International Journal of Database and Theory and Application
Vol.9, No.2 (2016), pp. 71-80
http://dx.doi.org/10.14257/ijdta.2016.9.2.08

Hashing via Efficient Addictive Kernel for Logistics Image Classification

Xiao-jun Liu, Qiu-ling Li, Bin Zhang and Jun-yi Li

Department of Logistics and Information Management, Zhuhai College of Jilin University, Zhuhai 519041,Guangdong,China
Faculty of Business and Administration, University of Macau, Macau 999078, Macau
Department of Logistics and Information Management, Zhuhai College of Jilin University, Zhuhai 519041,Guangdong,China
Electrical and Computer Engineering Department, National University of Singapore, Singapore 119077, Singapore
Lxj02041820@163.com

Abstract

In this paper, fast image search with efficient additive kernels and kernel locality-sensitive hashing has been proposed. As to hold the kernel functions, recent work has probed methods to create locality-sensitive hashing , which guarantee our approach's linear time, however existing methods still do not solve the problem of locality-sensitive hashing (LSH) and indirectly sacrifice the loss in accuracy of search results in order to allow fast queries. To improve the search accuracy, we show how to apply explicit feature maps into the homogeneous kernels, which help in feature transformation and combine it with kernel locality-sensitive hashing. We prove our method on several large datasets, and illustrate that it improve the accuracy relative to commonly used methods and make the task of object classification, content-based retrieval more fast and accurate.

Keywords*: kernel methods, feature map, homogeneous kernel, locality-sensitive hashing, image search*

1. Introduction

In Web 2.0 applications era, we are experiencing the growth of information and confronted with the large amounts of user-based content from internet. As each one can publish and upload their information to the internet, it is urgent for us to handle the information brought by these people from internet. In order to organize and be close to these vision data from Internet, it has caused considerable concern of people. Therefore, the task of fast search and index for large video or image databases is very important and urgent for multimedia information retrieval such as vision search especially now the big data in some certain domains such as travel photo data from the website and social network image data or other image archives.

With the growth of vision data, we focus on two important aspects of problem including nearest neighbor search and similarity metric learning. For metric learning, many of the researchers have proposed some algorithms such as Information-Theoretic metric learning [1]. As for nearest neighbors search, the most common situation and task for us is to locate the most similar image from an image database. Among all the methods, given the similarity of example and query item, the most common method is to find all the vision data among the

ISSN: 1975-0080 IJDTA

vision database and then sort them. However time complexity of this algorithm is too large and also impractical. Especially when we handle image or video data, this complexity will be not calculated, because it is very difficult for us to compute the distance of two items in higher dimensional space and also vision datum is sparse, so we can not complete it by limited time.

Many researchers believe that linear scan can solve this problem although we believe it is a common approach and not suitable for computing in large-scale datasets, however it promote the development of ANN. LSH was used in ANN algorithms. To get fast query response for high-dimensional space input vectors [3-5,1,6], when using LSH, we will sacrifice the accuracy. To assure a high probability of collision for similar objects, randomized hash function must be computed, this is also referred in many notable locality-sensitive hashing algorithms [7-8].

Although in object similarity search task, the LSH has played an important role, some other issues and problems have been neglected. In image retrieval, recognition and search tasks, we find they are very common:

A. In the sample feature space, traditionally LSH approaches can only let us to get a relatively high collision probability for items nearby. As a lot of vision datasets contained much rich information, we can find that the category tags are attached to YouTube and Flickr data and the class labels are attached to Caltech-101 images. However the low-level and high-level of vision samples has great gap, which means that the gap low-level features and high-level semantic information exists. To solve this problem, we intend to utilize the side additional information for constructing hash table.
B. As to manipulate nonlinear data which is linear inseparable, we commonly use kernel method in vision task because of its popularity. For instance, in vision model, objects are often modeled as BOF and kernel trick is an important approach in classifying these data from low-dimension space to high-dimension space. However, how to create hash-table in kernel spaces is a tough problem for us.

To verify our idea, we did several experiments in object search task. For example, we show our results on the Caltech-101[10] dataset, and demonstrate that our approach is superior over the existing hashing methods as our propose algorithm

In order to test our algorithm performance on dataset, We design some experiments on certain visual task such as Caltech-101 [10], and demonstrate that the performance of algorithm in our paper beyond the traditional LSH approaches on the dataset, as hash functions can be calculated beyond many kernels .Arbitrary kernel in ANN is suitable in our scheme, actually we can find that a lot of similarity hashing functions can be accessed in the task of vision search tasks based on content retrieval.

