Bootstrap Sequential Projection Multi Kernel Locality Sensitive Hashing

Harsham Mehta, Deepak Garg
Department of Computer Science and Engineering
Thapar University
Patiala, India -147004
Email: harsham1990@gmail.com, dgarg@thapar.edu

***Abstract*—In Recommender system we have similarity search as a key part for making efficient recommendations. Similarity search have always been a tough task in a high dimensional space. Locality Sensitive Hashing which is most suitable for extracting data in a high dimensional data (Multimedia data). The Idea of locality sensitive hashing is that it decreases the high dimensional data to low dimensions using distance functions and then store this data using hash functions which ensures that distant data is placed much further. This technique has been extended to kernelized Locality sensitive hashing (KLSH). One limitation of regular LSH is they require vector representation of data explicitly. This limitation is addressed by kernel functions. Kernel functions are capable of capturing similarity between data points. KLSH is a breakthrough in content based systems. This method takes a kernel function, a high dimensional database for data inputs and size of hash functions to be built. These kernel functions that are being used may give different degree of result precision. Hence we try to combine these kernels with a bootstrap approach to give an optimal result precision. In this paper we present the related work that has been done in locality sensitive hashing and at the end we propose algorithms for data preprocessing and query evaluation.**

I. Introduction

Recommender System has been a domain of interest since first paper on collaborative filtering emerged in 1995. It constitutes problem rich research areas and huge no. of practical application that help users to deal with information overload and provide services for real-life application. The problem is of estimating ratings of the items that have not been rated by the users. The current generation of recommender system requires much more improvements to make recommendations methods more effective and applicable to much wider scope like recommending vacations, certain financial services or purchasing products while shopping for example smart. The problem here lies within the fact that dataset available is high dimensional. As the profile of products to be recommended and the profile of people to whom it is to be recommended must contain a lot of information so as to be able to served in a better way, thus it is supposed to be high dimensional. Various data structures have been proposed and implemented for indexing and processing data points [1], [2], but their efficiency is limited to low dimensional data. They become less efficient (time and space complexity increases exponentially) as the dimensionality of data increases. This is depicted as Curse of Dimensionality. To tackle this problem, recent studies focuses on approximation approaches. Instead of finding nearest neighbor p from query q approximation approach returns all data points in $(1+\epsilon)$ time distance between p and q. Locality sensitive hashing is a well known technique in this direction which has been applied to numerous applications. The idea is to reduce their dimensionality and then hash all the similar data points together according to a similarity criterion. The basic LSH technique requires data points in vector form. KLSH Kernelized LSH which is an extension to LSH does not require explicit vector representation. But KLSH only adopts a single kernel which limits it to use of only particular feature descriptors. Thus multiple kernels are introduced in LSH and with it different techniques are acquired for combining these multiple kernels for optimal result [3].The objective is to find optimal combination of different kernels, to get more approximate result. To this end we propose our technique Bootstrap Sequential Projection Multi Kernel Locality Sensitive Hashing (BSP-MKLSH) for combining multiple kernels through an efficient learning method. The rest of the paper is divided into following sections. Section 2 contains review related work. Section 3 reviews previously implemented algorithm KLSH. Section 4 describe Bootstrap sequential projection Multi- kernel Locality Sensitive Hashing. Section 5 gives experimental result. Section 6 concludes our work.

II. Related Work

Our literature work focus on applications for image retrieval. Spatial data structures like kd trees were used to handle NN problem [4]–[6].But they scale poorly with data dimensionality and perform like exhaustive linear search. So Locality Sensitive Hashing (LSH) was proposed [7] to tackle this curse of high dimensionality. Instead of solving exact similarity search for high dimensional indexing, recent studies follow approximate indexing for high dimensional data. The LSH technique is basically of two types: linear projection method and kernel based methods.In linear projection method piotr indyk et.al. [7] first introduced locality sensitive hashing in 1998. He introduced for $\{0,1\}^d$ Hamming metric space of dimension d. For an input dataset $X = R^d$. If PXcontains n data points i.e. $(p_1, p_2, ..., p_n)$.Distance of any point p to the rest of the set could represented as $d(p,P) = \min_{q \in P} d(p,q)$ where $d(p,q)$ is Euclidian distance. A family of hash function defined as $\aleph$ is calculated using distance $d(p,P)$. The result contains all data points up to distance $1+\epsilon$ from query. The complexity for this technique for ϵ-approximate result is $O(dn + n^{1+1/\epsilon})$ space and $O(n^{1/\epsilon})$ hash function evaluations for each query and $O(d)$ operations for evaluating hash functions. Venu satuluri et.al. [8] introduced two phases in similarity search phase. First candidate generations phase and second candidate verification phase. Junhao gan et. al. [9]

978-1-4799-3080-7/14/$31.00

Gave another variant of LSH which is currently, the primary choice of constructing an LSH function for Euclidean distance. Exact Euclidean LSH (E2LSH)exploits LSH as : a set of k LSH functions $h_1, h_2, ..., h_k$ are randomly chosen from LSH family then they are joined together to form a compound hash function $G(o) = (h_1(o), ..., h_k(o))$ for any object o.

