Applied Mechanics and Materials Vols. 651-653 (2014) pp 2168-2171
© (2014) Trans Tech Publications, Switzerland
doi:10.4028/www.scientific.net/AMM.651-653.2168

Submitted: 02.08.2014
Accepted: 05.08.2014

Optimized K-means Hashing for Approximate Nearest Neighbor Search

Qin-Zhen Guo, Zhi Zeng, Shuwu Zhang Yuan Zhang and Guixuan Zhang

95 Zhongguancun East Road, 100190, BEIJING, CHINA

{ qinzhen.guo, zhi.zeng, shuwu.zhang, yuan.zhang, guixuan.zhang }@ia.ac.cn

Keywords: Approximate nearest neighbor search, vector quantization, hashing, image retrieval.

Abstract. Hashing which maps data into binary codes in Hamming space has attracted more and more attentions for approximate nearest neighbor search due to its high efficiency and reduced storage cost. K-means hashing (KH) is a novel hashing method which firstly quantizes the data by codewords and then uses the indices of codewords to encode the data. However, in KH, only the codewords are updated to minimize the quantization error and affinity error while the indices of codewords remain the same after they are initialized. In this paper, we propose an optimized k-means hashing (OKH) method to encode data by binary codes. In our method, we simultaneously optimize the codewords and the indices of them to minimize the quantization error and the affinity error. Our OKH method can find both the optimal codewords and the optiaml indices, and the resulting binary codes in Hamming space can better preserve the original neighborhood structure of the data. Besides, OKH can further be generalized to a product space. Extensive experiments have verified the superiority of OKH over KH and other state-of-the-art hashing methods.

Introduction

Approximate nearest neighbor (ANN) search plays a very important role in many computer vision problems such as image retrieval, scene classification and 3D reconstruction. Nowadays, several ANN techniques have been developed including tree-based methods, vector quantization methods and hashing-based methods.

The tree-based methods such as modified KD-tree [1] use spatial partitions and recursive hyperplane decomposition to implement similarity search. However, for high-dimensional data, the tree-based methods can degenerate to exhaustive search.

Vector Quantization (VQ) is an effective method for ANN search. In VQ, the Euclidean distances of vectors can be approximated by the distances of the codewords that quantize them. To achieve satisfactory results, the VQ methods need numerous codewords. But this is intractable for practical application. To deal with this problem, product quantization (PQ) [2] was designed to decompose the original data space into the Cartesian product of m low-dimensional subspaces and each subspace is quantized separately by k-means.

The hashing-based methods [3, 4, 5, 6, 7, 8] have attracted more and more attentions for ANN search. These methods transform the original data into compact binary codes in Hamming space that preserve the original neighborhood structure of data. The similarity between two data points in the original space is approximated by the Hamming distance of their hashed codes in the Hamming space.

Recently, a novel method, k-means hashing (KH) [4] which combines the hashing methods with the vector quantization methods has been developed. In KH, data are quantized by the codewords and encoded by the indices of the codewords. The effectiveness of KH is largely affected by two aspects. One is the codewords $\boldsymbol{c}_i$. The other is the index of $\boldsymbol{c}_i$. The KH method updates $\boldsymbol{c}_i$ by minimizing the quantization error and affinity error. On the other hand, since all the data are encoded by the index of $\boldsymbol{c}_i$, the optimal indices play a very important role for the effectiveness of KH, even more important than the codewords. However this problem is unaddressed in KH. In KH, the index of $\boldsymbol{c}_i$ which is initiated by PCAH [5] is remaining the same along the whole procedure. This will incur ineffectiveness. In this paper, we propose an optimized k-means hashing (OKH) method to encode data by binary codes. In OKH, we simultaneously optimize the codewords and the indices of them to minimize the quantization error and the affinity error.