2. Homogeneous Kernel

In our paper, we mainly focus on some kernels similar like intersection, Jensen-Shannon, Hellinger's,$\chi 2$ kernels. In the fields of machine learning and vision search, we often use these kernels as learning kernels. These kernels has two common attributes: homogeneous and additive. The idea of kernel signature has been smoothly connected to these kernels in this section. Meanwhile we can use pure functions to represent these kernels. Also these attributes will be applied in our section three to obtain kernel feature maps. Through the kernel feature map, we can get their approximate expression.

Homogeneous kernels. A kernel k_l : $\square_0^+ \times \square_0^+ \to \square$ is $\gamma -$homogenous if

$$\forall m \geq 0: \quad k_l(ma, mb) = m^{\gamma} k_l(a, b) \tag{1}$$

When $\gamma = 1$ we believe $k_l(ma, mb)$ is homogeneous. Let $m = 1/\sqrt{ab}$,we can obtain a $\gamma -$homogeneous kernel and we can also write the formula as (2)

$$k_l(a,b) = m^{-\gamma} k_l(ma, mb) = (ab)^{\frac{\gamma}{2}} k_l(\sqrt{\frac{b}{a}}, \sqrt{\frac{a}{b}}) = (ab)^{\frac{\gamma}{2}} \kappa(\log b - \log a) \tag{2}$$

Here the pure function

$$\kappa(\lambda) = k_l(e^{\frac{\lambda}{2}}, e^{-\frac{\lambda}{2}}), \lambda \in \mathbb{R} \tag{3}$$

is called the kernel signature.

Stationary Kernels: A kernel $k_s : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}$ is called stationary kernels if

$$\forall l \in \mathbb{R} : k_s(l + a, l + b) = k_s(a, b). \tag{4}$$

let $l = -(a+b)/2$ the $k_s(a,b)$ is represented as

$$k_s(a,b) = k_s(l + a, l + b) = k_s(\frac{b-a}{2}, \frac{a-b}{2}) = \kappa(b-a) \tag{5}$$

$$\kappa(\lambda) = k_s(\frac{\lambda}{2}, -\frac{\lambda}{2}), \lambda \in \mathbb{R} \tag{6}$$

Here we call formula (6) kernel feature. In the field of machine learning or computer vision , most of the homogeneous kernels are composed of the Jensen-Shannon, intersection, χ2, Hellinger's kernels. So we can also view them additive kernels. In the next section ,we will focus on these kernels and their kernel maps. Table 1 shows the details [11].

χ^2 kernel. We define $k(a,b) = 2ab/(a+b)$ as the χ^2 kernel [13][14]. Here the χ^2 distance is then defined as $D^2(a,b) = \chi^2(a,b)$.

Jensen-Shannon(JS) kernel. We define $k(a,b) = (a/2)\log_2(a+b)/a + (b/2)\log_2(a+b)/b$ as the JS kernel. Here the JS kernel distance $D^2(a,b)$ can be obtained by $D^2(a,b) = KL(a|(a+b)/2) + KL(b|(a+b)/2)$, where we import the concept of Kullback-Leibler divergence computed by $KL(a|b) = \sum_{l=1}^{d} a_l \log_2(a_l / b_l)$.

Intersection kernel. We defined $k(a,b) = \min\{a,b\}$ as the intersection kernel[12]. The distance metric $D^2(a,b) = \| a - b \|_1$ is l^1 distance between variant a and b.

Hellinger's kernel. We defined $k(a,b) = \sqrt{ab}$ as the Hellinger's kernel and specified distance metric $D^2(a,b) = \| \sqrt{a} - \sqrt{b} \|_2^2$ as Helinger's distance between variant a and b. The function expression $\kappa(\lambda) = 1$ is the signature of the kernel, which is constant.

$\gamma -$ homogeneous parameters. In previous research paper, we can see that the homogeneous kernels are used by parameters $\gamma = 1$ and $\gamma = 2$ .When $\gamma = 2$ the kernel becomes $k(a,b) = ab$. Now in our paper, we can derive the $\gamma -$homogeneous kernel by formula (2).