But our research is more based on second type i.e. kernel based method. Kernel locality sensitive hashing [10] in 2009 was proposed to overcome regular LSH limitation. KLSH does not require data to be in multidimensional vector space and only needs to know the underlying embedding of data. It explores a powerful framework that uses kernel/similarity function. Its variants have been used that force the hamming distance between the binary codes of two vectors is related to a shift invariant kernel value [11]. Recently this type has been upgraded to use of Multikernels. In [3] it has been shown that different kernels will lead to very different approximation errors. Hence some kernel functions may have good retrieval accuracy and others may have better approximation performance. Thus including combination of kernel functions could provide good approximation and better retrieval accuracy. But for that we need to make a good tradeoff between retrieval accuracy and approximation error. For this different kernel functions should be weighted accordingly. A simple approach is to allocate equal weights to the all the kernels. Such approach will not fully explore power of kernel function. To address this limitation various algorithm have already been proposed. Multi Kernel LSH [12] proposes bit allocation optimization step for different kernels. For m kernels, we allocate $b_1, b_2, b_3, ..., b_m$ bits to respected kernels where $b = b_1 + b_2 + b_3 + ... + b_m$. It assigns same no. of bits to each kernel . [3] also introduces 2 new techniques namely WMKLSH and BMKLSH.

Weighted Multi Kernel Locality Sensitive Hashing WMKLSH uses a supervised learning approach for determining allocation of bit size. A kernel is assigned bits size (weight) based on similarity level captured by it between data points. Training set consist of small set of queries and their relevance judgment. First retrieve performance for a query of each kernel function then calculate mean Average Performance (mAP) for each query. Finally bit sizes of a kernel are allocated according to their weight of mAP. Another extent to this approach was proposed in [3] Boosting Multi Kernel Locality Sensitive Hashing BMKLSH which introduced the concept of boosting rounds [13] for learning of the bit size of each kernel. The optimization is done according to the following problem:

$$\max_{b_1,...,b_m \in |b_m|} \sum_{i=1}^{n_q} \exp \sum_{l=1}^{m} AP_l(t) subjectto \sum_{l} b_l = b \quad (1)$$

A bootstrap learning algorithm is introduced in [14] that performs regularized learning based hashing. The paper uses learning technique which corrects the error effectively from holistically learned bits in previous projections with no computational overhead. Table 1 shows some of the notations that are used in this paper.

Symbol	Meaning
X	Matrix of data points of size dxn, where d is dimensionality and n is no. of images.
P1	Subset of X containing r elements, $r < n$.
$ker(x_i, x_j)$	An RBF kernel function that defines similarity level between x_i and x_j.
h_i	Hash function for i^{th} bit.
$\phi(x)$	Feature set for input x.
K	Kernel matrix made from kernel function.
b	Total no. of hash bits.
$d(x_i, x_j)$	Eucledian distance.
U	Vector containing cluster centers.
G	Vector of length m containing mAP of ith kernel gi where i=1,2,m.
θ	Standard deviation.
Z	Regression matrix.
W	Eigen projection matrix.
H	Hamming matrix.
$\aleph$	Family of hash functions

TABLE I. SYMBOLS USED.

