Kernel VA-Files for Nearest Neighbor Search in Large Image Databases

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

Jing Peng
Department of Electrical Engineering
and Computer Science
Tulane University
New Orleans, LA 70118
Email: jp@eecs.tulane.edu

Abstract— **Many data partitioning index methods perform poorly in high dimensional space. The vector approximation file (VA-File) approach overcomes some of the difficulties of high dimensional vector spaces, but cannot be applied when using a kernel distance metric in the data measurement space. This paper introduces a novel KVA-File (kernel VA-File) that extends VA-File to kernel-based retrieval methods. A key observation is that kernel metrics may be non-linear in the input data space but is still linear in an induced feature space. It is this linear invariance in the induced feature space that enables KVA-File to work with kernel metrics. 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 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.**

I. Introduction

Content-based image retrieval is an active area of research. Kernel methods have been exploited in classification regression, and information retrieval [6], [13]. Kernel methods for image retrieval have been presented and shown to outperform weighted Euclidean distances [4], [8]. But they rely on a sequential scan of data. The goal of this work is to provide an efficient indexing method usable by kernel methods.

The curse of dimensionality [3] is a problem facing many fields. Traditional data partitioning approaches to indexing become inefficient as the number of dimensions increase, eventually taking longer time than a sequential scan of the data. A vector approximation file (VA-File) takes a signature or filter approach to indexing data [17]. 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. To be able to conduct the filtering, upper and lower bounds on the distance from the query to the data point needs to be calculated. This calculation, however, is invalid for kernel-based approaches. This paper presents KVA-File that allows efficient calculation of upper and lower distance bounds in the input data space, thus allowing the VA-File approach to work for kernel-based methods.

II. Kernel Distance for Relevance Feedback

The kernel trick has been applied to numerous problems [6], [9]. The kernel allows an algorithm to work in a 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 [6], [16]. With $\mathbf{a}$ and $\mathbf{b}$ in the input space, the squared feature space distance is

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

Two known kernel distances in the relevance feedback literature are Adaptive Quasiconformal Kernel (AQK) [8] and One-Class SVM (1-SVM) [4], [14]. We briefly present each approach.

Adaptive Quasiconformal Kernel (AQK) distance [8] combines the kernel distance (1) with a quasiconformal mapping [1], $\tilde{k}(\mathbf{a},\mathbf{b}) = c(\mathbf{a})c(\mathbf{b})k(\mathbf{a},\mathbf{b})$, to create a new kernel distance:

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

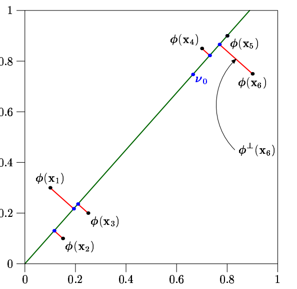
where $c(\mathbf{a})$ is a positive real valued function of $\mathbf{a}$ in the input space. One can select $c(\mathbf{a})$ from relevance feedback to expand the spatial resolution around irrelevant samples and contract the spatial resolution around relevant samples [8]. That is, distance to irrelevant samples from the query is increased and distance to relevant samples are decreased.

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 [4], [14]. After finding the center, $\mathbf{c} = \sum_i \gamma_i \boldsymbol{\phi}(\mathbf{x}_i)$, the one-class SVM kernel distance of vector $\mathbf{z}$ to $\mathbf{c}$ is

$$\text{dist}(\mathbf{z},\mathbf{c})^2 = k(\mathbf{z},\mathbf{z}) - 2\sum_i \gamma_i k(\mathbf{x}_i,\mathbf{z}) + \sum_{i,j} \gamma_i \gamma_j k(\mathbf{x}_i,\mathbf{x}_j)\,. \tag{3}$$

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

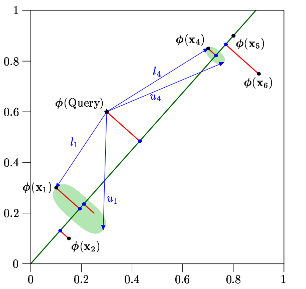
$$\min_{R,\Xi,c} R^2 + \frac{1}{vl}\sum_i \xi_i \quad \text{subject to } ||\boldsymbol{\phi}(\mathbf{x}_i)-c||^2 \leq R^2+\xi_i, \quad \xi_i \geq 0.$$



(a) Feature Space generated by an Identity Kernel. A one dimension basis is generated by $\boldsymbol{\nu}_0$.