3. Homogeneous Kennel Map

When handling low-dimensional data which is inseparable , we should create kernel feature map $\psi(x)$ for the kernel so that we can map our input data information in low-dimensional space to relatively high-dimensional (Hilbert) information space with <.,.>.

$$\forall a,b \in R^D : K(a,b) = <\psi(a), \psi(b)> . \quad (7)$$

In order to compute the feature maps and get approximate kernel feature maps expression for the homogeneous kernels, we should use Bochner's theorem by expanding the configuration of γ-homogeneous expression. Here we notice if a homogeneous kernel is Positive Definite [15], its signature will also be Positive Definite expression. The assumption condition is suitable for a stationary kernel. So depending on formula (2) and Bochner's

Theorem (9), we can derive the $k(a,b)$ and closed feature map .

We can compute the kernel density and feature map closed form [11]for most machine learning kernels .TABLE.1 illustrates the results.

Table 1. Common Kernels, Signature, and Their Feature Maps

kernel	$k(a,b)$	**signature** $\kappa(\theta)$	$\kappa(w)$	**feature** $\psi_\omega(a)$
Helinger's	$\sqrt{ab}$	1	$\delta(\omega)$	$\sqrt{a}$
χ^2	$\frac{2ab}{a+b}$	$\sec h(\theta/2)$	$\sec h(\pi\omega)$	$e^{iw\log a}\sqrt{a \sec h(\pi\omega)}$
Intersection	min{a, b}	$e^{-\lvert\theta\rvert/2}$	$\frac{2}{\pi}\frac{1}{1+4\omega^2}$	$e^{iw\log a}\sqrt{\frac{2a}{\pi}\frac{1}{1+4\omega^2}}$
JS	$\frac{a}{2}\log_2\frac{a+b}{a}+\frac{b}{2}\log_2\frac{a+b}{b}$	$\frac{e^{\theta/2}}{2}\log_2(1+e^{-\theta})+\frac{e^{-\theta/2}}{2}\log_2(1+e^{\theta})$	$\frac{2}{\log 4}\frac{\sec h(\pi\omega)}{1+4\omega^2}$	$e^{iw\log a}\sqrt{\frac{2}{\log 4}\frac{\sec h(\pi\omega)}{1+4\omega^2}}$

$$\begin{aligned} k(a,b) &= (ab)^{\frac{\theta}{2}}\int_{-\infty}^{+\infty} e^{-iw\lambda}\kappa(\omega)d\omega, \theta = \log\frac{b}{a}, \\ &= \int_{-\infty}^{+\infty}(e^{-iw\log a}\sqrt{a^\theta\kappa(\omega)})^*(e^{-iw\log b}\sqrt{b^\theta\kappa(\omega)})d\omega. \end{aligned} \quad (8)$$

$$\psi_w(a) = e^{-iw\log a}\sqrt{a^\gamma\kappa(\omega)} . \quad (9)$$

4. Kernelized Locality-Sensitive Hashing

To create and conduct the data association, we take the approach of Kernelized LSH [17] which is also a hash table-based algorithm. KLSH is proposed based on LSH algorithm, which is more efficient and accurate for query search and matching. When searching the input query, the KLSH approach can quickly locate the possible similar and nearest neighbor items in the hash table and match it. In addition, KLSH has another characteristic, traditional LSH methods can only find a part of hashes in the kernel space, while KLSH can locate all the possible hash-tables in kernel space. Moreover KLSH has been applied in the vision search tasks by large scale dataset such as Tiny Image and other datasets[17].

Similar to LSH, constructing the hash functions for KLSH has been the key problem for us. That means if we intend to compute the collision probabilities of input query and the database points, we should compute the extent of similarity between them in the database as proposed by [18].

KLSH principle: Any locality-sensitive hashing algorithm is based on the probability of distribution of hash function clusters. So we should compute the collision probability of a bundle of points, for example m and n.

$$P_r(h(m)=h(n)) \quad sim(\quad . \tag{10}$$

We can also view the problem as the issue of computing the similarity of objects between m and n. Here $sim(m,n)$

in the algorithm is the measure function of calculating the similarity, while $h(m)$ and $h(n)$ are randomly selected from the hash function cluster H. The instinct beyond this is that we find the fact that m and n will collide in the same hash bucket. So those objects which are significantly similar will be more possible to be memorized in the hash table and this eventually results in confliction[1].