Optimized K-means Hashing

First of all, let us introduce some notations. The purpose of hashing is to map a d-dimensional data $\boldsymbol{x} \in \mathbb{R}^d$ to a B-bit binary code $I(\boldsymbol{x}) \in \{-1,1\}^B$. We have a training set $\boldsymbol{T}$ of n data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ that form the rows of the data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. The codebook is a set $\boldsymbol{C}$ of k codewords $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$. $c(\boldsymbol{x})$ is the codeword that quantizes $\boldsymbol{x}$. The binary code (or index) of $\boldsymbol{c}_i$ is $I(\boldsymbol{c}_i)$. $\boldsymbol{X}^i$ is the i-th subspace of $\boldsymbol{X}$.

K-means Hashing. There are two main steps in KH. The first step is quantizing the data by codewords based on nearest neighbor (NN) rule, which will cause quantization error as Eq. 1

$$E_{quan} = \frac{1}{n} \sum_{x \in T} \|x - c(x)\|^2 \tag{1}$$

And the Euclidean distance between data can be approximated by Eq. 2

$$D(x_i, x_j) \simeq D(c(x_i), c(x_j)) \tag{2}$$

$D(x_i, x_j) = \|x_i - x_j\|$ denotes the Euclidean distance. The second step is encoding the data by indices of the codewords and approximating the Euclidean distance between codewords by the Hamming distance of their indices. This step will cause the affinity error as Eq. 3,

$$E_{aff} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} w_{ij} (D(c_i, c_j) - H(I(c_i), I(c_j)))^2 \tag{3}$$

where $w_{ij} = n_i n_j / n^2$, and n_i and n_j are the number of data quantized by $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ respectively. Here $H(I(\boldsymbol{c}_i), I(\boldsymbol{c}_j)) = s \cdot \mathrm{sqrt}(h(I(\boldsymbol{c}_i), I(\boldsymbol{c}_j)))$ where $h(\cdot, \cdot)$ is the Hamming distance and s is a constant scale.

Taking both quantization error and affinity error into consideration, KH find the optimal codewords $\boldsymbol{c}_i$ by minimizing the following objective function,

$$E = E_{quan} + \lambda E_{aff} \tag{4}$$

where λ is weight. To minimize Eq. 4, KH uses an alternating method: firstly fix $\boldsymbol{c}_i$ and assign $\boldsymbol{x}$ to $\boldsymbol{c}_i$ based on NN rule. Then update $\boldsymbol{c}_i$ to minimize Eq. 5.

$$\begin{aligned} c_j = \arg\min_{c_j} (&\frac{1}{n} \sum_{x, c(x) = c_j} \|x - c_j\|^2 \\ &+ 2\lambda \sum_{i; i \neq j} w_{ij} (D(c_i, c_j) - H(I(c_i), I(c_j)))^2) \end{aligned} \tag{5}$$

For the whole data space, if we use B-bit codes to encode the data, 2^B codewords need to be computed. This is intractable especially when B is a large, such as B=64, 128 even 512. This issue can be tackled by generalize KH to product space. Before encoding the data, the data are divided into m subspaces and KH use B/m bits codes to encode every subspace. Thus only $2^{(B/m)}$ codewords are needed in every subspace.

Proposed Method. In order to deal with the issue of optimal indices of $\boldsymbol{c}_i$, in our method, we update both the cluster centers $\boldsymbol{c}_i$ and the indices of $\boldsymbol{c}_i$, $I(\boldsymbol{c}_i)$ to minimize Eq.4.

