multiple kernel evaluations when using 1-SVM or AQK. The average distance calculations for KVA-File are much lower since the approximation file is processed without the need for kernel evaluations. After the initial iteration, KVA-File averaged about 4800 distance calculations for 1-SVM and about 800 distance calculations for AQK.

6 Conclusion

Many data partitioning index methods perform poorly in high dimensional space. The VA-File approach overcomes the difficulty of high dimensional vector spaces, but can not be applied when using a distance metric in the feature space associated with a kernel. This paper presents KVA-File as an extension of

VA-File for kernel-based methods. An efficient approach to approximate vectors in feature space is presented with the corresponding upper and lower distance bounds.

This approach provides two levels of data compression. The first is in the selection of the number of basis vectors. This may be less than the original dimensionality of the input space. The second level of data compression is in the number of bits to represent the coefficients of an approximated vector. Both components depend on data distribution and the ability of the kernel to capture key components of that distribution. Experimental results on image data sets with high dimensionality demonstrated a computational and I/O efficiency improvement of nearest neighbor search using kernel distances.

Acknowledgements

The authors gratefully acknowledge support from the National Science Foundation under Grant No. 0136348.

References

[1] Amari, S. and S. Wu: 1999, 'Improving support vector machine classifiers by modifying kernel functions'. *Neural Networks* **12**(6), 783–789.

[2] Bach, F. R. and M. I. Jordan: 2002, 'Kernel Independent Component Analysis'. Journal of Machine Learning Research,**3**, 1-48.

[3] Chen, Y., X. Zhou, and T. Huang: 2001, 'One-Class SVM for Learning in Image Retrieval'. In: *Proceedings of IEEE ICIP, Thessaloniki, Greece*. pp. 815–818.

[4] Ciaccia, P., M. Patella, and P. Zezula: 1997, 'M-tree: an efficient access method for similarity search in metric spaces'. In: *Proceedings of the International Conference on Very Large Databases*. pp. 426–435.

[5] Cristianini, N. and J. Shawe-Taylor: 2000, *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press.

[6] Cristianini, N., J. Shawe-Taylor, and H. Lodhi: 2001, 'Latent Semantic Kernels'. In: C. Brodley and A. Danyluk (eds.): *Proceedings of ICML-01, 18th International Conference on Machine Learning*. Williams College, US, pp. 66–73.

[7] Ferhatosmanoglu, H., E. Tuncel, D. Agrawal, and A. Abbadi: 2000, 'Vector approximation based indexing for non-uniform high dimensional data sets'. In: *Proc. of ACM International Conf. on Information and Knowledge Management*. pp. 202–209.

[8] Guttman, A.: 1984, 'R-tree: a dynamic index structure for spatial searching'. In: *Proceedings of ACM SIGMOD International Conference on Management of Data*. pp. 47–47.

[9] Heisterkamp, D., J. Peng, and H. Dai: 2001, 'An Adaptive Quasiconformal Kernel Metric for Image Retrieval'. In: *Proceedings of IEEE CVPR, Kauai Marriott, Hawaii*. pp. 236–243.

[10] Ishikawa, Y., R. Subramanya, and C. Faloutsos: 1998, 'MindReader: Querying Databases Through Multiple Examples'. In: *Proceedings of 24th International Conference on Very Large Data Bases*. pp. 218–227.

[11] Manevitz, L. M. and M. Yousef: 2001, 'One-class SVMs for document classification'. *Journal of Machine Learning Research* **2**, 139–154.

[12] Muller, K., S.Mika, G. Ratsch, K. Tsuda, and B. Scholkopf: 2001, 'An introduction to kernel-based learning algorithms'. *IEEE Transactions on Neural Networks* **12**(2), 181–201.

[13] Murphy, P. and D. Aha, 'UCI repository of machine learning databases'. www.cs.uci.edu/∼mlearn/MLRepository.html.

[14] Peng, J., B. Banerjee, and D. R. Heisterkamp: 2002a, 'Kernel Index for Relevance Feedback Retrieval in Large Image Databases'. In: *9th International Conference on Neural Information Processing*.

[15] Peng, J., B. Bhanu, and S. Qing: 1999, 'Probabilistic Feature Relevance Learning for Content-Based Image Retrieval'. *Computer Vision and Image Understanding* **75**(1/2), 150–164.

[16] Peng, J., D. R. Heisterkamp, and H. K. Dai: 2002b, 'Adaptive Kernel Metric Nearest Neighbor Classification'. In: *Proceedings of IEEE International Conference on Pattern Recognition*. Quebec City, Canada.

[17] Rui, Y. and T. Huang: 2000, 'Optimizing Learning in Image Retrieval'. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina*. pp. 236–243.

[18] Scholkopf, B. and etal: 1999, 'Input space versus feature space in kernel-based methods'. *IEEE Transactions on Neural Networks* **10**(5), 1000 –1017.

[19] Scholkopf, B., A. Smola, and K.-R. Muller: 1998, 'Nonlinear component analysis as a kernel eigenvalue problem'. *Neural Computation* **10**, 1299–1319.

[20] Scholkopf, B. and A. J. Smola: 2002, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press.

[21] Tax, D. and R. Duin: 1999, 'Data domain description by support vectors'. In: *Proceedings of ESANN*. pp. 251–256.

[22] Tešić, J. and B. Manjunath: 2003, 'Nearest Neighbor Search for Relevance Feedback'. In: *Proceedings of IEEE Computer Society on Computer Vision and Pattern Recognition*, Vol. II. pp. 643–648.

[23] Tipping, M. E.: 2000, 'Sparse Kernel Principal Component Analysis'. In: *Advances in Neural Information Processing Systems*. pp. 633–639.

[24] Vapnik, V. N.: 1998, *Statistical learning theory*, Adaptive and learning systems for signal processing, communications, and control. New York: Wiley.

[25] Webber, R., J. Schek, and S. Blott: 1998, 'A quantitative analysis and performance study for similarity-search methods in high-dimensional space'. In: *Proceedings of the International Conference on Very Large Databases*. pp. 194–205.

[26] Weber, R. and S. Blott: 1997, 'An Approximation-Based Data Structure for Similarity Search'. Technical Report 24, ESPRIT project HERMES (no. 9141). Available at http://www-dbs.ethz.ch/∼weber/paper/HTR24.ps.

[27] Wu, P. and B. Manjunath: 2000, 'Adaptive nearest neighbor search for relevance feedback in large image databases'. In: *Proceedings of the ACM Multimedia*. pp. 202–209.

[28] Yianilos, P.: 1992, 'Data structures and algorithms for nearest neighbor search in general metric spaces'. In: *the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*. pp. 311–321.

[29] Zhou, X. and T. Huang: 2001, 'Small Sample Learning during Multimedia Retrieval using BiasMap'. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Kauai Marriott, Hawaii*, Vol. 1. pp. 11–17.