We can derive the similarity function expression according to the vector inner product:

$$sim(m,n) = m^T n \tag{11}$$

In [18,34],the definition of LSH function has been extended from formula (10) as

$$h_{\vec{r}}(m) = \begin{cases} 1, & if\ \vec{r}^T m \geq 0 \\ 0, & else \end{cases} \tag{12}$$

Here we create a random hyper plane vector $\vec{r}$ .The distribution of $\vec{r}$ fit has a zero-mean multi-Gaussian

$N(0,\sum)$ distribution. The dimensionality of $\vec{r}$ is the same with the input vector m. This demonstrates that the statistical characteristic of input vector is uniquely matched with each hash function. Meanwhile this verification has been detailed reported in the LSH attribute in [19] . When we project on a point m, actually the sigh function we obtain in this process is a hash function and then we repeat it k times, a couple of hashes can be created. We can also call this couple of hashes hash bucket . The hash bucket can be formed as formula (13)

$$g(m) = \langle h_1(m),...h_t(m),...h_k(m) \rangle \tag{13}$$

From (13), we can see that after repeating k times, we can get one column of hash bucket (14), then repeating b times, we can finally obtain the hash bucket $g(m)$.

$$g_j(m) = \langle h_1(m),...h_t(m),...h_k(m) \rangle (1 < j < b) \tag{14}$$

When given the value of b, we can get the all the hash functions located in the bucket, we can see formula (15).

$$g(m) = \begin{Bmatrix} h_{1,1\vec{r}}(m) & h_{2,1\vec{r}}(\mathrm{m}) & \ldots & h_{s,1\vec{r}}(m) & \ldots & h_{t,1\vec{r}}(m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{1,j\vec{r}}(p) & h_{2,j\vec{r}}(m) & \ldots & h_{s,j\vec{r}}(m) & \ldots & h_{t,j\vec{r}}(m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{1,b\vec{r}}(m) & h_{2,b\vec{r}}(m) & \ldots & h_{s,b\vec{r}}(m) & \ldots & h_{t,b\vec{r}}(m) \end{Bmatrix} (1 < \mathrm{j} < \mathrm{b}; 1 < \mathrm{s} < \mathrm{t}) \tag{15}$$

Due to the fact that we compute the similarity measure function in high-dimensional kernel space, so the similarity function can also be extended and wrote as:

$$\kappa(\mathrm{m}_{i,}\, m_j) = \phi(\mathrm{m}_i)^T \phi(\mathrm{m}_j) \tag{16}$$

In formula (16), we use kernel function $\phi(m)$ to construct $\kappa(\mathrm{m}_{i,}\, m_j)$ to complete the kernel

mapping for the points of m_i and m_j. And $\phi(m_i)^T \phi(m_j)$ is a product of projection on hash function from the $\Re$ space. The problem is that nothing is known about the data while in

kernel space to generate $\vec{r}$ from $N(0, \Sigma)$ .Therefore, in order to construct the hash function, $\vec{r}$ needs to be created so that we can quickly compute the $\vec{r}^T \phi(m)$ function based on the kernel.

Similar with normal $\vec{r}^T$, we could use only the kernel of $\phi(m)$ to approximately compute the function of $\vec{r}^T \phi(m)$. We should select a subset of database to construct $\vec{r}$. By the large number of central limit theory, if we intend to choose a parts of database items from the whole database to form the dataset S, the sample of kernel data must be satisfied by the distribution with mean μ and variance Σ. The variable z_a can be wrote as:

$$z_a = \frac{1}{k} \sum_{i \in s} \phi(m_i) \tag{17}$$

With the growth of variable a , the theory tells us that the vector $\tilde{z}_a = \sqrt{t}(z_a - \mu)$ has also been satisfied by the distribution of normal Gaussian.