III. ANALYSIS OF KLSH

In this section we first try to review the previously implemented KLSH [10]. Our data points are represented as a feature matrix. Let $X = \Re^d$ be a $d \times n$ matrix containing features for n data points. The algorithm efficiently computes the kernel matrix K from a kernel function $ker(x_i, x_j)$, for a sampled data points that are taken from the original dataset X. This kernel matrix acts as a similarity measure for high dimensional data points. Then LSH projects all the data points to hash keys $h_1, h_2, h_3, ..., h_b$ which is a low dimensional binary space where b is the length of the hash keys. These hash keys are being produced using kernel matrix K. Now when a query $q \in \Re^d$ arrives to find s similar results or nearest neighbor, this query is also projected on b hash keys using LSH through which we find s most approximate nearest neighbor. The KLSH function [10] first of all creates a set $p1$, contains r random data points from the dataset of n data points. $p1$ subset of $X = x_1, x_2, ..., x_r$. These r data points are then applied on kernel function $ker(x_i, x_j)$ to produce a kernel matrix K of normalized form. Where i,j=1,2,....,r. $ker(x_i, x_j)$ calculates the inner product of x_i, x_j and maps data points x_i, into a functional space. This $ker(x_i, x_j)$ maps data point x_i, through a nonlinear feature mapping function $\phi(x_i)$. This function satisfies the condition $\phi^T(x_i)\phi(x_i) = ker(x_i, x_j)$. After that it generates b random vectors relating to the subset $p1$ as $e_1^{p1}, e_2^{p1}, ..., e_b^{p1}$. Calculate vector $W_y = K^{-1/2} e_y^{p1}$. We will have $W_y = w_y^1, w_y^2, ..., w_y^r$ for $y = 1, 2, ..., b$ where $K^{-1/2}$ is calculated from svd of K which gives U, V, S matrices and then calculating $K = UVU^T$ and $K^{-1/2} = UV^{-1/2}U^T$. From $K^{-1/2}$ we derive hash function

$$h_y(\phi(x)) = sign(\sum_{j=1}^{r} w_y^j ker(x_i, x_j))$$

where $j = 1, 2, ..., r$. The detailed explanation of the procedure is in algorithm 1. The trick for making a better implementation of KLSH is by making $r \to n$ i.e. the sample set is same as the original dataset. The matrix produced is of order $n \times n$ called the gram matrix This would make the approximation exactly equal to the query. Hence the to achieve high precision make $r \to n$. but this would make the calculation more complex and hence increase the complexity hence the value of r must be decided appropriately. [3] discusses the analysis of this algorithm by providing a theorem that shows kernel similarity is well approximated by $\frac{p}{tb} h_i^T h_j$. The error decreases as the no. of bits b increases.

Algorithm 1 KLSH

Require: :A database with n images :$x_i : i = 1, 2, ..., n, x_i \in \Re^d$.
Length of the hash key: b.
A kernel function $ker(.,.)$.

Ensure: A set of b hash function $\aleph = \{h_k | k = 1, ..., b\}$.

1: Select r random data points $p1 = x1, x2, ..., xr$.
2: Calculate distance matrix D as $d_{i,j} = mind(x_i, x_j)$ for $j = 1, ..., p, j \neq i$.
3: Calculate standard deviation σ as $\sigma = \sqrt{\frac{\sum x - \overline{x}^2}{n}}$.
4: Calculate kernel matrix K as $K = [ker(x_i, x_j)]_(p \times p)$.
5: **for** $l = 1, ..., b$ **do**
6: Form e^l by selecting t indices at random from $[1, ..., p]$ as $e^l = e^l_1, e^l_2, ..., e^l_t$.
7: Form w^l as $K^{-1/2}e^l$ where $U.V.S = svd(K)$ and $K^{-1/2} = UV^{-1/2}U^T$ i.e. w^l will be a vector of length p.
8: Construct hash function as $h_l(\phi(x)) = sign(\sum_{j=1}^{r} w^j_l ker(x_i, x_j))$.
9: **end for**

IV. Bootstrap sequential projection multi kernel locality sensitive hashing

There are various methods to combine multiple kernel as we discussed earlier. To make multiple kernel matrices, we use different feature sets to define input matrix X and apply RBF kernel functions. The different feature set used are mentioned in further section. We propose our technique for combining multiple kernel in a more efficient way. Bootstrap sequential projection multi kernel locality sensitive hashing (BSPMKLSH). This algorithm creates a difference in the error condition as compared to the former boosting approach. In [3] BMKLSH judges all the previous bits separately by using boosting technique [13]. But the latter judges all previous bits holistically i.e. error will occur with either positive pairs with large hamming distance or negative pairs with small hamming distance. αk and βk are threshold to determine large and small hamming distance after k projections. Our objective is to learn the optimal allocation of bit size to the kernel function. We use annotations described in table 2.