	$\boldsymbol{\phi}(\mathbf{x})$		KVA-File α_0	$\boldsymbol{\phi}^\perp(\mathbf{x})$
1	0.1	0.3	0.29	0.015
2	0.15	0.1	0.17	0.0020
3	0.25	0.2	0.32	0.0029
4	0.7	0.85	1.1	0.0017
5	0.8	0.9	1.2	0.0
6	0.9	0.75	1.2	0.030
Query	0.3	0.6	0.65	0.03

(b) Data is decomposed into the basis and remaining component, e.g., $\boldsymbol{\phi}(\mathbf{x}_6) = \alpha_6\boldsymbol{\nu}_0 + \boldsymbol{\phi}^\perp(\mathbf{x}_6)$.



(c) Upper and lower distance bounds

Fig. 1. KVA-File approximations

III. Kernel Index

There are a number of metric space based index methods, such as M-Trees [5], any of which can be constructed in the feature space to give rise to a kernel index scheme. In [11], M-Trees were used to create a kernel index scheme. For approaches that do not modify the kernel, such as 1-SVM, M-Tree can be an effective kernel indexing scheme. But if in the learning process, the kernel is modified, such as in AQK, the index structure of an M-Tree can no longer be used. This motivations our investigation into a kernel vector approximation file that can be applied when relevance feedback dynamically changes the kernel.

A. Vector Approximation File

The vector-approximation file (VA-File) uses a signature file as a filter [17]. The signature file is a compressed approximation of the original data file. Each data vector is stored in the approximation file as the bit encoding of the hypercube in which it lies. The hypercubes are generated by partitioning each data dimension into the number of bins representable by the number of bits used for that dimension. Typically, the compressed file is 10% to 15% of the size of the original data file. 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.

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.

B. Kernel Vector Approximation File

The approximation file is a reduced representation of the data that allows an efficient calculation of upper and lower distance bounds. 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 [12] and a Gram-Schmidt approach [2], [7] 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, but it is not a compact representation. 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 [15] may also be useful.

A method to find an orthogonal basis and create data approximations is presented in Figure 2. For now, we assume a reduced set of basis vectors, $\boldsymbol{\nu}_t$, of the feature space is available and that a point on feature space can be decomposed into a linear combination of basis vectors and a component, $\boldsymbol{\phi}^\perp(\mathbf{x})$ that is orthogonal to the basis. With this decomposition, we can represent points $\mathcal{P}$ and $\mathcal{Q}$ as $\mathbf{P} = \boldsymbol{\phi}(\mathbf{x}_p) = \boldsymbol{\phi}^\perp(\mathbf{x}_p) + \sum_{t=0}^{d-1} \alpha_t\boldsymbol{\nu}_t$ and $\mathbf{Q} = \boldsymbol{\phi}(\mathbf{x}_q) = \boldsymbol{\phi}^\perp(\mathbf{x}_q) + \sum_{t=0}^{d-1} \beta_t\boldsymbol{\nu}_t$ where d is the number of basis vectors. Distance between $\mathcal{P}$ and $\mathcal{Q}$ can then be expressed as

$$\begin{aligned}\text{dist}(Q,P)^2 = \boldsymbol{\phi}^\perp(\mathbf{x}_q)^T\boldsymbol{\phi}^\perp(\mathbf{x}_q) + \boldsymbol{\phi}^\perp(\mathbf{x}_p)^T\boldsymbol{\phi}^\perp(\mathbf{x}_p) \\ - 2\boldsymbol{\phi}^\perp(\mathbf{x}_q)^T\boldsymbol{\phi}^\perp(\mathbf{x}_p) + \sum_{t=0}^{d-1}(\alpha_t - \beta_t)^2. \quad (4)\end{aligned}$$

We approximate a point in feature space by the basis coefficients, α, and the magnitude of the error, $\sqrt{\boldsymbol{\phi}^\perp(\mathbf{x})^T\boldsymbol{\phi}^\perp(\mathbf{x})}$. With the approximation of points $\mathcal{P}$ and $\mathcal{Q}$, the unknown term in (4) 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. The angle is not represented in the approximation but the extremes can be used to generate bounds on the distance. Using the notation of $\hat{\mathbf{G}}_d(p,p) = \boldsymbol{\phi}^\perp(\mathbf{x}_p)^T\boldsymbol{\phi}^\perp(\mathbf{x}_p)$ and $\hat{\mathbf{G}}_d(q,q) = \boldsymbol{\phi}^\perp(\mathbf{x}_q)^T\boldsymbol{\phi}^\perp(\mathbf{x}_q)$, then the upper distance bounds, $\mathcal{U}(Q,P)$, and the lower distance bounds, $\mathcal{L}(Q,P)$, is