We used the whitening transform to obtain $\vec{r}$:

$$\vec{r} = \Sigma^{-1/2} \tilde{z}_a \tag{18}$$

The LSH function has been yielded:

$$h(\phi(m)) = \begin{cases} 1, & if\ \phi(m)^T \Sigma^{-1/2} \tilde{z}_a \geq 0 \\ 0, & else \end{cases} \tag{19}$$

As analyzed above, we use kernel function to represent the database data, then the statistical data like variance and mean are uncertain. If we intend to estimate and calculate μ and Σ, we could sample the data from the database by

KPCA and eigen decomposition in [20] and we let $\Sigma = V \Lambda V^T$ and $\Sigma^{-1/2} = V \Lambda^{-1/2} V^T$, therefore we can obtain the hash function $h(\phi(m))$:

$$h(\phi(m)) = sign\ \phi(m)^T V \Lambda^{-1/2} V^T \tilde{z}_a \tag{20}$$

From the above, we can see how to construct the hash function for the kernel matrix input vectors. In this case, we

let the kernel matrix input be $K = U \Omega U^T$ by decomposing the K matrix. Here Ω and Λ have the same none-zero eigen value, it is also be viewed as another form of kernel matrix input. From [20],we compute the projection

$$v_t^T \phi(m) = \sum_{i=1}^{n} \frac{1}{\sqrt{\theta_t}} u_t(i) \phi(m_i)^T \phi(m). \tag{21}$$

Here u_t and v_t are respectively the t-th eigenvector of the kernel matrix and its covariance matrix.

As mentioned before, we choose n data points from the database to form $\phi(m_i)$, traverse all the t eigenvectors and conducting the computation yields:

$$\begin{aligned} h(\phi(m)) &= \phi(m)^T V \Lambda^{-1/2} V^T \tilde{z}_a \\ &= \sum_{t=1}^{n} \sqrt{\theta_t}\, v_t^T \phi(m)^T v_t^T \tilde{z}_a. \end{aligned} \tag{22}$$

Substituting (21) into equation (22) yields

$$\sum_{t=1}^{n}\sqrt{\theta_t}\,(\sum_{i=1}^{n}\frac{1}{\sqrt{\theta_t}}u_t(i)\phi(m_i)^T\phi(m)).(\sum_{i=1}^{n}\frac{1}{\sqrt{\theta_t}}u_t(i)\phi(m_i)^T\tilde{z}_a). \tag{23}$$

Simplifying (23) yields

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(\phi(m_i)^T\phi(\mathrm{m})).(\phi(m_j)^T\tilde{z}_a)(\sum_{t=1}^{n}\frac{1}{\sqrt{\theta_t}}u_t(i)u_t(j)). \tag{24}$$

$K_{ij}^{-1/2}=\sum_{k=1}^{n}\frac{1}{\sqrt{\theta_k}}u_k(i)u_k(j)$

Since , we further simplify the (24) yields

$$h(\phi(m))=\sum_{i=1}^{n}w(i)(\phi(m_i)^T\phi(m)), \tag{25}$$

where $w(i)=\sum_{j=1}^{n}K_{ij}^{-1/2}\phi(\mathrm{m}_j)^T\tilde{z}_a$.

Through the above derived formula $w(i)$ we can obtain $\vec{r}=\sum_{i=1}^{n}w(i)\phi(\mathrm{m}_i)$ which obeys random Gaussian distribution, then we can substitute the equation (17) into $w(i)$.

$$w(i)=\frac{1}{a}\sum_{j=1}^{n}\sum_{l\in s}K_{ij}^{-1/2}K_{jl} \tag{26}$$

We neglect the term of $\sqrt{a}$,and finally the $w(i)$ simplified yields (27). e_s represents the unit vector for S.

And therefore hash function for kernel input will finally be (28).

$$w=K^{1/2}.\frac{e_s}{k} \tag{27}$$

$$h(\phi(m))=sig\sum_{i=1}^{n}(\quad w(i),(m \tag{28})$$

κ is the kernel mapping matrix for points m and m_i in space. After several iterations , the hash function will form a hash bucket.

In order to get the suitable parameters in this process , we implement the query matching for several iterations. The detailed algorithm illustrates finally in Algorithm 1.

Consider

(1). Process Data
(a) Obtain K matrix from database throughout the n points.
(b) Obtain e_s by randomly sampling a subset from the { 1,2,..,n }
(c) Project on a^{th} subset to obtain $h_a(\phi(m))$.
(d) Obtain $w = K^{1/2}.e_s / a$
(e) Project $w(i)$ onto the points in kernel space
(f) Obtain hash bucket $g_j(m)$
(2). Query Processing
(a) Obtain the same hash bucket in (29) from the database
(b) Use KNN search for query matching.

$$g_j(m) = [h_1(\phi(m)), h_2(\phi(m)), \ldots, h_t(\phi(m))] \quad (29)$$

5. Experimental Result

In the experiment, we proposed the homogenous kernel-hashing algorithm and verify the high efficiency

On the dataset. In our scheme, homogenous kernel-KLSH method makes it possible to get the unknown feature embeddings. We use these features to conduct vision search task to locate the most similar items in the database, and the neighbors we find in the task will give their scores on the tags. The method proved to be more effective and accurate than the linear scan search.