A. Bootstrap sequential projection Learning

A clustering method is firstly performed on the similarity distance of all inputs to obtain array of cluster centers U

$$U = \{u_i \in \Re^d\}_i^r \tag{2}$$

that act as anchors. Then the anchor graph defines the truncated similarities defined by Z_{ij}, between r anchors and n data points.

$$Z_{ij} = \begin{cases} \dfrac{\exp - \dfrac{d^2(u_i, g_j)}{\theta}}{\sum_{i' \in r} \exp - \dfrac{d^2(u'_i, g_j)}{\theta}} & \text{if } i \in r \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Eq (3) forms a Z matrix that measures the underlying similarities between raw samples and their corresponding data points. $Z(x)$ represents column of element x in regression matrix Z. By introducing hash functions using anchors creates a non linear hashing function that derives a projection matrix W. A relaxation is introduced through removing binary constraints to the following objective function:

$$\max_W \frac{1}{2} tr\{W^T Z S Z^T W\}. \tag{4}$$

To maximize the information provided by each bit a regularizer (given by eq (5))is added.

$$W^T Z Z^T W \tag{5}$$

After applying the relaxation we can solve the projection problem by directly obtaining the eigen values of Q where Q is defined as:

$$Q = ZSZ^T + \lambda ZZ^T. \tag{6}$$

Hence the error condition formulated as after k projections:

$$\begin{aligned} \frac{k}{2} - \sum_{t=1}^{k} sign(w_t^T Z(g_i) Z(g_j)^T w_t) > \alpha k \quad (g_i, g_j) \in P \\ \frac{k}{2} - \sum_{t=1}^{k} sign(w_t^T Z(g_i) Z(g_j)^T w_t) < \beta k \quad (g_i, g_j) \in P \end{aligned} \tag{7}$$

We represent the error conditions with matrix H^k with data points given as:

$$H^k_{ij} = \sum_{t=1}^{k} sgn(w_t^T z(g_i) z(g_j)^T w_t). \tag{8}$$

In matrix form we can write:

$$H^k = \sum_{t=1}^{k} sgn(Z^T w_t w_t^T Z). \tag{9}$$

H^k can be calculated recursively as:

$$H^k = H^{k-1} + sgn(Z^T w_t w_t^T Z). \tag{10}$$

S is a label matrix that defines the relationship between the elements x_i, x_j. In this case we use the similarity degree between these two elements as relationship.

$$S_{ij} = \begin{cases} 1 & \text{if } (g_i, g_j) \in P \\ -1 & \text{if } (g_i, g_j) \in N \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

S^1_{ij} represent a logical relationship of pair (x_i, x_j).

$$S^1_{ij} = \begin{cases} 1 & \text{if } d^2(g_i, g_j) - \alpha k < 0 \\ -1 & \text{if } d^2(g_i, g_j) - \beta k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Error condition can be rewritten w.r.t. S^1_{ij} and H^k as:

$$\begin{aligned} H^k_{ij} - \alpha k < 0, S^1_{ij} > 0 \\ H^k_{ij} - \beta k > 0, S^1_{ij} < 0 \end{aligned} \tag{13}$$

The updating rule for S is:

$$S^{k+1} = S^1 + \Delta S^k. \tag{14}$$

Algorithm 2 BSP-MK-LSH

Require: :mAP of all kernel functions from algorithm 1 in the form of vector G,
l = no. of projection iterations required ,
constant λ.

Initialize $H^0 = 0, C^1 = ZZ^T$.
Calculate S1 using equation 12.
Calculate Z using equation 3.
for $k = 1, ..., l$ **do**
  Calculate $Q = ZSZ^T + \lambda ZZ^T$.
  Form e as eigen vectors of Q^k set $w_k = e$.
  $H^k = H^{k-1} + sgn(Z^T w_t w_t^T Z)$.
  Calculate ΔS^k according to equation 15.
  $S^{k+1} = S^1 + \Delta S^k$.
  $U^k = I - w_k w_k^T$.
  $C^{k+1} = U^k C^k U^{k^T}$.
  $W = \{w_1, w_2, ..., w_l\}$.
end for
Calculate $b_i = \frac{\sum_l w_l^i}{\sum_{f=1}^{m} \sum_l w_l^f} * b$.

