Iterative Self-Organizing Hash Algorithm Post-print

Authors: Han Xuelian, Tian Aikui, Wang Zhen, Lu Haitao

Date: 2019-04-01T00:00:00+00:00

Abstract

To address the issues of centroid uncertainty and limited expressiveness of discrete encoding in existing hashing algorithms, we propose the iterative self-organizing hashing (ISOH) algorithm. This algorithm employs iterative self-organizing data analysis to quantize space, thereby improving nearest neighbor retrieval accuracy. For cluster center initialization, the farthest average distance method is used to select initial cluster centers, avoiding the randomness of initial cluster centers. To solve the problem of limited varieties of binary codes represented by fixed code lengths, a multiple encoding mechanism is proposed. In terms of time complexity, the ISOH algorithm utilizes a product space to obtain longer codes at a lower cost. Experimental results demonstrate that on the SIFT, GIST, and CIFAR10 datasets, compared with concrete hashing algorithms such as K-means hashing and Scalable Graph Hashing, the ISOH algorithm can effectively improve the accuracy of nearest neighbor retrieval.

Full Text

Preamble

Vol. 37 No. 5
Application Research of Computers (ChinaXiv Cooperative Journal)
Accepted Paper

Iterative Self-Organizing Hashing Algorithm

Han Xuelian, Tian Aikui , Wang Zhen, Lu Haitao
(College of Computer Science & Technology, Shandong University of Technology, Zibo, Shandong 255000, China)

Abstract: To address the randomness of cluster centers and the limited representation capability of discrete binary codes in existing hashing algorithms, this paper proposes the Iterative Self-Organizing Hashing (ISOH) algorithm. ISOH

employs iterative self-organizing data analysis to quantize the feature space, thereby improving nearest neighbor retrieval accuracy. For cluster center initialization, the farthest average distance method is used to select initial centers, avoiding randomness. To overcome the limitation that fixed code lengths can represent only a finite number of binary codes, a multi-encoding mechanism is established. In terms of time complexity, ISOH utilizes product spaces to obtain longer codes at lower computational cost. Experimental results on SIFT, GIST, and CIFAR10 datasets demonstrate that ISOH significantly outperforms state-of-the-art hashing algorithms such as K-means Hashing and Scalable Graph Hashing in nearest neighbor retrieval accuracy.

Keywords: iterative self-organizing data analysis; multiple coding; product space; farthest average distance

0 Introduction

With the maturation of Internet technology applications, image and video data have exploded in volume, making rapid retrieval of relevant images from massive datasets a critical research focus. Text-based image retrieval techniques were proposed as early as the 1970s [1,2], relying on manual annotation of images. While simple and fast, this approach becomes increasingly impractical as image collections grow. Moreover, textual descriptions cannot precisely capture image semantics, often yielding unsatisfactory results. For instance, searching for "basketball" on search engines like Bing, Baidu, or 360 returns not only basketballs but also basketball stars, hoops, and courts [Figure 1: see original paper].

To overcome these limitations, researchers proposed tree-based image retrieval techniques [3]. These methods store image features in tree structures, with thresholds assigned to leaf nodes. During retrieval, the hierarchical structure enables rapid elimination of most data points, accelerating nearest neighbor search. The K-D tree [4], named for its tree-like structure [Figure 2: see original paper], recursively partitions the query space and searches subspaces concurrently. However, retrieval efficiency degrades as feature dimensionality increases. To address this, hash-based image retrieval techniques were introduced [5].

Hash-based methods map high-dimensional floating-point vectors into compact binary codes, using Hamming distance for nearest neighbor search. The earliest such algorithm, Locality Sensitive Hashing (LSH) [6], randomly generates linear hash functions to produce binary codes. While data-independent, LSH requires long codes for effective retrieval. To achieve compact yet effective codes, Shen and Weiss et al. [7,8] proposed Spectral Hashing (SH), which learns binary codes through spectral graph partitioning. However, SH suffers from high graph modeling complexity and assumes uniformly distributed data—an unrealistic condition for real datasets. Jiang et al. [9] introduced Scalable Graph Hashing (SGH) to reduce complexity by approximating the full graph through

feature transformations, eliminating explicit pairwise similarity matrix computation. Yet SGH requires parameter tuning via cross-validation.