In this part, we design and verify our algorithm on the Caltech-101 dataset in Figure.1.Caltech-101 dataset is a benchmark on image recognition and classification, which has 101 categories objects and each category has about 100 images, so 10000 images totally. In recent years ,many researchers has done useful research on this dataset such us proposing some important and useful image represent kernels[21].Also there are many published paper focused on this dataset ,some of which is high valuable and significantly historic. For example, paper [22-24] respectively state their contribution on the dataset. The author of [23] Grauman proposed the matching method for pyramid kernel of images histograms. While the author [22] Berg proposed and created the CORR kernel of image local feature using geometric blur for matching local image similarity.

In our paper, we apply our algorithm to complete the vision classification and similar search task. The platform of our benchmark is based on Intel 4 core 3.6 GHZ CPU and 16GB of memory and 2TB hard disk.

We used χ^2 kernel for γ-homogeneous kernel maps (γ=1/2) and applied the nonlinear RBF-χ^2 kernel designed in [25] [21] to the SIFT-based local feature. Meanwhile we applied and learnt the homogenous kernel map beyond it. Compared with the non-learnt kernel, our learnt kernel has been more accurate. And we use KNN classifier respectively for KLSH and linear scan to compute the accuracy of classification. We also compare it with CORR [26] and the result proves to be better than them, here we use 15 images per class for training task.

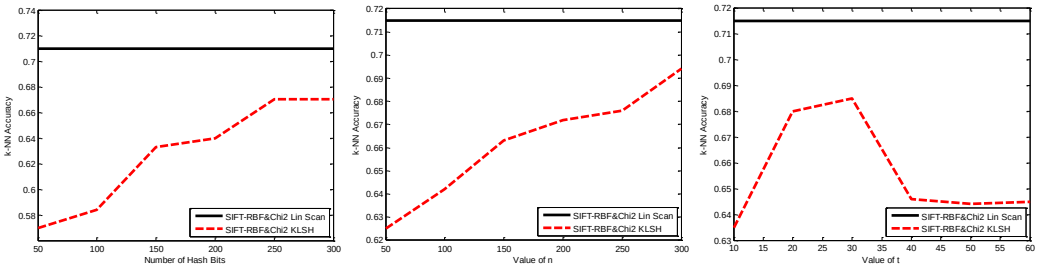

Figure 2. Hashing Using a RBF-χ^2 Kernel for SIFT Based on Homogenous Kernels χ^2 (γ=1/2). We Choose t = 30, n= 300, and b= 300 in Our Experiment

From Figure 2 we can see that the growth of parameters is closely related with accuracy. As is seen, the accuracy increased with the increase of n, while it has little relationship with the number of t and b. The value of (n,t,b) is chosen as n=300,b=300,t=30 as the best parameters through a series of experiments.

We find the combination of these parameters can result in better performance over the large-scale dataset. Meanwhile it can be seen that our approach with homogenous kernel map has higher accuracy than CORR-KLSH with metric learning [25].

Table 2. Accuracy on Caltech-101

#train	ours	[20]	[28]	[29]	[30]	[31]	[32]
15	68.5	59.05	56.4	52	51	49.52	44
30	75.2	66.23	64.6	N/A	56	58.23	63

6. Conclusions

In our paper , we properly use the concept of homogeneous kernel maps to help us to solve the problem of approximation of those kernels, including those commonly used in machine learning such as χ^2 ,JS, hellinger and intersection kernels .Combining with the KLSH scheme, it enable us to be access to any kernel function for hashing functions. Although our approach is inferior to linear scan search in time but it can guarantee the search accuracy will not be affected. Moreover we don't need to consider the distribution of input data, to some extent, it can be applicable for many other database as Flicker and Tiny Image. Experimental results demonstrate that it is superior to standard KLSH algorithm.