Algorithm 3 MK-LSH

Require: :A database with n images :$x_i : i = 1, 2, ..., n, x_i \in \Re^d$.
Bit allocation vector $[b_1, b_2, ..., b_m]$.
A set of m kernel function $k_l | l = 1, m$.
Ensure: A set of b hash function $\aleph = \{h_k | k = 1, ..., b\}$.

k=0.
Select r random data points $p1 = x1, x2, ..., xr$.
for y=1,.....,m **do**
  Calculate kernel matrix K_y as $K_y = [ker(x_i, x_j)]_(pp)$.
  for $l = 1, ..., b_y$ **do**
    Form e^l by selecting t indices at random from $[1, ..., p]$ as $e^l = e_1^l, e_2^l, ..., e_t^l$.
    Form w^l as $K_y^{-1/2} e^l$ where $U.V.S = svd(K_y)$ and $K_y^{-1/2} = UV^{-1/2}U^T$ i.e. w^l will be a vector of length p.
    Construct hash function as $h_l(\phi(x)) = sign(\sum_{j=1}^{r} w_l^j ker(x_i, x_j))$.
  end for
end for

ΔS^k is increased weight matrix which is determined by:

$$\Delta Sij^k = \begin{cases} \frac{\alpha k - H_{ij}^k}{2k} & \text{if } H_{ij}^k - \alpha k < 0, S_{ij}^1 > 0. \\ \frac{\beta k - H_{ij}^k}{2k} & \text{if } H_{ij}^k - \beta k > 0, S_{ij}^1 < 0. \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

After extracting k^{th} projections from Q^k , to minimize redundancy in bits, Z is updated as:

$$Z = (I - w_k w_k^T) Z. \quad (16)$$

We recursively calculate the update of ZZ^T to reduce computational effort. Since both C^k and U^k are with the size of $r \times r$, we can efficiently compute the residual. The detailed procedure is summarized in algorithm 2.

B. Combining Multiple Kernels

The value b_i learned from BSP-MK-LSH algorithm are then applied on Multi- kernel Locality sensitive Hashing [12](MK-LSH).The detailed procedure of MK-LSH is explained in algorithm 3.

V. Experiments

A. Test Bed

The dataset that we have used to examine our proposed algorithm, called Flickr [15] (known as MIRFLICKR-25000) has been widely used as a benchmark for scalable image retrieval in content based searching. This dataset contains 25,000 images maintained in groups, each having one query image. We implemented our algorithm in matlab language environment on a machine with configuration of 2gb ram intel i3 2.0 gHz processor.

B. Experimental Setup

We present our result in both complexity wise and in terms of percentage of data items searched with hashing function. We set parameter ρ to control the fractions of nearest neighbor to be linearly scanned using LSH. Next we calculate performance metric using mean average precision (mAP). Precision value is ratio of relevant examples over total retrieved examples. mAP is the mean of AP of all the queries.

C. Image Feature Extraction

We use 5 different feature extraction methods that are commonly used for describing image. All these methods are global feature descriptors. These are:
(1)GABOR filter.
(2)GIST features.
(3)Color histogram and color moments.
(4)Edge direction histogram.
(5)Local binary pattern.

D. Kernel Function

Each image is described by the above mentioned 5 features. Then we build kernel function for each of these 5 features. For this we adopt RBF kernel.

$$ker(x, x') = \exp -\gamma^{-1} d(x, x')$$

. Where $d(x, x')$ is the L2 distance.

E. Algorithm Comparision

As learning approach has only been implemented in BMK-LSH and BSP-MK-LSH, so it would only be fair to compare only these two algorithms. Table 2 shows comparision results. l is no. of projection rounds.

Algorithm	Learning Complexity	Searching Complexity
BMK-LSH	$Tmdn^{\frac{1}{1+\epsilon}}$	mdb
BSP-MK-LSH	$ldn^{\frac{1}{1+\epsilon}}$	mdb

TABLE II. ALGORITHM COMPLEXITY.

Algorithm	mean	std
Simple KLSH	0.16902	$\pm$0.00100
MK-LSH	0.16761	$\pm$0.00086
BMK-LSH	0.20460	$\pm$0.00400
BSP-MK-LSH	0.21139	$\pm$0.00362

TABLE III. MEAN AND STANDARD DEVIATION.

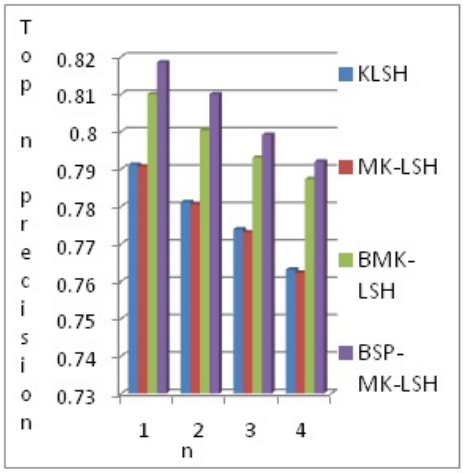

Fig. 1. Average Top n - precisions of different algorithm.