$$\mathcal{U}(Q,P) =$$

$$\left(\hat{\mathbf{G}}_d(q,q) + \hat{\mathbf{G}}_d(p,p) + 2\sqrt{\hat{\mathbf{G}}_d(q,q)}\sqrt{\hat{\mathbf{G}}_d(p,p)} + \sum_{t=0}^{d-1}(\alpha_t - \beta_t)^2\right)^{\frac{1}{2}}$$

$$\mathcal{L}(Q,P) =$$

$$\left(\hat{\mathbf{G}}_d(q,q) + \hat{\mathbf{G}}_d(p,p) - 2\sqrt{\hat{\mathbf{G}}_d(q,q)}\sqrt{\hat{\mathbf{G}}_d(p,p)} + \sum_{t=0}^{d-1}(\alpha_t - \beta_t)^2\right)^{\frac{1}{2}}$$

The ranges for each α_i and for $\hat{\mathbf{G}}_d(p,p)$ can be partitioned into bins and the bin locations represented by a bit encoding. Thus allowing two levels of compression: the number of basis dimensions and the number of bits per dimension.

The upper and lower bounds are illustrated in Figure 1. The vector $\boldsymbol{\nu}_0$ is used to create a one dimensional basis. The data is decomposed into the basis and an orthogonal component, e.g., $\boldsymbol{\phi}(\mathbf{x}_6) = 1.2\boldsymbol{\nu}_0 + \boldsymbol{\phi}^{\perp}(\mathbf{x}_6)$. The upper and lower distance bounds is the extremes of the distance of a query to the hyperdisk centered at the location specified by the basis coefficients and with the radius of the magnitude of the orthogonal component.

The distance bounds for 1-SVM and AQK can be developed in a similar manner. The upper and lower distance bounds for AQK is

$$\mathcal{U}(Q,P) = \left(c(\mathbf{x}_q)^2\hat{\mathbf{G}}_d(q,q) + c(\mathbf{x}_p)^2\hat{\mathbf{G}}_d(p,p)\right.$$

$$\left. + 2c(\mathbf{x}_q)c(\mathbf{x}_p)\sqrt{\hat{\mathbf{G}}_d(q,q)}\sqrt{\hat{\mathbf{G}}_d(p,p)} + \sum_{t=0}^{d-1}(c(\mathbf{x}_p)\alpha_t - c(\mathbf{x}_q)\beta_t)^2\right)^{\frac{1}{2}}$$

$$\mathcal{L}(Q,P) = \left(c(\mathbf{x}_q)^2\hat{\mathbf{G}}_d(q,q) + c(\mathbf{x}_p)^2\hat{\mathbf{G}}_d(p,p)\right.$$

$$\left. - 2c(\mathbf{x}_q)c(\mathbf{x}_p)\sqrt{\hat{\mathbf{G}}_d(q,q)}\sqrt{\hat{\mathbf{G}}_d(p,p)} + \sum_{t=0}^{d-1}(c(\mathbf{x}_p)\alpha_t - c(\mathbf{x}_q)\beta_t)^2\right)^{\frac{1}{2}}.$$

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 $c(\mathbf{x}_p)$.

C. Orthogonalization in Feature Space

An incremental Gram-Schmidt orthogonalization algorithm is presented in Figure 2. A basic summary of the approach is: select a vector; convert all other vectors into their component that is orthogonal to the selected vector; repeat by selecting a new vector for the remaining set.

The storage requires for the Gram matrix is $\mathcal{O}\left(n^2\right)$ where n is the size of the data set. Even a moderate size data set would require external storage of the Gram matrix. Instead of storing the components of the Gram matrix, only the current approximation of each vector is needed. This algorithm is similar to the incremental algorithm presented in [2]. Finding a basis for large data sets is feasible with the incremental algorithm. New data can be approximated with a similar algorithm.

```
Select first basis vector corresponding to data vector $\mathbf{x}_s$.
for each data vector $\mathbf{x}_p$ do
    $\alpha_0 = \frac{k(\mathbf{x}_p,\mathbf{x}_s)}{\sqrt{k(s,s)}}$
    $\hat{\mathbf{G}}_0(p,p) = k(p,p) - \alpha_0^2$
end-for
$t = 0$
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 $\beta_i$'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 $\beta_i$'s for later use.
    increment $t$
end-while
```

Fig. 2. Incremental Orthogonalization Algorithm

IV. Experimental Results

In the following, we use block I/O to compare nearest neighbor search using kernel indexing methods (MTree and KVAFile) over kernel-based relevance feedback approaches (1-SVM and AQK) on the following two real data sets.