Alternative approaches like Principal Component Analysis Hashing (PCAH) [10] and Random Rotating Hashing (RR) [11,12] quantize data using fixed hypercubes, which lack flexibility and adapt poorly to data distributions. K-means Hashing (KMH) [5,13,14] improves flexibility by clustering data with K-means, approximating inter-point distances with inter-center distances and allowing hypercube stretching. However, KMH suffers from random initialization and fixed cluster counts, limiting its retrieval performance.

To address these flexibility and accuracy issues, we propose ISOH, which integrates Iterative Self-Organizing Data Analysis (ISODATA) [15], farthest average distance initialization, multi-encoding [16], and product spaces [5,17] to achieve high accuracy. The algorithm framework is shown in [Figure 3: see original paper].

First, product space preprocessing is applied to training and test datasets [FIGURE:3(a)]. Second, similarity-preserving methods obtain cluster centers Z, which are used to classify the preprocessed training and test data [FIGURE:3(b)]. Third, a multi-encoding mechanism encodes the data [FIGURE:3(c)]. Finally, Hamming distances h(X , Y) are computed [FIGURE:3(d)].

Innovations of ISOH: a) **Adaptive quantization:** ISOH uses ISODATA [15] to dynamically adjust the number of cluster centers based on data distribution, unlike KMH' s fixed count.
b) **Robust initialization:** The farthest average distance method selects initial centers by first using the global mean, then iteratively splitting the most populous cluster at the point farthest from its center until K centers are obtained, eliminating randomness.
c) **Enhanced representation:** Multi-encoding [16] overcomes the limitation that fixed-length codes can represent only 2 patterns.
d) **Scalable complexity:** Product spaces [5,17] enable longer codes at lower computational cost by partitioning the original space.

1.1 Similarity-Preserving ISODATA Algorithm

KMH [14] suffers from unstable retrieval performance due to random cluster center initialization and fixed cluster counts. While ISODATA [15] dynamically adjusts cluster numbers, it has blind initialization and threshold setting issues. We address these as follows.

1.1.1 Cluster Center Initialization

Existing max-min distance methods [18] are sensitive to the first center and scaling factor , with high complexity requiring distance computations between

all points and established centers. We propose the **farthest average distance method**:

1. Compute the global mean as the first center.

2. Select the point farthest from the first center as the second center, partitioning data via nearest-neighbor assignment.

3. Iteratively split the most populous cluster using its farthest point as a new center until K centers are obtained.

Algorithm 1 summarizes this initialization.

1.1.2 Split and Merge Thresholds

Standard deviation across dimensions reflects positional deviation from cluster centers. Clusters with high dispersion and large populations should be split. Let be the maximum standard deviation in cluster j (j = 1,⋯,N_c). The split threshold is:

$$\theta_s = \alpha \times \frac{1}{N_c} \sum_{j=1}^{N_c} \sigma_{j\max}, \quad \alpha > 1$$

For merging, we minimize intra-class distance while maximizing inter-class distance [19]. Using a minimum spanning tree [20] with edge weights representing inter-center distances (total weight S_w), the merge threshold is:

$$\theta_c = \beta \times \frac{S_w}{N_c}, \quad 0 < \beta < 1$$

where N_c is the number of cluster centers.

1.2 Similarity-Preserving Objective Function

ISODATA introduces quantization error, and using Hamming distance to approximate Euclidean distance creates similarity error. Enumerating all possible index assignments for optimal mapping is infeasible (2 ! possibilities for code length b). We instead alternately optimize the objective function:

$$E = E_{\text{quan}} + \lambda E_{\text{aff}}$$

where quantization error is:

$$E_{\text{quan}} = \frac{1}{n} \sum_{x \in \mathcal{T}} \|x - c_{i(x)}\|^2$$

with i(x) being the codebook index for point x, c the corresponding codeword, and n the training set size.

Affinity error ensures codeword indices approximate Euclidean distances:

$$E_{\text{aff}} = \sum_{i=0}^{2^b-1} \sum_{j=0}^{2^b-1} w_{ij} \left(d(c_i, c_j) - d_h(i,j) \right)^2$$

where w = n n /n², n and n are sample counts for indices i and j, d(c ,c) is Euclidean distance, and d (i,j) is Hamming distance.