Directly Eq. 4 is infeasible since the Hamming distance computation is not differentiable. We notice that the Hamming distance between two binary codes $\boldsymbol{I}_i$ and $\boldsymbol{I}_j$ is given by the number of bits that are different between them, which can be calculated as

$$h(I_i, I_j) = \frac{1}{4} \|I_i - I_j\|^2 \tag{6}$$

Putting Eq. 1, Eq. 3, Eq. 6 into Eq. 4, we can optimize the index of $\boldsymbol{c}_j$, $I(\boldsymbol{c}_j)$ with other $\{I(\boldsymbol{c}_i)\}_{i \neq j}$ fixed:

$$\begin{aligned} I_j = \arg\min_{I_j} &\sum_{i \neq j; i=1}^{k} w_{ij} (D_{ij} - \frac{1}{2} s \|I_i - I_j\|)^2 \\ &+ \beta \cdot Tr(I_j I_j^T) \end{aligned} \tag{7}$$

$I(\boldsymbol{c_i})$ and $D(\boldsymbol{c_i}, \boldsymbol{c_j})$ are written as I_i and D_{ij} respectively for short. $Tr(I_jI_j^T)$ is the trace of $I_jI_j^T$, which controls the scale of $\boldsymbol{I_j}$. β is a weighting parameter. Let,

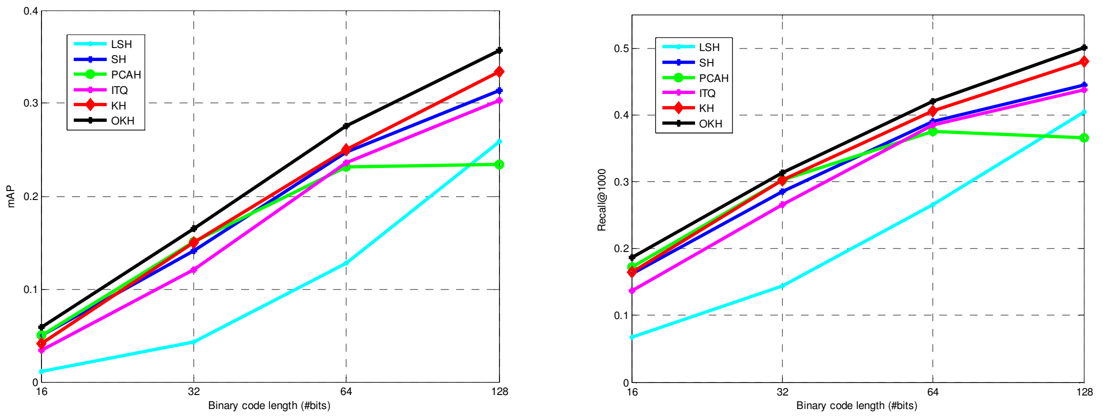
$$O=\sum_{i\neq j;i=1}^{k} w_{ij}(D_{ij}-\frac{1}{2}s\left\|I_i-I_j\right\|)^2+\beta\cdot Tr(I_jI_j^T) \tag{8}$$

We have,

$$\begin{aligned}\frac{\partial O}{\partial I_j}&=\sum_{i\neq j;i=1}^{k}(\frac{\partial(w_{ij}D_{ij}^2)}{\partial I_j}-\frac{\partial(w_{ij}D_{ij}s\left\|I_i-I_j\right\|)}{\partial I_j}\\&\quad+\frac{\partial(\frac{1}{4}w_{ij}s^2\left\|I_i-I_j\right\|^2)}{\partial I_j})+2\beta I_j\\&=\sum_{i\neq j;i=1}^{k}(-w_{ij}D_{ij}s(I_j-I_i)\left\|I_j-I_i\right\|^{-1}\\&\quad+\frac{1}{2}w_{ij}s^2(I_j-I_i))+2\beta I_j\\&=\sum_{i\neq j;i=1}^{k}w_{ij}(I_j-I_i)(\frac{1}{2}s^2-D_{ij}s\left\|I_j-I_i\right\|^{-1})+2\beta I_j\end{aligned}$$

Then we can optimize Eq.8 by gradient descent algorithm. After obtaining the optimal $\boldsymbol{I_j}$, we binarize it by zero.

By the same token with KH, OKH can be naturally generalized to the product space. Firstly, we decompose the whole data space into *m* subspaces. Then we do OKH in every subspace.