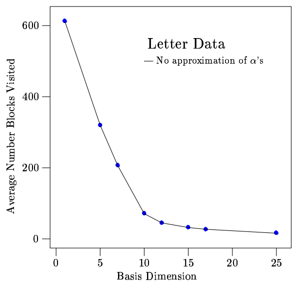
LIRD: The Letter image database, taken from [10], is the **Letter Image Recognition data** (LIRD) data set and consist of 20000 letter images. Each image is represented by 16 numerical features. The LIRD data used block size of 1988 bytes (31 records per block).

Image Data: The Hemera Photo-Object image data set consists of 94800 images that are very heterogeneous and having annotated ground truth. To represent the images, a color histogram is created for 14 regions. Each histogram has 11 bins (os 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).

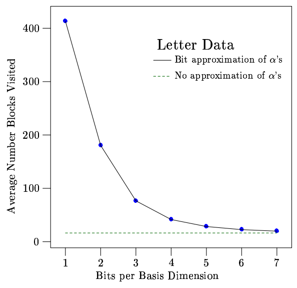
Two hundred random samples were selected from each data set to used as query locations. A Gaussian kernel was used to create the initial feature space.

The results of varying the basis dimensionality and the bits per dimension for the initial retrieval of the 10 nearest neighbors using KVA-File is presented in Figure 3. 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.

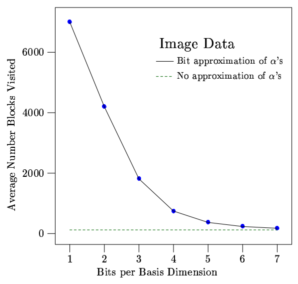
Both an M-Tree and an KVA-File was created for the Image data set. Each retrieval returned a set of 20 images for the next iteration of relevance feedback. Both AQK and 1-SVM were evaluated for KVA-File. Only 1-SVM was evaluated for M-Tree since AQK changes the distance metric between iterations and thus invalidates the M-Tree index structure. The results are presented in Figure 3(d). The KVAFile used 75



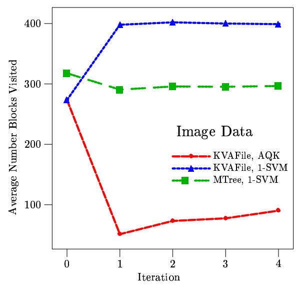
(a)



(b)



(c)



(d)

Fig. 3. Block I/O performance. The number of basis dimensions were varied in (a) with no compression of the α values. The number of bits per basis dimension used in the compression of the α values was varied in (b) and (c). Twenty-five basis vectors were used in (b). One hundred basis vectors were used in (c). Seventy-five basis vectors and eight bits of approximation were used for KVA-File in (c).

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.

In terms of CPU cost, M-Tree average about 18000 distance calculations per iteration. Each distance calculation may involved multiple kernel evaluations when using 1-SVM or AQK. The average distance calculations for KVA-File is 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.

V. 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 proposes a 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.

Acknowledgment

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

References

[1] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[2] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," University of California, Berkeley, Tech. Rep. UCB//CSD-01-1166, 2001.

[3] R. E. Bellman, *Adaptive Control Processes*. Princeton Univ. Press, 1961.

[4] Y. Chen, X. Zhou, and T. Huang, "One-class svm for learning in image retrieval," in *Proceedings of IEEE ICIP, Thessaloniki, Greece*, October 2001, pp. 815–818.

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

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

[7] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, "Latent semantic kernels," in *Proceedings of ICML-01, 18th International Conference on Machine Learning*, C. Brodley and A. Danyluk, Eds. Williams College, US: Morgan Kaufmann Publishers, San Francisco, US, 2001, pp. 66–73.

[8] D. Heisterkamp, J. Peng, and H. Dai, "An adaptive quasiconformal kernel metric for image retrieval," in *Proceedings of IEEE CVPR, Kauai Marriott, Hawaii*, 2001, pp. 236–243.

[9] K. Muller, S.Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–201, March 2001.

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

[11] J. Peng, B. Banerjee, and D. R. Heisterkamp, "Kernel index for relevance feedback retrieval in large image databases," in *9th International Conference on Neural Information Processing*, 2002.

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

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

[14] D. Tax and R. Duin, "Data domain description by support vectors," in *Proceedings of ESANN*, 1999, pp. 251–256.

[15] M. E. Tipping, "Sparse kernel principal component analysis," in *Advances in Neural Information Processing Systems*, 2000, pp. 633–639. [Online]. Available: citeseer.nj.nec.com/article/tipping01sparse.html

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

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