Optimization proceeds as follows: a) **Assignment step:** Fix codewords, optimize indices by assigning each point to its nearest codeword.
b) **Update step:** Fix indices, optimize codewords sequentially while holding others fixed:

$$c_j = \arg\min_{c_j} \sum_{x_i \in \mathcal{T}} \|x_i - c_j\|^2 + 2\lambda \sum_{i \neq j} w_{ij} \left(d(c_i, c_j) - d_h(i,j) \right)^2$$

We use a modified Hamming distance d (i,j) v · h¹ ²(i,j), where v is a constant initialized via PCAH and = 10. The process is illustrated in [Figure 4: see original paper].

2.1 Multi-Encoding

ISODATA may produce more clusters (K') than representable with b-bit codes (K = 2). For example, five clusters cannot be uniquely encoded with 2-bit codes (max 4 patterns). A naive solution groups centers into K clusters before encoding [Figure 5: see original paper], but this negates ISODATA' s benefits and reduces accuracy.

We adopt **multi-encoding** [16], assigning multiple binary codes per point and using average Hamming distance for retrieval:

$$d_L(X_i, Y_j) = \frac{1}{L} \sum_{l=1}^{L} d_h(X_i^{(l)}, Y_j^{(l)})$$

where L is the encoding multiplicity. For b-bit codes, L-fold encoding represents 2 patterns—double encoding yields 2² possibilities.

As shown in [Figure 6: see original paper], double hashing assigns two binary codes per point. Points (c , c) and (c , c) both have average Hamming distance 0.5, so querying c ' s neighbors returns both c and c .

2.2 Extension to Product Space

For large code lengths b, storing a 2 -sized codebook becomes infeasible. We partition the D-dimensional space into M subspaces, applying ISOH to each independently. The goal is balanced, independent subspaces with equal variance.

Using PCA [21], we sort principal components by decreasing eigenvalue. We distribute the top M components into M buckets, then iteratively assign remaining components to the bucket with smallest cumulative eigenvalue, ensuring each bucket contains D/M components. This balanced partitioning enables ISOH to generate large codebooks efficiently.

3.1 Datasets

We evaluate on three public datasets:

- **SIFT1M** [22]: 10 128-D SIFT features. We randomly sample 10 for training and 10^3 queries from 10 query points.
- **GIST** [22]: 5×10 training and 10^3 query points. We sample 10 training points.
- **CIFAR10** [23]: 6×10 GIST features extracted from images. We use 10^3 test and 10 training points.

3.2 Evaluation Metrics

We use **Recall** and **mean Average Precision (mAP)** [23,24]:

Recall measures the fraction of true nearest neighbors retrieved:

$$\text{Recall} = \frac{\#(\text{retrieved relevant points})}{\#(\text{all relevant points})}$$

mAP reflects retrieval quality—the higher the value, the faster true neighbors are returned:

$$\text{mAP} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{K_i}\sum_{j=1}^{K_i}\frac{j}{\text{rank}(j)}$$

where |Q| is query set size, K is the number of true neighbors for query i, and rank(j) returns the position of the j-th neighbor.

3.3 Results and Discussion

We compare ISOH with KMH, LSH, PCAH, SH, RR, and SGH using public implementations with default settings. Experiments use NN=10 Euclidean nearest neighbors as ground truth and code lengths B = 32, 64, 128 bits. Results are shown in [Figure 7: see original paper]–[Figure 9: see original paper].

Key findings: - ISOH consistently achieves the best performance across all datasets and code lengths.
- KMH performs well on SIFT but is surpassed by ISOH on all datasets.
- RR shows good performance only on SIFT at 128 bits.
- PCAH degrades as code length increases.
- SH remains moderate but never exceeds ISOH.
- SGH is competitive on CIFAR10 but weaker on SIFT.
- LSH performs well only on SIFT at 128 bits.

mAP values (Tables 1-3) confirm ISOH' s superiority. For SIFT1M at B=128 and NN=100, ISOH achieves 81.58% mAP, a 33.37% improvement over PCAH (48.21%). SH maintains moderate accuracy, while SGH performs well on GIST but poorly on SIFT.

ISOH' s advantages stem from: (1) farthest average distance initialization eliminating randomness, (2) adaptive ISODATA quantization enabling finer partitions, (3) multi-encoding overcoming fixed-length limitations, and (4) product spaces enabling efficient long codes. These innovations yield consistent performance gains across diverse datasets.