Acknowledgment

We would like to appreciate Brian Kulis from UC Berkeley University for helpful comments on experiments and also we should appreciate professor Yan Shuicheng for advice on our paper. Our work was supported in part by National Program on Key Basic Research Project (973 Program) 2010CB731403/2010CB731406

References

[1] B. Kulis and P. J. K. Grauman, "Fast Similarity Search for Learned Metrics", In IEEE Trans. Pattern Analysis and Machine Intelligence, **(2009)**.

[2] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality", In STOC, **(1998)**.
[3] G. Shakhnarovich, P. Viola and T. Darrell, "Fast Pose Estimation with Parameter-Sensitive Hashing", In ICCV, **(2003)**.
[4] G. Shakhnarovich, T. Darrell and P. Indyk, "Nearest-Neighbor Methods in Learning and Vision: Theory and Practice", The MIT Press, **(2006)**.
[5] G. Shakhnarovich, P. Viola and T. Darrell, "Fast pose estimation with parameter-sensitive hashing", In ICCV, **(2003)**.
[6] O. Chum, J. Philbin and A. Zisserman, "Near Duplicate Image Detection: min-hash and tf-idf Weighting", In BMVC, **(2008)**.
[7] M. Charikar, "Similarity Estimation Techniques from Rounding Algorithms", In ACM Symposium on Theory of Computing, **(2002)**.
[8] G. P. Indyk and R. Motwani, "Similarity Search in High Dimensions via Hashing", In Proceedings of the 25th International Conference on Very Large Data Bases, **(1999)**.
[9] A. Torralba, R. Fergus and Y. Weiss, "Small codes and large image databases for recognition", In CVPR, **(2008)**.
[10] L. F. Fei, R. Fergus and P. Perona, "Learning Generative Visual Models from Few Training Examples: an Incremental Bayesian Approach Tested on 101 Object Categories", In Workshop on Generative Model Based Vision, **(2004)**.
[11] M. Hein and O. Bousquet, "Hilbertian metrics and positive definite kernels on probability measures", in Proceeding AISTAT, **(2005)**.
[12] A. Barla, F. Odone and A. Verri, "Histogram intersection kernel for image classification", In ICIP, **(2003)**.
[13] J. Puzicha, Y. Rubner, C. Tomasi and J. Buhmann, "Empirical evaluation of dissimilarity measures for color and texture", In Proc. ICCV, **(1999)**.
[14] D. R. Martin, C. Fowlkes and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues", IEEE Trans. Pattern Analysis and Machine Intelligence, **(2004)**.
[15] B. Scholkopf and A. J. Smola, "Learning with Kernels", MIT Press **(2002)**.
[16] A. Rahimi and B. Recht, "Random features for large-scale kernel machines", In NIPS, **(2007)**.
[17] B. Kulis and K. Grauman, "Kernelized Locality-Sensitive Hashing for Scalable Image Search", In ICCV, **(2009)**.
[18] Z. Chen, J. Samarabandu and R. Rodrigo, "Recent Advances in Simultaneous Localization and Map building Using Computer Vision", Journal on Advanced Robotics, **(2007)**.
[19] M. Goemans and D. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfactory Problems Using Semidefinite Programming", Journal on communication of the ACM, **(1995)**.
[20] B. Scholkopf, A. Smola and K. R. Muller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem", Neural Computation, **(1998)**.
[21] A. Vedaldi, V. Gulshan, M. Varma and A. Zisserman, "Multiple kernels for object detection", In ICCV, **(2009)**.
[22] A. C. Berg, T. L. Berg and J. Malik, "Shape matching and object recognition using low distortion correspondence", In CVPR, **(2005)**.
[23] K. Grauman and T. Darrell, "Discriminative classification with sets of image features", In ICCV, **(2005)**.
[24] A. D. Holub, M. Welling and P. Perona, "Combining generative models and fisher kernels for object recognition", In ICCV, **(2005)**.
[25] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off", In CVPR, **(2007)**.

Authors

Xiao-jun Liu, He's a PhD candidate now in Macau University of Science and Technology (MUST) and he's doing his doctoral dissertation on SCM and logistics management information system and information technology. He is currently a lecturer in Department of Logistics and Information Management, Zhuhai College of Jilin University. He has worked on a research program which is A Cloud-based Supply Chain Management System.