F. Experimental Result

We perform experiments in terms of top-n precision where $n = 1, 2, 3, 4$. We fix parameters as $\rho = 0.1, b = 100, p1 = 100, l = 20, t = 30$. Parameter ρ is basically used to show experimental result in the form of database items searched instead of measuring search time. It is not difficult to understand that with increasing ρ value more image examples will be fetched and inspected and also increases the calculations of kernel matrix but after a certain point retrieval of relevent images halt and only resistance is added. As mentioned earlier in section 3 values of b and $p1$ represents the hashing bit size and size of subset used for computing kernel matrix respectively which and when they are increased our average precision also increases. Also the values of b and p needs to be same which is clearly visible in the $step8$ of algorithm 3 which carries the multiplication of matrices of size $p1 \times p1$ and $b \times 1$. The value of t indices represent the size of vector e^l which is not very sensitive to the algorithm though increasing t would not always increase performance it could also degrade the performance. Next is the value of l which represents the iteration rounds. Through our experiments we have found that l obtains a saturation value after which increasing l introduces no effect. The algorithm 2 uses a regularizer The effect of adding a regularizer prevent overfitting of models/features by penalizing them with extreme parameter value. They fine tune the complexity of the model by using augmented error function with cross validation. The data sets used in complex models produces levelling-off of validation as complexity of the models increases. The training data set error decreases while validation data set error remain constant. A second factor introduced by regularizer weighs the penalty against more complex models with an increasing varience in data errors providing increasing penalty as complexity increases. As a result produces fine judgement of size of feature set used than produced from learning by boosting iterations. We summarize detailed result of n precision in the figure 1.The mean Average precision (mAP) on Flickr dataset and standard deviation for all algorithm is shown in table 3.

VI. CONCLUSION

In this paper we have proposed a novel method of combining multiple kernels together by learning the optimal combination of the hashing bits holistically. We first analyzed the KLSH algorithm that is the basic building block of any multi-kernel LSH. Then we introduced a learning algorithm, Bootstrap Sequential Projection that learn the optimal weight of bits for each kernel. Bootstrapping is successfully applied to the hashing bit size learning. This learning algorithm increases its efficiency with increasing no. of kernel functions. We have conducted a thorough experiment to evaluate the performance of our proposed algorithm, which concludes this algorithm to be a better approach than boosting multi-kernel LSH In Future work we will try to implement our techniques in other domains like e-commerce, textual context etc.

REFERENCES

[1] L. J. Guibas, J. S. Mitchell, and T. Roos, "Voronoi diagrams of moving points in the plane," in *Graph-Theoretic Concepts in Computer Science*. Springer, 1992, pp. 113–125.

[2] K. L. Clarkson, "A randomized algorithm for closest-point queries," *SIAM Journal on Computing*, vol. 17, no. 4, pp. 830–847, 1988.

[3] H. Xia, P. Wu, S. C. Hoi, and R. Jin, "Boosting multi-kernel locality-sensitive hashing for scalable image retrieval," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 55–64.

[4] 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 (CSUR)*, vol. 33, no. 3, pp. 322–373, 2001.

[5] G.-H. Cha, X. Zhu, D. Petkovic, and C.-W. Chung, "An efficient indexing method for nearest neighbor searches in high-dirnensional image databases," *Multimedia, IEEE Transactions on*, vol. 4, no. 1, pp. 76–87, 2002.

[6] I. Daoudi, K. Idrissi, S. Ouatik, A. Baskurt, and D. Aboutajdine, "An efficient high-dimensional indexing method for content-based retrieval in large image databases," *Signal Processing: Image Communication*, vol. 24, no. 10, pp. 775–790, 2009.

[7] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.

[8] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," *Proceedings of the VLDB Endowment*, vol. 5, no. 5, pp. 430–441, 2012.

[9] J. Gan, J. Feng, Q. Fang, and W. Ng, "Locality-sensitive hashing scheme based on dynamic collision counting," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 541–552.

[10] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2130–2137.

[11] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels." in *NIPS*, vol. 22, 2009, pp. 1509–1517.

[12] S. Wang, S. Jiang, Q. Huang, and Q. Tian, "S3mkl: scalable semi-supervised multiple kernel learning for image data mining," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 163–172.

[13] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 1127–1134.

[14] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 1380–1393, 2013.

[15] M. J. Huiskes and M. S. Lew, "The mir flickr retrieval evaluation," in *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. New York, NY, USA: ACM, 2008.