---

4 Conclusion

ISOH addresses KMH' s limitations through farthest average distance initialization, dynamic threshold setting, multi-encoding, and product spaces. This eliminates randomness, reduces training complexity, and enables more expressive codes at lower cost. Experiments on CIFAR10, GIST, and SIFT1M demonstrate ISOH' s superior nearest neighbor retrieval performance compared to KMH, RR, PCAH, LSH, SH, and SGH.

References

[1] Chen Tianlang, Xu Chenliang, Luo Jiebo. Improving text-based person search by spatial matching and adaptive threshold [C]// Proc of IEEE Winter Conference on Applications of Computer Vision. 2018.

[2] Li Wen, Duan Lixin, Xu Dong, et al. Text-based image retrieval using progressive multi-instance learning [C]// Proc of IEEE International Conference on Computer Vision. 2011: 2049-2055.

[3] Chen Shizhi, Yang Xiaodong, Tian Yingli. Discriminative hierarchical K-means tree for large-scale image classification [J]. IEEE Trans on Neural Networks & Learning Systems, 2017, 26(9): 2200-2205.

[4] Silpaanan C, Hartley R. Optimised KD-trees for fast image descriptor matching [C]// Proc of IEEE Conference on Computer Vision & Pattern Recognition. 2008: 1-8.

[5] Wang Jun, Liu Wei, Kumar S, et al. Learning to hash for indexing big data: a survey [J]. Proceedings of the IEEE, 2015, 104(1): 34-57.

[6] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions [C]// Proc of the 20th Symposium on Computational Geometry. 2004: 253-262.

[7] Shen Fumin, Zhou Xiang, Yang Yang, et al. A fast optimization method for general binary code learning [J]. IEEE Trans on Image Processing, 2016, 25(12): 5610-5621.

[8] Weiss Y, Torralba A, Fergus R. Spectral hashing [C]// Proc of International Conference on Neural Information Processing Systems. 2008: 1753-1760.

[9] Jiang Qingyuan, Li Wujun. Scalable graph hashing with feature transformation [C]// Proc of International Conference on Artificial Intelligence. 2015: 2248-2254.

[10] Gong Yunchao, Lazebnik S. Iterative quantization: a procrustean approach to learning binary codes [C]// Proc of Computer Vision & Pattern Recognition. 2011: 817-824.

[11] Gong Yunchao, Lazebnik S, Gordo A, et al. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2013, 35(12): 2916-2929.

[12] Jegou H, Douze M, Schmid C, et al. Aggregating local descriptors into a compact image representation [C]// Proc of CVPR, 2010, 238(6).

[13] Irie G, Arai H, Taniguchi Y. Alternating co-quantization for cross-modal hashing [C]// Proc of IEEE International Conference on Computer Vision. 2016: 1886-1894.

[14] He Kaiming, Wen Fang, Sun Jian. K-means hashing: an affinity-preserving quantization method for learning binary compact codes [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2013: 2938-2945.

[15] Ball G H, Hall D J. A novel method of data analysis and classification [M]//Data Analysis Machine Learning & Knowledge Discovery. 2003.

[16] Xia Yan, He Kaiming, Wen Fang, et al. Joint inverted indexing [C]// Proc of IEEE International Conference on Computer Vision. 2013.

[17] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2010, 33(1): 117-128.

[18] Gu Hongbo, Zhao WanPing. Clustering algorithm based on max-min distance for students' score analysis in universities and applications [J]. Journal of Hebei University of Engineering, 2010.

[19] He Yan, Ye Qiaolin, Liu Yingan, et al. The gepsvm classifier based on L1-norm distance metric [M]. 2016.

[20] Jiang Bo, Zhang Li. Research on minimum spanning tree based on prim algorithm [J]. Computer Engineering & Design, 2009, 30(13).

[21] Gupta A, Barbu A. Parameterized principal component analysis [J]. Pattern Recognition, 2018, 78(6): 215-227.

[22] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search [J]. IEEE Trans on Pattern Analysis & Machine Intelligence, 2011, 33(1): 117.

[23] Torralba A, Fergus R, Freeman W T. 80 Million tiny images: a large data set for nonparametric object and scene recognition [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2008, 30(11): 1958-1970.

[24] Fu Xiping, Mccane B, Mills S, et al. Nokmeans: non-orthogonal k-means hashing [C]// Proc of Asian Conference on Computer Vision. 2014: 162-177.

Note: Figure translations are in progress. See original paper for figures.