(a) (b)

Figure 1. Experiments on SIFT1M in term of mAP and Recall: (a) 1000-mAP; (b) Recall@1000.

Experiments

In this section, we evaluate the proposed method OKH for approximate nearest neighbor (ANN) search. We compare our method with LSH[6], SH[7], ITQ[8], PCAH[5] and KH[4]. We perform experiments with SIFT1M [2] and GIST1M [2]. For GIST1M which is a 960-dimensional dataset, we reduce the dimensionality to 512 by PCA. We randomly choose 5K vectors as queries, 100K of the rest as training set, and the remaining as database. The ground truth is defined by the 1000 nearest neighbors computed by the linear scan. The retrieved points are computed by ranking their Hamming distance to the query. The performance is measured by mean Average Precision (mAP) and recall. For SIFT1M, we set the subcodes length b=(B/M)=4 and b=8 for GIST1M. λ and β are set to 10. The indices of codewords in every subspace are initialized by PCAH. For KH and OKH, we implement the corresponding algorithm for 50 iterations.

Results and Analysis. The results on SIFT1M in term of mAP and Recall are presented in Figure 1. As we can see, our OKH method achieves the best results from 16 bits to 128 bits. Due to that OKH can find both the optimal codewords and the indices while KH can only find the optimal codewords, our OKH performs better than KH with all testing code length especially when longer bits are used. LSH which adopts data-independent random projection gets worse results, since the random projections may be not independent of each other. Table 1 shows the experiments on GIST1M dataset in term of Recall@1000. The best results are shown in bold face. Our OKH also obtains the best results through all the bit length. SH which assumes that the data are multidimensional uniform distributed performs worse mainly because the assumption does not hold on this dataset.

Table 1. Experiments on GIST1M in term of Recall@1000.

#bits	16	32	64	128
LSH	0.0299	0.0620	0.1256	0.2167
SH	0.0376	0.0663	0.1091	0.1329
PCAH	0.0669	0.0838	0.0864	0.0813
ITQ	0.0863	0.1531	0.2143	0.2687
KH	0.0871	0.1569	0.2268	0.2796
OKH	**0.0878**	**0.1689**	**0.2342**	**0.2906**

Conclusion

In this paper, we propose an optimized k-means hashing (OKH) method, for fast approximate nearest neighbor search. Our OKH approach can find both the optimal codewords and the indices of them to minimize the objective function which combines quantization error with affinity error. The resulting binary codes of OKH in Hamming space can better preserve the original neighborhood structure of the data. However, in OKH, the indices of the codewords are initialized by PCAH which can not generate codes longer than the dimensionality of the data. This will make OKH can not generate codes longer than the dimensionality of the data. However, if we use other hashing method like [3], this will enable OKH produce longer codes. In the future, we will tackle this issue.

Acknowledgment

This work has been supported by the National Key Technology R&D Program of China under Grant No. 2012BAH04F02, 2012BAH88F02, 2013BAH61F01 and 2013BAH63F01 and the International S&T Cooperation Program of China under Grant No. 2013DFG12980.

References

[1] J. Beis, and D. Lowe, in: *CVPR*, 1997, p. 1000.

[2] H. Jegou, M. Douze, and C. Schmid, *IEEE TPAMI*, Vol. 33 (2011) p. 117.

[3] Q.-Z. Guo, Z. Zeng, S. Zhang, Y. Zhang, and F. Wang, in: *ICME*, 2013, p. 1.

[4] K. He, F. Wen, J. Sun, in: *CVPR*, 2013, p. 2938.

[5] B. Wang, Z. Li and M. Li, in: *ICME*, 2006, p. 353.

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, in: *Annu. Symp. Computational Geometry*, ACM, 2004, p. 253.

[7] Y. Weiss, A. Torralba and R. Fergus, in *NIPS*, 2008, p. 1753.

[8] Y. Gong and S. Lazebnik, in *CVPR*, 2011, p. 817.